UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

MARCOS PAULO FERNANDES DE MOURA

MECANISMO DE DETECÇÃO DE ATAQUES DDOS NA CAMADA DE APLICAÇÃO

QUIXADÁ

2023

MARCOS PAULO FERNANDES DE MOURA

MECANISMO DE DETECÇÃO DE ATAQUES DDOS NA CAMADA DE APLICAÇÃO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Arthur de Castro Callado

Coorientador: Prof. Dr. Críston Pereira de Souza

QUIXADÁ

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M888 Moura, Marcos Paulo Fernandes de.
Mecanismo de detecção de ataques ddos na camada de aplicação / Marcos Paulo Fernandes de Moura. – 2023.
52 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2023.
Orientação: Prof. Dr. Arthur de Castro Callado.
Coorientação: Prof. Dr. Críston Pereira de Souza.

1. Segurança computacional. 2. Ataque de negação de serviço. 3. Redes de computadores. I. Título.
CDD 005

MARCOS PAULO FERNANDES DE MOURA

MECANISMO DE DETECÇÃO DE ATAQUES DDOS NA CAMADA DE APLICAÇÃO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Aprovada em: 30/11/2023

BANCA EXAMINADORA

Prof. Dr. Arthur de Castro Callado (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Críston Pereira de Souza (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

Prof. Dr. Eduardo Luzeiro Feitosa
Universidade Federal do Amazonas (UFAM)

A minha mãe Francisca Vania Fernandes, peça fundamental em minha formação de vida e acadêmica, exemplo de guerreira para mim.

AGRADECIMENTOS

Agradeço à Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) pela concessão de bolsas de pesquisa no Projeto Governo Digital do Estado do Ceará que foram fundamentais para a realização deste projeto.

Também gostaria de agradecer aos professores que participaram do grupo de pesquisa e que foram peças fundamentais do desenvolvimento desse projeto, Criston Pereira, Davi Romero, João Marcelo de Alencar e Regis Magalhães. Outra pessoa que fez parte desse mesmo grupo de pesquisa e foi uma peça vital para que esse trabalho fosse desenvolvido, me auxiliando tanto no âmbito acadêmico quando no pessoal foi o meu orientador Arthur Callado, muito obrigado.

A uma turma muito especial que esteve presente no meu dia a dia ao longo de toda essa jornada através do Discord (Sótemnoob), Matvin, Jeffin, Amorim, Lambari e Kadu. E a minha namorada por me aguentar nos estresses quando tava tudo dando errado rsrs, Luzia.

"Um espírito nobre engrandece o menor dos homens."

(Theodore Roosevelt)

RESUMO

Com o avanço da tecnologia, tanto empresas quanto governos precisam se preocupar mais com a segurança. Com desenvolvimento da tecnologia, seja em computação em nuvem, internet das coisas ou inteligência artificial, atacantes conseguem realizar ataques distribuídos de negação de serviços DDoS com menor custo e se tornam cada vez mais difíceis a detecção e a prevenção desses ataques. Hoje já existem diversos métodos utilizados para a detecção desses ataques, tais como: teste de Turing, métodos de aprendizado de máquina e abordagem focada na modelagem dos perfis de comportamentos normais do usuário. Tais métodos têm um bom nível de eficácia, entretanto também têm alguns problemas como interferir na qualidade de experiência do usuário ou ter alto custo computacional. Neste trabalho, é proposto um novo mecanismo para detecção de ataques DDoS na camada de aplicação usando uma técnica de avaliação estatística baseada no perfil de tráfego do serviço. Os resultados obtidos demonstraram que o mecanismo possui 1% de taxa de falsos positivos para tráfego regular e 0% de taxa de falsos negativos nos cenários testados.

Palavras-chave: DDoS; ataque distribuído de negação de serviços; camada de aplicação; segurança; redes.

ABSTRACT

As technology advances, both businesses and governments need to be more concerned about security. With the development of technology, whether in cloud computing, internet of things or artificial intelligence, attackers are able to carry out distributed denial of service attacks DDoS at a lower cost and it also becomes increasingly difficult to detect and prevent these attacks. Today there are already several methods used to detect these attacks, such as Turing test, machine learning methods and an approach focused on modeling normal user behavior profiles. Such methods have a good level of efficiency, however they also have some problems such as interfering with the quality of the user experience or having a high computational cost. In this work, a new mechanism for detecting DDoS attacks at the application layer is proposed using a statistical evaluation technique based on the service's traffic profile. The obtained results demonstrated that the mechanism has a 1% false positive rate for regular traffic and 0% false negative rate for the scenarios tested.

Keywords: DDoS; distributed denial of dervice attack; application layer; security; networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

DoS	*Denial of Service*
DDoS	*Distributed Denial of Service*
API	*Application Programming Interface*
SDN	*Software Defined Network*
IP	*Internet Protocol*
VM	*Virtual Machine*
HTTP	Hypertext Transfer Protocol

SUMÁRIO

1 INTRODUÇÃO

A cada dia, a tecnologia avança, modifica-se e cresce. Devido a esse desenvolvimento da tecnologia, seja em computação em nuvem, internet das coisas ou inteligência artificial, atacantes conseguem realizar ataques de negação de serviços *Denial of Service* (DoS) com menor custo. Além disso, torna-se cada vez mais difícil a detecção e a prevenção desses ataques. Tais ataques trazem grandes perdas tanto para empresas como para governos (ZHANG *et al.*, 2017). No primeiro semestre de 2022, foram registrados mais de 6 milhões de ataques DDoS globais, ocasionando indisponibilidade em diversos setores, como organizações de mídia online, empresas financeiras, provedores e empresas relacionadas a criptomoedas. Grande parte desses ataques teve como alvo entidades governamentais (CISO ADVISOR, 2022).

Ataques distribuídos de negação de serviço, do inglês *Distributed Denial of Service* (DDoS), representam uma forte ameaça para qualquer rede. Detecção de ataques DDoS é uma tarefa difícil, afinal os pacotes do ataque podem ser confundidos com pacotes legítimos e vice-versa. Outra dificuldade que pode ser vista é o grande número de pacotes a serem analisados, o que pode afetar tanto a precisão na detecção quanto o tempo de resposta ao ataque.

Há inúmeros mecanismos para combater ataques DDoS baseados em inundação, e esses mecanismos estão sempre em constante evolução (BAKSHI; DUJODWALA, 2010; DOU *et al.*, 2013; CHEN *et al.*, 2016; ISMAIL *et al.*, 2013). Esses ataques são facilmente identificados com base na assinatura dos pacotes ou no comportamento da rede, o que faz com que sejam detidos na entrada através do firewall. Entretanto, os ataques DDoS de baixa taxa se comportam similarmente à comunicação comum, iludindo os mecanismos de detecção (LI *et al.*, 2020). Esse tipo de ataque pode ser direcionado a um recurso ou serviço específico do sistema, ou aplicativo em questão, dessa forma reduzindo o custo do ataque.

Uma categoria de ataques de negação de serviços são os ataques DDoS à camada de aplicação. Diferente dos ataques DDoS direcionados à camada de rede e de transporte, esses podem ser de baixa taxa, sendo seu tráfego similar ao de um usuário comum, dificultando sua detecção (MANTAS *et al.*, 2015).

Jiang *et al.* (2017) citam três categorias de técnicas de defesa utilizadas para combater ataques DDoS na camada de aplicação. A primeira delas é uso do teste de Turing (por exemplo, um *captcha*) para diferenciar computadores e humanos automaticamente. Entretanto, esse método traz aos usuários encargos extras que são prejudiciais à qualidade de experiência no sistema. A segunda técnica emprega métodos de aprendizado de máquina para detectar anomalias.

Finalmente, a terceira abordagem foca na modelagem dos perfis de comportamentos normais do usuário.

O objetivo geral deste trabalho é propor um mecanismo capaz de detectar ataques DDoS na camada de aplicação em tempo real, utilizando uma das três técnicas citadas por (JIANG *et al.*, 2017), a avaliação estatística baseada no perfil do tráfego destinado a um serviço. Para isso o trabalho apresenta alguns objetivos específicos como: definir o perfil de tráfego real direcionado a um serviço ou aplicativo, definir como diferenciar o tráfego real de um tráfego de ataque e desenvolver o mecanismo de forma a ser utilizado em qualquer *Application Programming Interface* (API) Gateway sem afetar o desempenho do serviço a qual ele monitora. A definição desse perfil foi baseada no cálculo do primeiro dígito significativo definido na Lei de Benford (BENFORD, 1938), adaptado para a modelagem específica do tráfego da camada de aplicação. Nessa modelagem também foi incluída a frequência do zero ao cálculo do dígito mais significativo, pois para o tráfego ele representa o momento de não conexão. A partir disso são avaliadas as duas maiores frequências dos dígitos não nulos com o intuito de identificar uma anomalia no tráfego. Para isso foram realizados vários testes de aderência a distribuições para identificar possíveis caminhos para a detecção, isso será discutido melhor no Capítulo 4.

Cenários de experimentação preliminares também foram utilizados para definir os fatores, níveis e métricas a serem utilizados pelo mecanismo. Definindo variáveis como tamanho da coleta, tempo entre conexões e periodicidade de avaliações, todas elas voltadas para uma mesma origem.

O restante deste trabalho está dividido da seguinte maneira. O Capítulo 2 descreve a fundamentação teórica utilizada nesse estudo. O Capítulo 3 apresenta os trabalhos relacionados que serviram como base para este estudo. O Capítulo 4 descreve a metodologia utilizada. O Capítulo 5 apresenta os resultados obtidos. O Capítulo 6 apresenta as conclusões e os trabalhos futuros. E ao final, são listadas as Referências.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a base teórica que foi utilizada para o desenvolvimento deste trabalho.

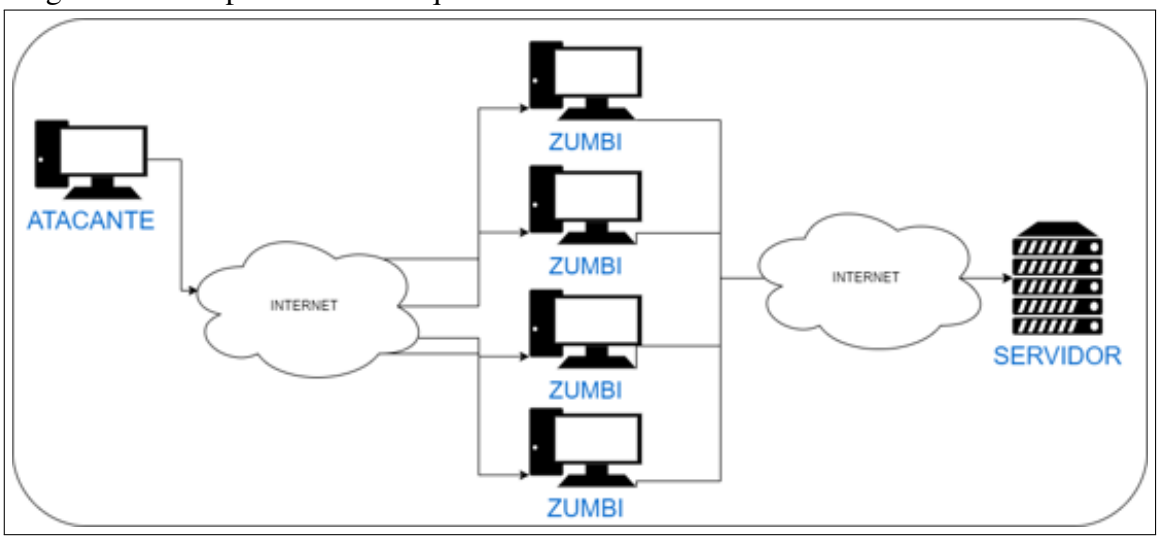
2.1 Ataques de Negação de Serviço

Ataques de negação de serviços DoS são um problema sério tanto para aplicações *Web* quanto para outros serviços de rede, onde o atacante tenta tornar o serviço inacessível para os usuários legítimos. Com o foco na exaustão de recursos, os ataques DoS geram um amplo espectro de variantes capazes de esgotar os recursos em qualquer camada da arquitetura TCP/IP tradicional (MANTAS *et al.*, 2015). No ataque de negação de serviço DoS, uma única fonte executa o ataque, enquanto no DDoS são utilizados vários *hosts* para atacar um sistema.

2.1.1 Ataques de Negação de Serviço Distribuídos

A Figura 1 apresenta o comportamento de um ataque DDoS, onde um atacante utiliza a internet como meio de acesso para se comunicar com as máquinas zumbis, criando uma distância que dificulta a identificação do atacante a partir da vítima/servidor.

Figura 1 – Comportamento Ataque DDoS



Fonte: elaborado pelo autor (2021).

No processo, o atacante primeiro compromete um grande número de sistemas criando uma rede para realizar o ataque. Esses dispositivos, chamados de zumbis, podem ser alugados ou infectados pelo atacante. Podendo ser esses dispositivos de usuários inocentes, que se quer

sabem que participam da execução do ataque. A partir desses zumbis a rede é inundada por inúmeros pacotes falsos enviados. Esse tipo de ataque tem como objetivo esgotar rapidamente o poder computacional ou a capacidade de comunicação de um alvo, inundando-o com um grande volume de tráfego malicioso (RAI; CHALLA, 2016).

Zhang *et al.* (2017) diferencia ataques DDoS em relação ao volume e ao tempo. Os ataques podem ter enormes volumes de tráfego em curto espaço de tempo, baixos volumes de tráfego em tempo longo e enormes volumes de tráfego em tempo longo. Enquanto ataques de alto volume de tráfego são facilmente detectáveis por ferramentas como Akamai Prolexic e AWS Cloudflare, os ataques de baixo volume são bem mais difíceis de identificar (BAARZI *et al.*, 2020).

2.1.2 Ataques DDoS na camada de aplicação

Para Mantas *et al.* (2015), embora ataques direcionados à camada de rede e a transporte, como ataques de inundação ainda sejam um método de interrupção viável, os ataques DoS modernos raramente precisam exceder a capacidade de taxa de transferência da rede, pois o destino geralmente falha muito antes da capacidade ser atingida. Isso deu um impulso ao desenvolvimento de ataques DoS da camada de aplicação. Tais ataques conseguem afetar aplicativos ou serviços específicos sem necessariamente causar danos aos recursos da rede, dificultando sua detecção.

Ataques DDoS na camada de aplicação buscam exaurir recursos computacionais como ciclos de CPU, memória, capacidade da pilha TCP/IP, recursos de dispositivos de entrada e saída, dentre outros. A discrição desses ataques é uma de suas características mais problemáticas, pois os tornam praticamente indistinguíveis do tráfego de usuários legítimos (MANTAS *et al.*, 2015; BEITOLLAHI; DECONINCK, 2014).

2.1.3 Ataque Slowloris

O Slowloris é um dos mais populares ataques DDoS. O ataque busca explorar o protocolo HTTP estabelecendo uma grande quantidade de solicitações pendentes com o servidor WEB alvo (SABRI *et al.*, 2021). O Slowloris é categorizado como um ataque de baixa taxa.

O ataque Slowloris. Uma solicitação HTTP GET é enviada ao servidor. Esta solicitação não é encerrada de forma válida devido à ausência do caractere de finalização \r \n (quebra de linha dupla). O servidor aguarda então pela próxima parte da solicitação, que

conterá um caractere de terminação. Esta espera é limitada na configuração do servidor. Após o cronômetro ser excedido, o servidor fecha a conexão TCP. Porém, antes que esse tempo expire, o ataque envia outra parte da solicitação. Este pacote *keep-alive* geralmente contém apenas alguns caracteres aleatórios. Este pacote zera o cronômetro e então o invasor é silenciado novamente. Então, todo o processo é repetido (SIKORA *et al.*, 2021; MURALEEDHARAN; JANET, 2017).

2.1.4 Ataque SlowPOST

O Slowpost ou como também é conhecido RUDY(R U Dead Yet) usa uma solicitação HTTP POST. Esse tipo de solicitação geralmente é utilizado para enviar dados preenchidos em formulários. O cabeçalho da solicitação HTTP contém o campo *Content-Length*, que especifica o tamanho dos dados transmitido após o cabeçalho da solicitação. Neste ataque, o campo *Content-Length* contém um valor muito alto, o que significa que o servidor esperará receber uma grande quantidade de dados. O cabeçalho da solicitação falsificada é então encerrado de forma válida e a solicitação contém um pequeno pedaço de dados, que geralmente é representado por alguns caracteres aleatórios. O invasor então espera e envia outro pequeno dado antes que o temporizador de encerramento da conexão expire. Dessa forma, o invasor mantém a conexão ativa e da mesma forma tenta estabelecer o maior número possível. Isso leva ao esgotamento de todos os recursos disponíveis do servidor (SIKORA *et al.*, 2021; MURALEEDHARAN; JANET, 2017).

2.1.5 Ataque SlowREAD

O ataque Slowread usa protocolos HTTP e TCP. No início, um invasor solicita alguns dados maiores, como uma imagem, enviando uma solicitação GET válida ao servidor. O ataque define o parâmetro *window-size* no cabeçalho TCP para um valor de janela TCP muito baixo. Este parâmetro determina a quantidade de dados que o servidor pode enviar sem qualquer confirmação. O servidor é forçado a enviar uma resposta em partes muito pequenas. Dessa forma, o ataque pode chegar a um estado em que a transferência de um arquivo de 1 MB pode levar vários dias (SIKORA *et al.*, 2021; MURALEEDHARAN; JANET, 2017).

2.1.6 Ataque Apachekiller

Oo ataque Apachekiller aproveita a maneira como o servidor Apache lida com as solicitações de intervalo HTTP. O ataque reestabelece conexões enviando um grande número de intervalos de bytes sobrepostos. Quando o servidor web recebe solicitações de intervalo, ele cria partes de uma resposta de várias partes para cada intervalo especificado. Neste ataque o número de intervalos de bytes sobrepostos é maior, o que utiliza excessivamente a memória e a CPU do servidor e, posteriormente, esgotará os recursos do servidor e causará a indisponibilidade do serviço (MURALEEDHARAN; JANET, 2017).

Tabela 1 – Porcentagens dos primeiros dígitos da Lei de Benford

Primeiro dígito	1	2	3	4	5	6	7	8	9
Porcentagem(%)	30.1	17.6	12.5	9.7	7.9	6.7	5.8	5.1	4.8

Fonte: elaborada pelo autor.

Segundo Prandl *et al.* (2017) os tempos de chegada de pacotes SYN em um servidor tem conformidade com a lei de Benford, e sugere que a conformidade, ou não, com essa lei pode ser utilizada para detectar anomalias na rede.

2.2 API Gateway

Red Hat (2021) define API *Gateway* como uma ferramenta de gerenciamento de APIs que fica entre o cliente e uma coleção de serviços de *back-end*. O *Gateway* é um *proxy* que intermedeia requisições para sistemas internos. É como um portão de entrada para serviços de um sistema, oferecendo diversas funcionalidades relacionadas ao acesso de uma API. Como prova de conceito, a ferramenta proposta neste estudo foi desenvolvida como um *plugin* para o Kong API Gateway. Kong é um *Gateway* de API escrito na linguagem Lua (ORG., 2023) integrado ao servidor Nginx(NGINX, 2023). Atravéz de *plugins* ele pode customizar o proxy do Nginx para, ao encaminhar requisições, prover, com bom desempenho, funcionalidades como autenticação, controle de tráfego, informações analíticas, logs, dentre outras funcionalidades (KONG, 2022).

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados ao tema, que serviram como base inicial de pesquisa e estudo. Na seção 4.2 descrevemos como foram identificados os trabalhos relacionados a este estudo.

3.1 DDoS attack detection using fast entropy approach on flow-based network traffic

Em David e Thomas (2015) os autores propõem uma melhoria da detecção de DDoS com base no método de entropia rápida baseado em solicitações por fluxo, que envolve o cálculo da média e do desvio padrão da contagem de fluxo em um determinado intervalo de tempo. É utilizado um algoritmo adaptativo para o cálculo do limiar da variação de entropia, pois as métricas da rede e o comportamento do usuário variam ao longo do tempo. O método de detecção proposto pelos autores é baseado em três objetivos:

1. Agregação de fluxo para detecção de ataque baseada em fluxo;
2. Cálculo de entropia rápida para detectar ataques DDoS com menos tempo computacional;
3. Algoritmo de limite adaptativo para melhorar a precisão de detecção.

A entropia se trata de uma medida que representa a homogeneidade de um conjunto de dados. A entropia varia entre 0 e 1, sendo 0 um conjunto totalmente não entrópico e por tanto homogêneo. Já um conjunto em que todas as possibilidades contidas tem a mesma probabilidade, a entropia é máxima. A métrica de entropia é comumente utilizada em teoria da informação, uma aplicação é no treinamento de modelos de aprendizado de máquina baseados em árvores de decisão. Neste artigo o autor usa a entropia do conjunto de dados de contagem de fluxo para detectar anomalias, como será descrito a seguir.

Os autores tratam a agregação de fluxo como uma série unidirecional de pacotes de um determinado protocolo viajando entre uma origem e um destino com um par IP/porta durante um período. A entropia rápida da contagem de fluxo é calculada para cada conexão. Havendo um ataque, a entropia cai, pois uma faixa de fluxo está dominando. Em caso de ausência de ataque, a entropia se mantém em uma faixa constante. Os autores consideram o valor limite de fluxo algo importante para a detecção de ataques de inundação, sendo necessário que o mesmo seja ajustado conforme o valor da entropia.

A partir da avaliação de desempenho do mecanismo utilizando o conjunto de dados

CAIDA, disponível em Shirsath (2007) , os autores identificaram que o valor de entropia rápida é consideravelmente reduzido quando uma conexão tem um alto volume de fluxo em um espaço de tempo. O ataque DDoS é detectado quando a diferença entre a entropia da contagem de fluxo em cada instante e o valor médio da entropia naquele intervalo de tempo é maior que o valor limite. Como o valor do limite é atualizado de forma adaptativa com base na condição do padrão de tráfego, a precisão da detecção é aprimorada.

No trabalho citado, os autores buscam detectar ataques DDoS de inundação utilizando entropia rápida, o que faz com que o mesmo não tenha uma boa eficácia com ataques de baixo fluxo. Neste trabalho é proposto um mecanismo de detecção para ataque de baixo fluxo com foco na camada de aplicação.

3.2 Early detection of DDoS attacks against SDN controllers

Mousavi e St-Hilaire (2015) tratam o controlador *Software Defined Network* (SDN) como um ponto único de gerência e buscam mitigar ataques DDoS utilizando as características dele. O objetivo principal dos autores é conseguir detectar o ataque em seu estágio inicial, ou seja, enquanto o ataque ainda está em centenas de pacotes. Como o mecanismo funciona dentro do controlador SDN, é necessário que além de rápido e eficaz, o mecanismo seja leve, para evitar consumo excessivo de processamento, especialmente no pico de um ataque.

A solução proposta utiliza a entropia para medir a possibilidade de ataques, por exemplo, em uma rede de 64 *hosts*, todos os *hosts* devem ter uma probabilidade próxima de receber novos pacotes. Isso resultará em alta entropia. Se um ou mais hosts começar a receber pacotes de entrada excessivos, a aleatoriedade diminui e a entropia cai. Os autores utilizam essa propriedade para detectar os possíveis ataques. A coleta de dados para o cálculo da entropia é realizada adicionando código a coleta de dados padrão do controlador ao *switch* SDN, coletando o IP destino dos pacotes para o cálculo da entropia. A principal razão da utilização da entropia para a detecção de ataques DDoS é a sua capacidade de medir a aleatoriedade.

Os autores utilizaram o Mininet (MININET, 2022) para simular a rede SDN e o controlador POX. Seu modelo de detecção apresenta sucesso nos primeiros 250 pacotes capturados, com uma taxa de acerto de 96%. No artigo citado, os autores propõem uma solução para SDN, enquanto neste artigo é proposto uma solução que possa ser acoplada a qualquer tipo de rede.

3.3 DDoS detection and prevention based on artificial intelligence techniques

Em Zhang *et al.* (2017) os autores defendem que os ataques DDoS ainda trazem grandes perdas tanto para empresas como para governos. Com o desenvolvimento de tecnologias como computação em nuvem, internet das coisas e inteligência artificial, os atacantes conseguem realizar ataques DDoS com menor custo e também se tornam mais difíceis a detecção e a prevenção desses ataques.

Os autores definem que os ataques DDoS podem ser enormes volumes de tráfego em curto espaço de tempo, baixos volumes de tráfego em longo tempo, enormes volumes de tráfego em longo tempo e consideram que o último traz maior dificuldade de identificação por se parecer com o tráfego normal da rede. No artigo são apresentadas técnicas de inteligência artificial mostradas em pesquisas anteriores para apresentar as principais técnicas de inteligência artificial para detecção de ataques DDoS. Através desse estudo os autores também definem que as métricas utilizadas para detectar ataques DDoS são: número de pacotes, tamanho médio do pacote, variação do intervalo de tempo, variação do tamanho do pacote, número de bytes, taxa de pacote e taxa de bits.

Por fim, os autores identificam que entre as técnicas de inteligência artificial apresentadas, a floresta aleatória e *Naive Bayes* são as mais eficientes para classificar o tráfego entre tráfego malicioso e tráfego normal. Vários algoritmos de aprendizagem de máquina combinados podem detectar ataques DDoS com melhor acurácia e desempenho. Neste trabalho é proposto um mecanismo que consiga identificar ataques DDoS utilizando uma análise estatística baseada no fluxo.

3.4 An Investigation of Power Law Probability Distributions for Network Anomaly Detection

Em Prandl *et al.* (2017) os autores identificam que os tempos de chegada entre pacotes SYN estão em conformidade com a lei de Benford, prevendo a frequência dos dígitos iniciais em coleções de números que ocorrem naturalmente. Sugerindo assim que a conformidade ou não conformidade com essa lei poderia ser utilizada para detectar anomalias na rede. Eles também definem as seguintes afirmações para tráfego de rede, primeiramente verificam que a conformidade com a lei de tempos entre chegadas de Benford também é verdadeira para certos tipos de pacotes TCP e UDP. Em segundo, afirmam que o tamanho do pacote também pode ser

outra alternativa para os tempos entre chegadas, com a vantagem de seguir as leis de Benford e Zipf que descreve a relação entre o número de ocorrências de uma palavra em um texto e sua posição no ranking de frequência das palavras daquele texto. Um exemplo hipotético seria um texto no qual a palavra mais recorrente aparece 1000 vezes, é esperado que a segunda palavra ocorra com metade da frequência da primeira, a terceira com um terço e assim por diante. Esta por tanto é frequentemente utilizada na análise de linguagem natural. E por fim, exploram a aplicação potencial de leis de potência na detecção específica de ataques de negação de serviço DoS usando tempos entre chegadas e comprimento de pacote.

No trabalho citado, os autores usam a Lei de Benford e a Lei de Zipf para analisar a chegada de pacotes em uma rede, buscando detectar anomalias utilizando o teste estatístico de Watson que é um teste estatístico usado para avaliar a adequação de uma distribuição de probabilidade a um conjunto de dados. Já neste trabalho, a proposta é utilizar o cálculo do dígito mais significativo da lei de Benford na frequência de chegada de conexões de rede de uma origem a um servidor específico, mas usando uma proporção adequada ao tráfego de rede.

3.5 Concept of Intelligent Detection of DDoS Attacks in SDN Networks Using Machine Learning

Em Klymash *et al.* (2020) é apresentado um conceito de detecção inteligente de ataques DDoS em redes SDN usando aprendizado de máquina. Os autores propõem um método de análise de logs para ensinar o controlador SDN a detectar ataques e criar regras apropriadas para bloqueá-los. Eles usam a abordagem de Kullback-Leibler para detectar anomalias de fluxo ao longo do tempo da sessão. O artigo também menciona outras propostas de IDS - *Intrusion Detection System* baseados em *Lachine Learnig* em diferentes cenários de rede.

A medida de divergência de Kullback-Leibler foi utilizada para calcular a diferença entre a distribuição de tráfego normal com a distribuição de tráfego anômalo, quanto maior a divergência, maior a diferença entre as distribuições e maior a probabilidade de que o tráfego seja anômalo.

Os autores utilizaram o conjunto de dados KDD99 para simular o modelo de detecção de ataques proposto no artigo. Eles usaram o Tensorflow para implementar o modelo de detecção de ataques baseado em aprendizado profundo com uma Máquina de Boltzmann Restrita (RBM). O modelo proposto pelos autores alcançou uma precisão de 94% na detecção de ataques.

No trabalho citado, os autores utilizaram aprendizado de máquina para detectar

ataques DDoS em redes SDN. Neste trabalho a proposta é utilizar uma análise estatística para o mesmo e a ferramenta pode ser acoplada a qualquer tipo de rede.

3.6 Comparação entre artigos

Nesta seção é apresentado o Quadro 1 que lista a comparação das metodologias utilizadas e objetivos de cada trabalho relacionado e neste estudo.

Quadro 1 – Comparação entre metodologias e objetivos dos artigos.

Artigo	Método	Objetivo	Conjunto de dados
David e Thomas (2015)	Entropia rápida	Ataques DDoS de inundação	CAIDA
Mousavi e St-Hilaire (2015)	Probabilidade de chegada de pacotes em um controlador POX (SDN)	Ataques DDoS em redes SDN	Ferramenta Scapy
Zhang *et al.* (2017)	Algoritmos de IA	Ataques DoS e DDoS	DARPA, CAIDA e LLS-DDoS
Prandl *et al.* (2017)	Lei de Benford juntamente com a Lei de Zipf e teste de Watson	Ataques DoS ou DDoS	MAWI, Conjunto de Dados Capturados na rede da *Curtin University* e ataques gerados pelos autores
Klymash *et al.* (2020)	Aprendizado de máquina e avaliação de distribuição	Ataques DDoS em redes SDN	KDD9
Este estudo	Adaptação do cálculo do dígito mais significativo proposto na Lei de Benford utilizando uma frequência diferente da proposta na mesma	Ataques DoS ou DDoS na camada de aplicação	Conjunto de dados obtidos no servidor PEC-ESUS na Secretaria Municipal de Quixadá e ataques gerados pela ferramenta slowhttptest

Fonte: elaborada pelo autor.

4 METODOLOGIA

Neste capítulo, são apresentados os passos realizados no desenvolvimento deste estudo. Inclui-se a definição da proposta, pesquisa bibliográfica, definições dos cenários de experimentação, captura do tráfego real e elaboração do tráfego de ataque, desenvolvimento do algoritmo, calibração de variáveis e análise dos resultados obtidos. A Figura 2 apresenta o diagrama da metodologia definida. As setas verdes indicam o avanço do passo, enquanto a seta vermelhas indicam a necessidade de retornar ao passo anterior.

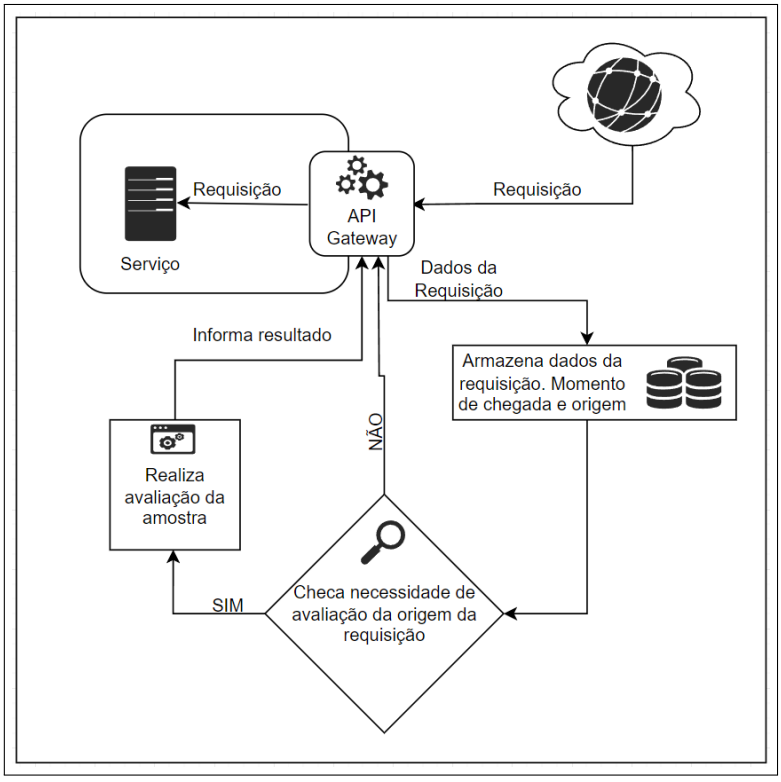
Figura 2 – Diagrama da Metodologia

Fonte: elaborado pelo autor (2023).

4.1 Definição da Proposta

A Figura 3 apresenta o diagrama de funcionamento do *plugin*. O mecanismo foi desenvolvido de forma a não afetar o funcionamento do serviço ao qual ele monitora, funcionando em paralelo ao API Gateway. A coleta dos dados é realizada medindo a quantidade de requisições de uma mesma origem dentro de um intervalo de tempo. A origem é identificada pelo seu endereço *Internet Protocol* (IP), sendo armazenado o momento de chegada das requisições vindas da origem. Quando a amostra de uma origem chega ao tamanho determinado é verificada a necessidade de avaliação daquela amostra. A ideia é que caso a quantidade de requisições seja muito pequena, pode se tratar de um tráfego comum ou em caso de um ataque o mesmo não tem um volume significativo e pode ser ignorado. Caso haja a necessidade de avaliação, a mesma é realizada e o resultado é informado ao API Gateway.

Figura 3 – Diagrama de funcionamento do *plugin*.



Fonte: elaborado pelo autor (2023).

O mecanismo proposto neste trabalho visa detectar ataques DDoS na camada de aplicação em tempo real. Para isso, utiliza uma técnica de avaliação estatística baseada no perfil do tráfego destinado a um serviço ou aplicativo. A definição desse perfil ocorre por meio da análise da frequência relativa do primeiro dígito mais significativo, conforme definido na Lei de Benford. Esta técnica foi escolhida devido a sua simplicidade de funcionamento, capacidade de

adaptação e por ser citada em diversos trabalhos de pesquisa.

O método requer a definição de um perfil de tráfego comum, identificado a partir de experimentos no conjunto de dados. A partir disso, a avaliação mede se a amostra está dentro desse perfil definido. A avaliação do tráfego aplica o cálculo do primeiro dígito significativo à amostra coletada de intervalos de requisições, incluindo a frequência do zero e somando as duas maiores frequências dos dígitos não nulo. Se o valor somado dessas frequências for maior que o limite definido, ele é identificado como um tráfego anômalo. O valor de limite definido foi de 55%. A definição de como esses parâmetros foram definidos é discutido na seção 4.6, onde são apresentados os experimentos realizados, com o ajuste de variáveis e modificações no método de definição do perfil.

4.2 Pesquisa Bibliográfica

Inicialmente, foi realizada uma pesquisa bibliográfica buscando o tópico de ataques DDoS na camada de aplicação. As bases de dados definidas para pesquisa foram IEEE e ACM. Ambas foram escolhidas por concentrar a maior quantidade de trabalhos sobre o tema e por outras apresentarem repetições de artigos presentes nelas. As palavras-chave utilizadas foram “DDoS”, “*Distributed Denial of Service*”, “DoS”, “*Denial of Service*”, “*application layer*”, “*mitigate*”. Aplicando as palavras-chave juntamente com os conectores AND e OR foram montadas as *strings* de busca. As *Strings* de busca utilizadas nas respectivas bases foram:

- (“detection”) AND (“Application Layer” OR “DDoS Attack” OR “DoS Attack"OR "Low Rate”)
- (“mitigate”) AND (“Application Layer” OR “DDoS Attack” OR “DoS Attack” OR “Low Rate”)
- “DdoS Attack” OR “Defense DdoS Attack” OR “Distributed Denial-of-Service” OR “DdoS”

Com essa pesquisa foram encontrados aproximadamente 3 mil trabalhos. Utilizando critérios de inclusão e exclusão como faixa de ano de publicação, definida como artigos a partir de 2015, e palavras-chave como “microsserviços”, “DDoS”, “camada de aplicação”, removendo também artigos que possuíam os termos, mas não tratavam diretamente do assunto buscado, esse valor foi reduzido. Após passar pelos critérios de inclusão e exclusão e por uma leitura preliminar dos artigos restantes dessa pesquisa, foi possível reduzir esse número para aproximadamente 387 artigos, a partir dos quais foram selecionados os 5 trabalhos relacionados e 16 estudos

auxiliares. Os trabalhos foram escolhidos por apresentarem maior contribuição de conhecimento e relevância para este estudo. Alguns trabalhos podem ser considerados como antigos por terem mais de 5 anos, entretanto eles se mostraram mais completos que trabalhos mais recentes e também o fato dos mais recentes os citarem em seus textos.

4.3 Definição dos Cenários de Experimentação

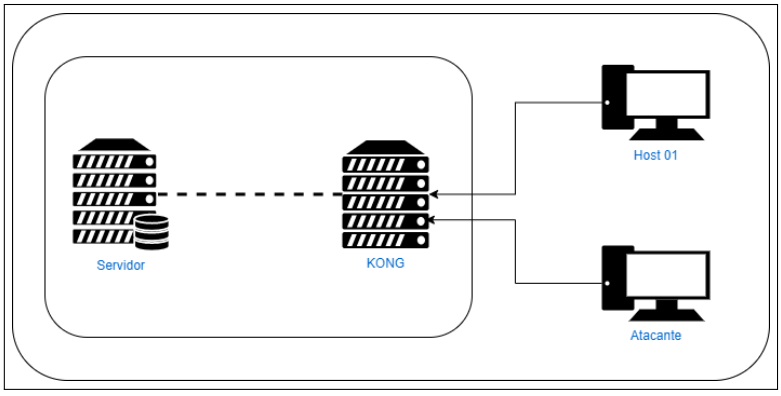
O cenário de experimentação deste trabalho foi desenvolvido utilizando o Kong API Gateway. Por estar na borda da conexão entre servidor e clientes, todo o tráfego passa por ele, o que nos permite capturar e analisar todo tráfego destinado ao servidor. O cenário foi montado utilizando *Virtual Machine* (VM), rodando o sistema operacional Linux Ubuntu 20.04 com a seguinte configuração:

- VM que hospeda o Kong Gateway e o serviço acessado - 4 CPUs, 16GB de RAM e 24GB de armazenamento;
- 2 VMs de tráfego e ataque - 2 CPUs, 2GB RAM, 10GB armazenamento.

A Figura 4 apresenta a topologia de rede utilizada, com os seguintes componentes:

1. Servidor - componente que é alvo dos ataques;
2. KONG - componente com o API Gateway Kong onde é instalado o mecanismo de detecção em formato de *plugin*;
3. Host 01 - componente responsável por gerar tráfego real (acesso legítimo ao servidor);
4. Atacante - componente responsável por gerar o tráfego de ataque.

Figura 4 – Cenário



Fonte: elaborado pelo autor (2021).

4.4 Tráfego

4.4.1 Captura do Tráfego Real

O conjunto de dados de tráfego real utilizado neste experimento foi obtido através uma coleta de tráfego de rede realizada no servidor PEC-ESUS da Secretaria Municipal de Saúde de Quixadá em 2020. PEC-ESUS é um sistema de portuário eletrônico, desenvolvido pelo governo para cadastro de atividades em uma UBS — Unidade Básica de Saúde e atividades realizadas pelos ACS — Agentes Comunitário de Saúde. O trecho de tráfego coletado corresponde a aproximadamente 480h de tráfego, contento um total de 1.165.748 requisições ao servidor. Os dados foram coletados em dias úteis, já que o uso dos serviços disponíveis está atrelado à jornada de trabalho dos servidores municipais.

Durante o período nenhuma anomalia de tráfego, tentativa de invasão ou falhas na rede foram detectadas. Por tanto, os dados reais são considerados em sua totalidade tráfego não malicioso. O serviço era acessado tanto por dispositivos internos como externos, e também por um grupo de máquinas que estava sob NAT (*Network Address Translation*).

4.4.2 Elaboração do Tráfego de Ataque

O tráfego de ataque utilizado neste experimento foi gerado sinteticamente com a ferramenta *Slowhttptest* (KALI ORG., 2022) com quatro configurações distintas, tais configurações foram obtidas a partir da documentação disponível na documentação da ferramenta:

1. slowhttptest -H -c 2000 -g -o output -i 10 -r 300 -t GET -u **url** -x 24 -p 3
   O primeiro cenário se trata de um ataque Slowloris com 300 conexões por segundo até um total de 2000. Essas conexões fazem requisições do tipo GET do protocolo Hypertext Transfer Protocol (HTTP) e enviam novos dados a cada 10 segundos para evitar desconexão. Após 3 segundos sem resposta o servidor é considerado inacessível.
2. slowhttptest -c 3000 -B -g -o output -i 110 -r 200 -s 8192 -t FAKEVERB -u **url** -x 10 -p 3
   O segundo cenário é um ataque SlowPOST. Esse faz 200 conexões por segundo até o total de 3000. O conteúdo da requisição é de 8192 bytes e cabeçalhos que se seguem a cada 110 segundos. Estes cabeçalhos, tem tamanho limitado a 10 bytes. Após 3 segundos sem resposta o servidor é considerado inacessível.
3. slowhttptest -R -u **url** -t HEAD -c 1000 -a 10 -b 3000 -r 500 -g -o output
   O terceiro cenário se trata de um Apachekiller, com 500 conexões por segundo até o total

de 1000 conexões. O intervalo requisitado no cabeçalho para gerar sobreposição e por tanto consumo de processamento começa em 10 e vai até 3000.

4. slowhttptest -c 8000 -X -g -o output -r 200 -w 512 -y 1024 -n 5 -z 32 -k 3 -u **url** -p 3
   O quarto cenário se trata de um SlowRead com 200 conexões por segundo até um total de 8000. O valor inicial do intervalo da janela TCP é de 512 e o final de 1024. Cada conexão faz até 3 requisições iguais. Após 3 segundos sem resposta o servidor é considerado inacessível.

Cada comando apresentado define uma configuração distinta de ataque, variando rajadas de conexão, tempo de espera de cada conexão, quantidade de conexões, dentre outros parâmetros. A configuração é definida a partir dos seguintes parâmetros de configuração:

– -H - ataque no padrão *slowloris*, ataque na camada de aplicação com requisições parciais;
– -R - ataque no padrão *Apache Killer*, onde são enviados dados maliciosos dentro do cabeçalho, com intervalo de x a y;
– -B - padrão slow POST (definido por enviar uma requisição POST incompleta, e atrasa severamente o envio dos dados restantes);
– -X - padrão slow Read (definido por abrir uma conexão de leitura com o servidor, mas não fecha-la, mantendo-a aberta ocupando o canal);
– -a - início do intervalo do padrão -R;
– -b - fim do intervalo do padrão -R;
– -c - quantidade de conexões realizadas;
– -g - gera estatísticas de *socket* do ataque;
– -o - arquivo de saída onde as estatísticas são salvas;
– -i - intervalo entre conexões geradas em segundos;
– -r - quantidade de conexões por segundo;
– -t - padrão de requisições enviadas;
– -u - url a ser usada no ataque;
– -x - comprimento máximo de cada par de nome/valor aleatório gerado;
– -p - tempo de espera para resposta;
– -s - valor do cabeçalho;
– -w - tamanho inicial do valor aleatório da janela TCP;
– -y - tamanho final do valor aleatório da janela TCP;
– -k - total de solicitações em espera para cada conexão.

Nos cenários de experimentação também foram realizados ataques utilizando as configurações dos comandos 1 e 2, onde as rajadas de conexões seguiam uma distribuição normal. Isso foi realizado com o intuito de enganar a ferramenta variando o padrão de conexão buscando deixar o tráfego de ataque similar ao tráfego real.

4.5 Desenvolvimento do Algoritmo

O algoritmo proposto neste trabalho é dividido em duas partes: coleta e avaliação. Definimos algumas variáveis para o funcionamento desse algoritmo:

- Variáveis de entrada:
  - CollSize - Tamanho da amostra de BINs a serem analisados;
  - TB - Tempo de um BIN em segundos.
- Variáveis locais:
  - Lista de BINs - Quantidade de requisições de uma mesma origem dentro de um intervalo de tempo, armazenados em uma lista;
  - TIP - Momento do primeiro BIN de uma origem;
  - NBA - Quantidade de BINs armazenados de uma origem;
  - TAV - Teste de nova avaliação;
  - FNA - Fator de nova avaliação;
  - connectionMoment - Armazena o momento da conexão da origem;
  - NBA - Quantidade de BINs já armazenado de uma origem.

CollSize é um valor configurável que define a quantidade de BINs que deverão ser armazenados de uma origem. **BIN** é um contador de conexões dentro de um tempo **TB**. **TB** é o tempo de um BIN em segundos, a cada **TB** segundos deverá ser gerado um novo BIN para aquela coleta. **FNA** é o fator de uma nova avaliação utilizado para definir quando realizar uma nova avaliação de uma mesma origem, isto é, a cada **FNA** conexões o algoritmo avalia novamente os dados coletados. **FNA** tem efeito sobre a **TAV**, que é o teste de uma nova avaliação, a variável **TAV** guarda a contagem de conexões para a próxima avaliação definida em **FNA**. A variável **connectionMoment** armazena o momento da conexão da origem a ser armazenada em *timestamp*.

A Figura 5 apresenta o diagrama de fluxo da coleta do algoritmo. Ela apresenta a lógica utilizada pela ferramenta para realizar a coleta dos dados de conexão com o API *Gateway*. Sendo assim, o trecho do algoritmo não apresenta a lógica de classificação do tráfego como

sendo ou não malicioso. A função de avaliação será descrita mais a frente.

Ao ser acionado, o mecanismo realiza o armazenamento da variável **connectionMoment** recebendo o momento da conexão da origem em *timestamp*. Em seguida, entra na primeira condição onde, se não houver um registro prévio de conexão da origem, este: cria o primeiro **BIN** com valor 1; insere o valor de **connectionMoment** em **TIP** que é o tempo do primeiro BIN da origem; cria o **TAV** da origem; por fim encerra o funcionamento do mecanismo. Caso contrário, se existir registro prévio da origem, cria o **NBA** da origem com o tamanho da lista de **BIN**s e passa para a próxima condição.

A segunda condição verifica se o momento da conexão é maior que o tempo limite defino por **TIP + TB** * **NBA**, se sim, ele passa para a próxima condição. Se não for, ele incrementa o último BIN armazenado da origem e encerra o mecanismo.

A terceira condição verifica se o **NBA** é maior que o **CollSize**, se sim, ele passa para a próxima condição. Se não ele entra no bloco de laço do mecanismo que será explicado mais a frente.

A quarta condição verifica se a soma dos **BIN**s é maior que o **CollSize**, se sim passa para a próxima condição. Se não passa para o bloco de laço.

A quinta condição verifica se o **TAV** é igual ao **FNA**, se sim, ele chama a função de avaliação da amostra, insere o valor zero no contador **TAV** e entra no bloco de laço. Se não, ele incrementa o **TAV** e entra no bloco de laço.

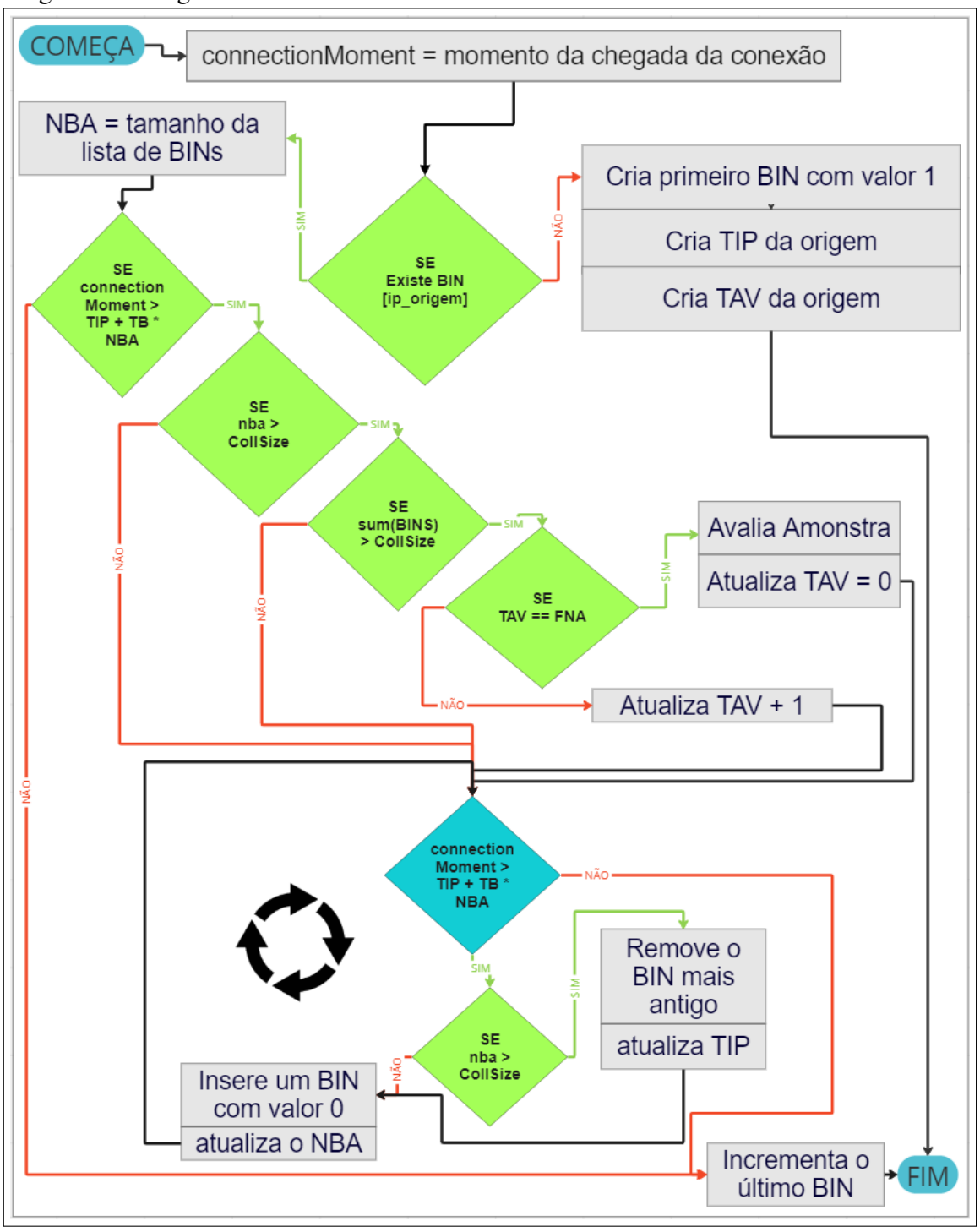
O bloco de laço é responsável por fazer o controle da lista de **BIN**s armazenados. Enquanto o momento da conexão for maior que o tempo limite definido por **TIP + TB** * **NBA**, ele realiza as seguintes instruções: verifica se o **NBA** é maior que o **CollSize**, se sim, remove o **BIN** mais antigo armazenado da origem, atualiza o **TIP** da origem, se não, não realiza nenhuma ação. Em sequência, insere um novo **BIN** com valor zero e atualiza o valor de **NBA**.

Ao encerrar o bloco de laço, ele incrementa o último **BIN** armazenado da origem e encerra o mecanismo.

Para uma melhor compreensão é apresentado o 1. Que é o pseudocódigo do algoritmo de coleta descrito anteriormente na Figura 5.

A segunda parte do algoritmo é a função de avaliação da amostra. A função que realiza essa etapa recebe a lista de **BINS** de uma origem e retorna a frequência relativa dos dígitos mais significativos calculada a partir da fórmula proposta por Benford com a alteração de considerar a ocorrência do valor zero na fórmula. O motivo dessa alteração da fórmula é descrito

Figura 5 – Diagrama coleta



Fonte: elaborado pelo autor (2022).

na seção 4.6.

Após o cálculo do primeiro dígito significativo, são somadas as duas maiores frequências dos dígitos não nulos. O resultado da soma é comparado a um valor limite definido. Com base neste limite, o tráfego é então classificado como malicioso ou não. A definição desse limite é discutida na seção 4.6.

Algoritmo 1: Coleta

```
início
    connectionMoment ← momento da requisicão em timestamp;
    if Não existe BIN para a origem then
        Cria primeiro BIN com valor 1;
        Cria TIP da origem;
        Cria TAV da origem;
    else
        NBA ← tamanho da lista_de_BINS do ip_origem;
        if connectionMoment > TIP + TB * NBA then
            if NBA > CollSize then
                if soma da lista_de_BINS do ip_origem > CollSize then
                    CHAMA A FUNÇÃO DE AVALIAÇÃO;
                    TAV ← 0;
                else
                    TAV ← TAV + 1
                end
            end
            while connectionMoment > TIP + TB * NBA do
                if NBA > CollSize then
                    Remove o BIN mais antigo;
                    Atualiza o TIP da origem;
                end
                adiciona um novo BIN com valor 0;
                atualiza NBA;
            end
        end
        Incrementa o último BIN da origem;
    end
fim
```

Fonte: elaborado pelo autor (2023).

Como prova de conceito, esse mecanismo foi desenvolvido em forma de um *plugin* para o Kong API Gateway. Foi utilizada a linguagem GO para o desenvolvimento do algoritmo por afinidade com a linguagem e pelo bom paralelismo com o sistema. O código completo do *plugin* está disponível no apêndice A e suas recomendações de uso no apêndice B.

4.6 Calibração de Variáveis

Para calibrar as variáveis utilizadas na ferramenta foram realizados diversos experimentos buscando identificar uma forma de diferenciar o tráfego real de um tráfego de ataque. Como definição de intervalos de treino e de teste, os conjuntos de dados de tráfego real e de ataque foram divididos em 70% para treino e 30% para testes.

Devido ao tipo de dados, séries temporais, optou-se por dividi-los com base no tempo. Logo os dados de treino são a primeira parte dos dados e os dados de teste a segunda parte, o tamanho das partes se baseiam na já citada proporção 70:30. Essa abordagem permite manter a ordem cronológica de cada evento inserido no conjunto de dados, não alterando os dados que são vizinhos e mantendo o sentido da série temporal.

Após o algoritmo pronto, foi feita uma primeira rodada de treinos utilizando o cálculo do dígito mais significativo padrão, buscando uma forma de identificar divergências entre os tráfegos real e de ataque. Devido à baixa capacidade de classificação obtida utilizando esse cálculo, novas rodadas de treino utilizando variações do mesmo foram necessárias a fim de obter resultados mais significativos.

De início era necessário definir duas variáveis utilizadas no algoritmo: TB - Tempo de BIN em segundos, a qual é o intervalo de tempo de cada BIN onde é armazenada a quantidade de conexões e CollSize - Tamanho da coleta, ou seja, o tamanho da amostra a ser avaliada. Ambas as variáveis afetam o tempo de detecção.

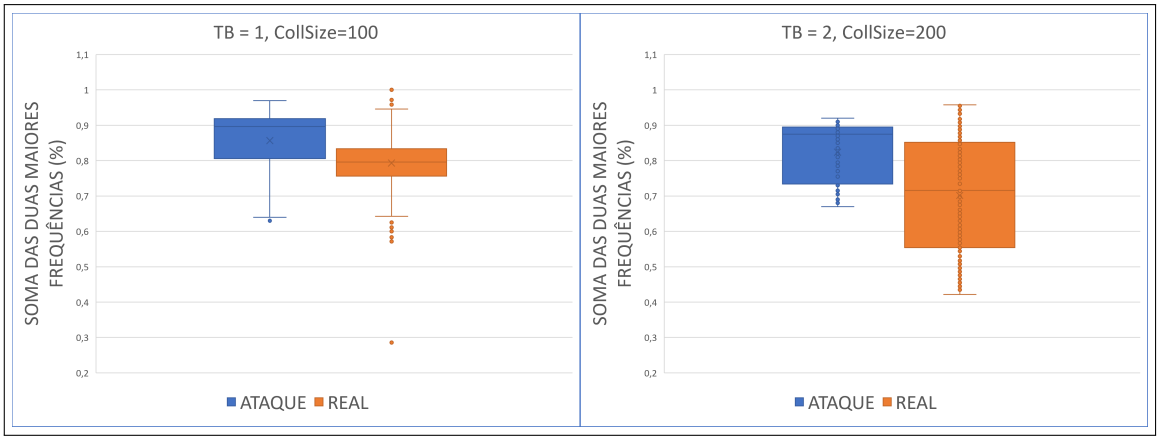
Para definir os valores que seriam utilizados nas variáveis foi adotada uma estratégia iterativa de calibração. Inicialmente foram escolhidos valores arbitrários para TB e CollSize. Baseado no desempenho obtido pelo algoritmo, os valores foram sendo refinados até chegar em um conjunto de elementos para cada uma das variáveis, sendo os níveis de TB sendo {1, 1,5, 2} e os de CollSize {100, 150, 200}. Os conjuntos, ao serem combinados entre si, geraram 12 cenários distintos.

Seguindo a lógica do princípio de Pareto, 20% das ocorrências deveriam ser responsáveis por 80% dos casos. Logo, como estamos usando o dígito mais significativo, dois dígitos, que são aproximadamente 20% das possibilidades quando usamos o sistema decimal sem o 0, deveriam ocorrer em 80% dos intervalos analisados. Com este fim foram então identificados os dois dígitos mais frequentes de cada intervalo e em seguida somados.

A Figura 6 apresenta os gráficos de valores mínimos e máximos da soma das duas maiores frequências. Foi observado um afunilamento desde a menor configuração que é TB=1

e CollSize=100 até a maior configuração que é TB=2 e CollSize=200. Em ambos os gráficos é possível observar que o intervalo entre os valores mínimos e máximos de tráfego de ataque e real se tocam, chegando em um momento onde os limites do tráfego de ataque ficam dentro dos limites do tráfego real, impossibilitando a diferenciação entre ambos nesta análise. Ainda foi cogitada a possibilidade que a soma da frequência dos 3 dígitos mais recorrentes pudessem produzir melhores resultados, entretanto não foi o caso e a ideia foi abandonada.

Figura 6 – Gráficos - Soma das duas maiores frequências.



Fonte: elaborado pelo autor (2023).

Prandl *et al.* (2017) definiu que a chegada de pacotes em um servidor obedece à Lei de Benford. Utilizando o conjunto de dados de treino foram realizados dois teste de aderência, Q-quadrado e Kolmogorov-Smirnov, para a Lei de Benford. Os resultados obtidos definiram que em janelas de tempo menores de 700 segundos, nem o tráfego de ataque, nem o tráfego real aderiam à Lei de Benford.

Já em janelas maiores chegando a 1000 segundos, menos de 50% do tráfego real adere à Lei de Benford, e o tráfego de ataque continua sem aderir. Janelas maiores que essa tornariam a ferramenta ineficaz, já que passariam de 16 minutos. Dessa forma se adequando melhor para análise posterior do tráfego do que para uma ferramenta de defesa em tempo real.

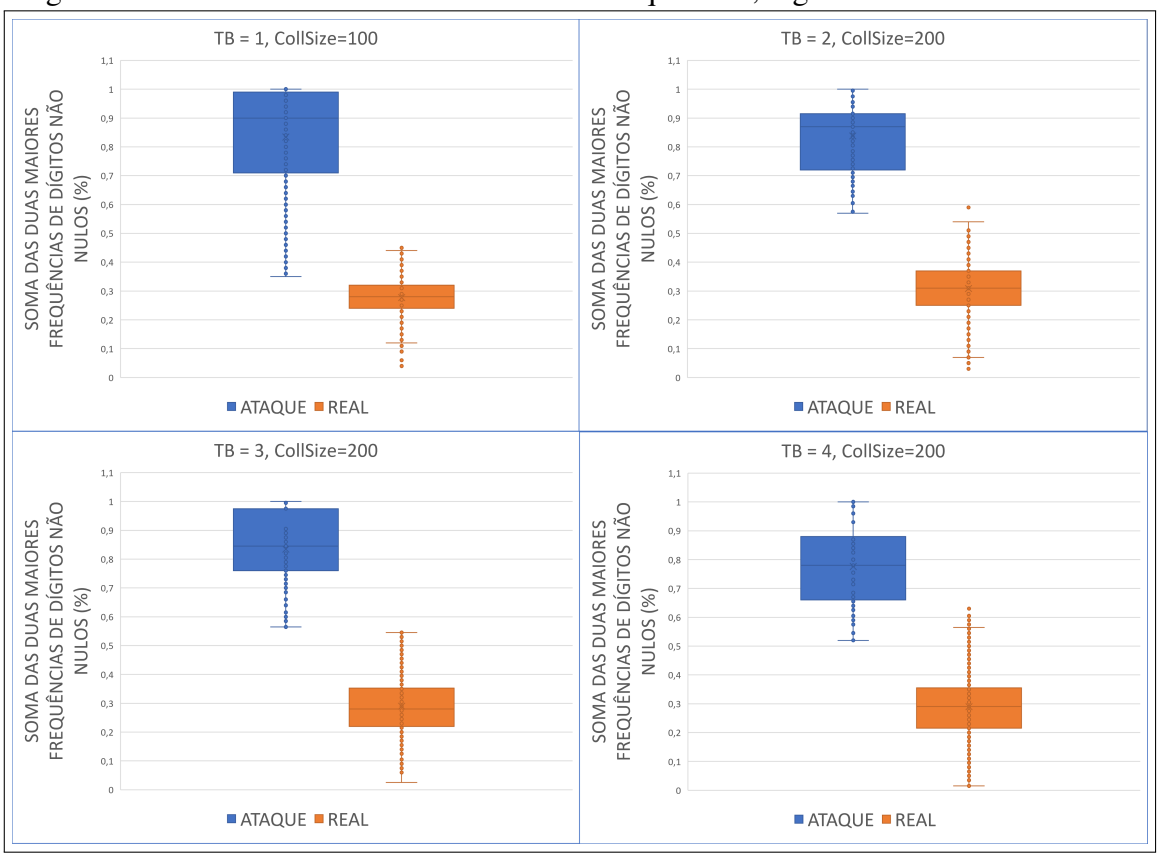
Uma preocupação que surgiu foi considerar o fator de não conexão dentro da amostra avaliada. Para isso foi incluindo o valor 0 no cálculo do dígito mais significativos. Levando em consideração que o vetor de BINS é composto apenas por números inteiros. Essa modificação do cálculo faz com que as frequências não sejam mais calculadas apenas entre [1, ... ,9] e sim entre [0, ... , 9].

A partir dessa modificação do cálculo foi realizado um novo treino considerando o fator 80–20, onde as duas maiores frequências dos contadores não nulos são somadas. Para essa

nova rodada de treino, além dos 9 cenários utilizados, mais 6 foram adicionados. Isso se deu graças a expansão do conjunto de TB, ao qual foram adicionados novos valores 3 e 4. A decisão de adicionar esses valores veio do comportamento que os gráficos da soma das frequências de tráfego real e de ataque apresentavam, já que pareciam estar cada vez mais distintos um do outro com a evolução do TB.

Como resultado dessa adição foi possível notar que de fato em um momento os gráficos se tornaram distintos a ponto do valor mínimo dos resultados do tráfego ataque ser 2% maior que o valor máximo obtido com os dados de tráfego real. Essa separação completa ocorreu em apenas um dos cenários que foi a combinação TB=3 e CollSize=200. Todos os outros cenários apresentavam intersecções dos gráficos em maior ou menor grau.

Figura 7 – Gráficos - Soma das duas maiores frequências, digitos não nulos.



Fonte: elaborado pelo autor (2023).

A Figura 7 apresenta 4 gráficos com os valores obtidos nos cenários: (TB=1, CollSize=100), (TB=2, CollSize=200), (TB=3, CollSize=200) e (TB=4, CollSize=200). Os dois primeiros cenários foram escolhidos para que haja a comparação entre os resultados deste treino e os do treino anterior, com o cálculo do dígito significativo padrão. Essa comparação mostrou que esta nova versão do cálculo tem resultados mais promissores. Já o terceiro cenário foi escolhido

por ser o único em que houve separação total dos gráficos e o quarto cenário foi escolhido por ser o TB seguinte ao do cenário mais promissor, mostrando que qualquer incremento nessa variável além desse ponto causa, na verdade, uma piora dos resultados.

Vale lembrar que algumas configurações do conjunto de dados do tráfego de ataque foram geradas usando as rajadas de conexões seguindo a distribuição normal, com a intenção de espaçar as conexões para se tornar similar ao tráfego real.

4.7 Análise de Resultados

A partir da etapa de calibração de variáveis foi possível definir os níveis das variáveis TB como 3 e CollSize como 200. E também definir o limite utilizado para diferenciar tráfego real de tráfego de ataque, sendo 55%. Como citado anteriormente, o conjunto de dados aplicado ao treino representa 70% do conjunto de dados total e os resultados são obtidos a partir da aplicação no conjunto de dados de teste que representa 30% do conjunto de dados total.

Para avaliar a classificação do perfil de tráfego obtido nos resultados, será utilizado a métrica de avaliação matriz de confusão. Formalmente, a matriz de confusão é uma matriz NxN não negativa de valores inteiros, onde N representa o número de classes (SUSMAGA, 2004). Neste estudo utilizamos duas classes, sendo tráfego normal ou tráfego de ataque.

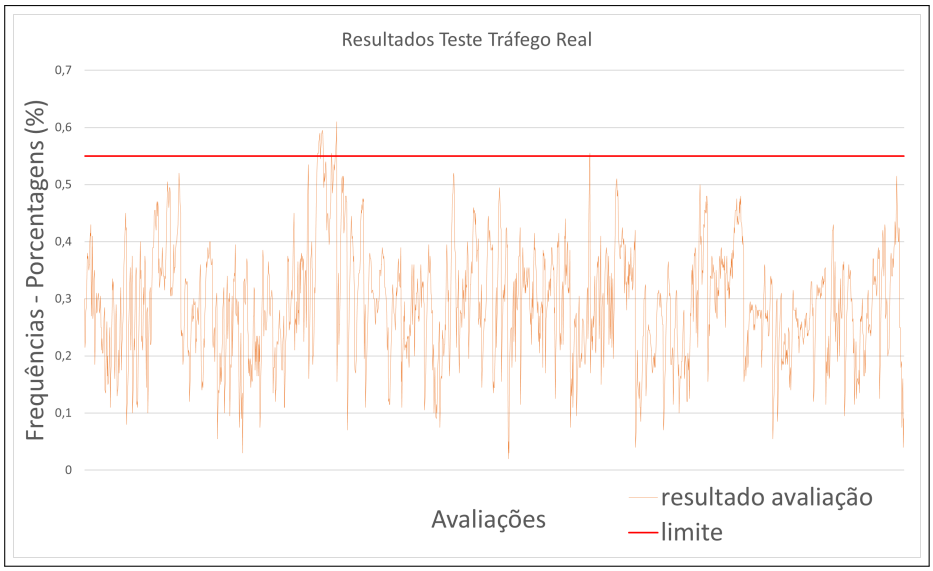
A partir da matriz de confusão serão derivadas as métricas: acurácia; precisão; e *recall*. Acurácia, indica o desempenho geral da modelo, dentre todas as classificações, quantas o modelo classificou corretamente. Precisão, indica dentre todas as classificações da classe positiva do modelo, quantas estão corretas. E por fim, *Recall*, indica dentre todas as situações da classe positiva do valor esperado, quantas estão corretas (DALIANIS, 2018).

5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos com a aplicação do mecanismo proposto no conjunto de dados de testes. Nos teste foram aplicadas as variáveis TB igual a 3 e CollSize igual a 200 como discutido no Capítulo 4.

A Figura 8 apresenta um gráfico gerado a partir das avaliações realizadas no tráfego real. Onde, o eixo *Y* representa a soma das frequências dos dois dígitos não nulos com maior frequência relativa. O eixo *X* são as avaliações realizadas na amostra. A linha laranja representa as avaliações realizadas no tráfego real. Já a linha vermelha representa o valor limite definido para considerar o tráfego como sendo normal (sem ataque).

Figura 8 – Gráfico de resultados trafego real.



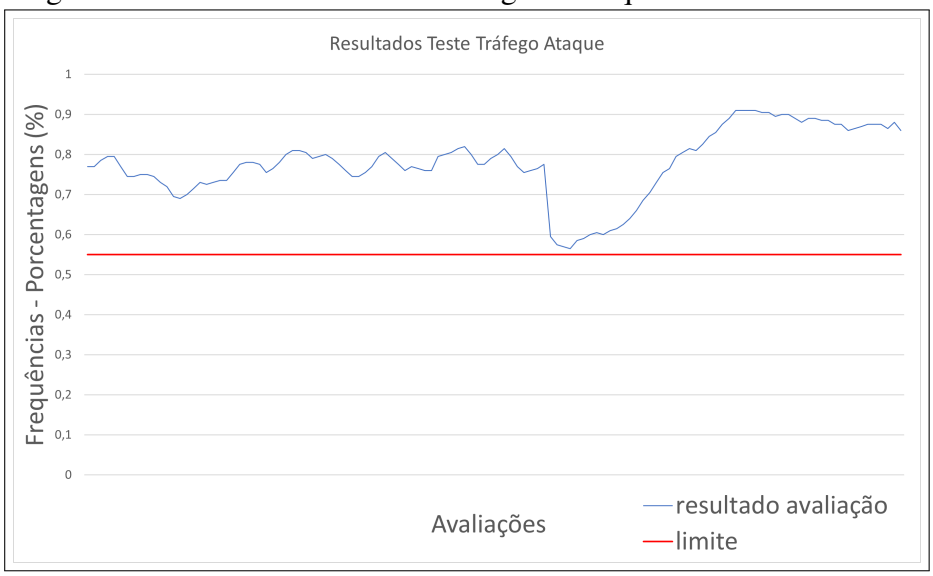
Fonte: elaborado pelo autor (2021).

A Figura 9 apresenta um gráfico gerado a partir das avaliações realizadas no tráfego de ataque. Onde, o eixo *Y* representa a soma das frequências dos dois dígitos não nulos com maior frequência relativa. O eixo *X* são as avaliações realizadas na amostra. A linha azul representa as avaliações realizadas no tráfego de ataque. Já a linha vermelha representa o valor limite definido para considerar o tráfego como sendo normal (sem ataque).

Observando ambos os gráficos, é possível afirmar que em um determinado momento o tráfego real é considerado tráfego de ataque, pois o valor ultrapassa o limite definido. Entretanto, todas as avaliações no tráfego de ataque foram eficazes.

A Figura 10 apresenta a matriz de confusão gerada a partir dos resultados obtidos. Podemos perceber que a maior parte das classificações se agrupam na diagonal principal da

Figura 9 – Gráfico de resultados trafego de ataque.



Fonte: elaborado pelo autor (2021).

matriz, o indica poucos erros de classificação.

A partir da matriz de confusão obtida, foram derivadas as seguintes métricas: acurácia igual a 0,991; precisão igual a 0,856; e *Recall* igual a 1. A partir destas métricas, se pode afirmar que o mecanismo apresenta eficácia em identificar ataques reais, a custo de classificar tráfego real como ataques. Isso poderia ser alterado movendo o limite, entretanto acarretaria o surgimento de falsos negativos.

Figura 10 – Matriz de confusão.

		PREDITO	
		ATAQUE	NORMAL
REAL	ATAQUE	125	0
	NORMAL	21	2186

Fonte: elaborado pelo autor (2023).

6 CONCLUSÕES E TRABALHOS FUTUROS

Com o limite definido em 55%, utilizando a configuração de variáveis TB igual a 3 e CollSize igual a 200, para a soma das duas maiores frequências dos dígitos significativos não nulos, o algoritmo apresentou uma taxa de 1% de falsos positivos e 0% de falsos negativos. Apresentando, uma acurácia de 99,1%, uma precisão de 85,6%, e um *recall* de 100%. Logo, esse limite gera um custo de falsos positivos, o que poderia ser alterado movendo o limite para cima. Entretanto, isso acarretaria o surgimento de falsos negativos. Com todo estudo elaborado nesse trabalho, é possível afirmar que uma avaliação estatística focada apenas na frequência de chegada de conexões a um serviço não é suficientemente eficaz para mitigar ataques DDoS na camada de aplicação, podendo o atacante contornar essa avaliação utilizando distribuições para enganar a ferramenta.

Uma dificuldade encontrada durante a elaboração desse estudo foi o conjunto de dados para teste, pois mesmo sendo uma quantidade de horas considerável, representa apenas um tipo de serviço. Para tráfego gerado por ferramentas, os valores de frequência relativa do primeiro dígito significativo são sempre iguais para cada distribuição definida na ferramenta. Vale ressaltar que não conhecemos distribuições que se adequem exatamente a cenários reais.

Uma limitação identificada do mecanismo proposto é o fato de o atacante pode emular distribuições na geração de requisições, o que provavelmente conseguirá evadir a detecção.

Para trabalhos futuros, é possível considerar alguns pontos como utilizar mais dados de acesso a diferentes serviços, buscando serviços que se comportem de forma diferente em relação à comunicação com o servidor, utilizar métricas adicionais com tamanho de pacotes como citado por outros trabalhos. Outro ponto é explorar diferentes ferramentas de ataque. E também tentar gerar uma ferramenta que tente evadir ao ataque, por exemplo, testando distribuições específicas que se adequem ao perfil.

REFERÊNCIAS

BAARZI, A. F.; KESIDIS, G.; FLECK, D.; STAVROU, A. Microservices made attack-resilient using unsupervised service fissioning. **Association for Computing Machinery**, New York, NY, USA, p. 31–36, 2020.

BAKSHI, A.; DUJODWALA, Y. B. Securing cloud from ddos attacks using intrusion detection system in virtual machine. In: **2010 Second International Conference on Communication Software and Networks**. [*S. l.*: *s. n.*], 2010. p. 260–264.

BEITOLLAHI, H.; DECONINCK, G. Connectionscore: A statistical technique to resist application-layer ddos attacks. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 5, n. 3, p. 425–442, 2014.

BENFORD, F. The law of anomalous numbers. **Proceedings of the American philosophical society**, JSTOR, p. 551–572, 1938.

CHEN, Z.; XU, G.; MAHALINGAM, V.; GE, L.; NGUYEN, J.; YU, W.; LU, C. A cloud computing based network monitoring and threat detection system for critical infrastructures. **Big Data Research**, v. 3, p. 10–23, 2016. ISSN 2214-5796. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214579615000520. Acesso em: 30 nov. 2023.

CISO ADVISOR. **Cibercrime sofisticou seus métodos de ataque DDoS**. 2022. Disponível em: https://www.cisoadvisor.com.br/cibercrime-sofisticou-seus-metodos-de-ataque-ddos/. Acesso em: 30 nov. 2023.

DALIANIS, H. Evaluation metrics and evaluation. In: ____. **Clinical Text Mining:**: Secondary use of electronic patient records. Cham: Springer International Publishing, 2018. p. 45–53. ISBN 978-3-319-78503-5. Disponível em: https://doi.org/10.1007/978-3-319-78503-5_6. Acesso em: 30 nov. 2023.

DAVID, J.; THOMAS, C. Ddos attack detection using fast entropy approach on flow- based network traffic. **Procedia Computer Science**, v. 50, p. 30–36, 2015. ISSN 1877-0509. Big Data, Cloud and Computing Challenges. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050915005086. Acesso em: 30 nov. 2023.

DOU, W.; CHEN, Q.; CHEN, J. A confidence-based filtering method for ddos attack defense in cloud environment. **Future Generation Computer Systems**, v. 29, n. 7, p. 1838–1850, 2013. ISSN 0167-739X.

ISMAIL, M. N.; ABORUJILAH, A.; MUSA, S.; SHAHZAD, A. Detecting flooding based dos attack in cloud computing environment using covariance matrix approach. Association for Computing Machinery, New York, NY, USA, 2013. Disponível em: https://doi.org/10.1145/2448556.2448592. Acesso em: 30 nov. 2023.

JIANG, M.; WANG, C.; LUO, X.; MIU, M.; CHEN, T. Characterizing the Impacts of Application Layer DDoS Attacks. p. 500–507, 2017.

KALI ORG. **Tool Documentation: Slowhttptest.** 2022. Disponível em: https://www.kali.org/tools/slowhttptest/. Acesso em: 30 nov. 2023.

KLYMASH, M.; SHPUR, O.; PELEH, N.; MAKSYSKO, O. Concept of intelligent detection of ddos attacks in sdn networks using machine learning. p. 609–612, 2020.

KONG. **API Gateway**. 2022. Disponível em: https://konghq.com/products/api-gateway-platform. Acesso em: 01 nov. 2023.

LI, Z.; JIN, H.; ZOU, D.; YUAN, B. Exploring new opportunities to defeat low-rate ddos attack in container-based cloud environment. **IEEE Transactions on Parallel and Distributed Systems**, IEEE, v. 31, n. 3, p. 695–706, 2020.

MANTAS, G.; STAKHANOVA, N.; GONZALEZ, H.; JAZI, H. H.; GHORBANI, A. A. Application-layer denial of service attacks: taxonomy and survey. **International Journal of Information and Computer Security**, v. 7, n. 2-4, p. 216–239, 2015. Disponível em: https://www.inderscienceonline.com/doi/abs/10.1504/IJICS.2015.073028. Acesso em: 30 nov. 2023.

MININET. **An Instant Virtual Network on your Laptop (or other PC).** 2022. Disponível em: http://mininet.org/. Acesso em: 30 nov. 2022.

MOUSAVI, S. M.; ST-HILAIRE, M. Early detection of ddos attacks against sdn controllers. In: **2015 International Conference on Computing, Networking and Communications (ICNC)**. [*S. l.*: *s. n.*], 2015. p. 77–81.

MURALEEDHARAN, N.; JANET, B. Behaviour analysis of http based slow denial of service attack. p. 1851–1856, 2017.

NGINX. **Nginx**. 2023. Disponível em: https://www.nginx.com/. Acesso em: 24 nov. 2023.

ORG., L. **Lua, The Programing Language**. 2023. Disponível em: https://www.lua.org/. Acesso em: 24 nov. 2023.

PRANDL, S.; LAZARESCU, M.; PHAM, D. S.; SOH, S. T.; KAK, S. An investigation of power law probability distributions for network anomaly detection. p. 217–222, 2017.

RAI, A.; CHALLA, R. K. Survey on Recent DDoS Mitigation Techniques and Comparative Analysis. In: **2016 Second International Conference on Computational Intelligence Communication Technology (CICT)**. [*S. l.*: *s. n.*], 2016. p. 96–101.

RED HAT. **Qual é a função de um gateway de API**. 2021. Disponível em: https://www.redhat.com/pt-br/topics/api/what-does-an-api-gateway-do. Acesso em: 24 ago. 2023.

SABRI, S.; ISMAIL, N.; HAZZIM, A. Slowloris dos attack based simulation. **IOP Conference Series: Materials Science and Engineering**, IOP Publishing, v. 1062, n. 1, p. 012029, feb 2021. Disponível em: https://dx.doi.org/10.1088/1757-899X/1062/1/012029. Acesso em: 30 nov. 2023.

SHIRSATH, V. **CAIDA UCSD DDOS 2007 ATTACK DATASET**. 2007. Disponível em: https://ieee-dataport.org/documents/caida-ucsd-ddos-2007-attack-dataset. Acesso em: 24 nov. 2023.

SIKORA, M.; FUJDIAK, R.; KUCHAR, K.; HOLASOVA, E.; MISUREC, J. Generator of slow denial-of-service cyber attacks. **Sensors**, v. 21, n. 16, 2021. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/21/16/5473. Acesso em: 30 nov. 2023.

SUSMAGA, R. Confusion matrix visualization. In: KŁOPOTEK, M. A.; WIERZCHOŃ, S. T.; TROJANOWSKI, K. (Ed.). **Intelligent Information Processing and Web Mining**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 107–116.

ZHANG, B.; ZHANG, T.; YU, Z. Ddos detection and prevention based on artificial intelligence techniques. In: **2017 3rd IEEE International Conference on Computer and Communications (ICCC)**. [*S. l.*: *s. n.*], 2017. p. 1276–1280.

APÊNDICE A – ALGORITMO DO PLUGIN

Código-fonte 1 – Código fonte do algoritmo desenvolvido em Go para o Kong API Gateway

```
1  package main
2
3  import (
4    "fmt"
5    "log"
6    "os"
7    "sort"
8    "strconv"
9    "time"
10   "github.com/Kong/go-pdk"
11 )
12
13 var Version = "0.5"
14 var Priority = 1
15
16 //tamanho do bin
17 var tb int = 3
18
19 //map que guarda o vetor de BINS [ip_origem][BINS]
20 var bins = make(map[string][]int)
21
22 //map que guarda so TIP, do primeiro BIN
23 var tips = make(map[string]int64)
24
25 //fator de nova avalia  o
26 var fna int = 10
27
28 //teste de para realizacao de nova avaliacao
29 var tav = make(map[string]int)
```

```
30
31 type Config struct {
32   CollSize int
33 }
34
35 func New() interface{} {
36   return &Config{}
37 }
38
39 func getPercentsSigniDigits(list_intervals []int)[]float64
     {
40   digits := make(map[string]float64)
41   qtd := 0
42   var percents []float64
43   for i := 0; i < 10; i++ {
44     digits[strconv.Itoa(i)] = 0
45   }
46
47   for _, t := range list_intervals {
48     digits[strconv.Itoa(t)[0:1]] = digits[strconv.Itoa(t)
         [0:1]] + 1
49     qtd = qtd + 1
50   }
51   for _, t := range digits {
52     percents = append(percents, float64(t)/float64(qtd))
53   }
54   sort.Float64s(percents)
55   return percents
56 }
57
58 func av(ip string, list_coleta []int) {
59   var path = "/home/kong/" + ip + ".txt"
```

```
60  var texto string
61  //testa se o arquivo existe, se nao, cria
62  var _, errt = os.Stat(path)
63  if os.IsNotExist(errt) {
64    log.Printf("Novo arquivo criado para: %s", ip)
65    var criador, errc = os.Create(path)
66    if errc != nil {
67      return
68    }
69    defer criador.Close()
70  }
71
72  f, err := os.OpenFile(path, os.O_APPEND|os.O_WRONLY,
       0644)
73  if err != nil {
74    log.Printf("%", err)
75    return
76  }
77  list_percents := getPercentsSigniDigits(list_coleta)
78
79  var check float64 = list_percents[8] + list_percents[9]
80
81  text_check := fmt.Sprintf("%f ", check)
82
83  if check < 0.55 {
84    texto = "Avaliado como nao positivo: " + text_check
85  }else{
86    texto = "Avaliado como positivo: " + text_check
87    }
88  }
89
90  _, err = fmt.Fprintln(f, texto)
```

```
91    if err != nil {
92      fmt.Println(err)
93      f.Close()
94      return
95    }
96    err = f.Close()
97    if err != nil {
98      fmt.Println(err)
99      return
100   }
101
102 }
103
104 func sum(vetor []int) int{
105   x := 0
106   for _, num := range vetor{x += num}
107   return x
108 }
109
110 func (conf Config) Access(kong *pdk.PDK) {
111   ip_origem, errip_origem := kong.Client.GetIp()
112   conectionMoment := time.Now().Unix()
113   teste_for := 0
114   if errip_origem != nil {
115     log.Printf("Erro ao ler o IP origem: %s", errip_origem.
          Error())
116   }
117   //checa se e a primeira conexao dessa origem, ou seja ela
        nao possui dados nos vetores
118   if _, checkTIP := bins[ip_origem]; !checkTIP{
119     //seta conectionMoment do primeiro BIN
120     tips[ip_origem] = conectionMoment
```

```
121     //cria o primeiro BIN
122     bins[ip_origem] = append(bins[ip_origem], 1)
123     //cria o TAV da origem
124     tav[ip_origem] = fna
125   }else{
126         // se nao e a primeira conexao desse cara ja existe
              dados dessa origem
127     nba := len(bins[ip_origem])
128     if conectionMoment > tips[ip_origem] + int64(tb * nba){
129       //checa o vetor de bins esta cheio
130       if nba >= conf.CollSize{
131         //checa se deve realizar avalia  o
132         if sum(bins[ip_origem]) >= conf.CollSize{
133           if tav[ip_origem] == fna{
134             av(ip_origem, bins[ip_origem])
135             tav[ip_origem] = 0
136           }else{
137             tav[ip_origem] = tav[ip_origem] + 1
138           }
139         }
140       }
141       for conectionMoment > tips[ip_origem] + int64(tb *
             nba){
142         if nba >= conf.CollSize{
143           //remove o bin mais antigo
144           bins[ip_origem]=bins[ip_origem][1:]
145           //atualiza o primeiro bin incrementa um tb de bin
146           tips[ip_origem] = tips[ip_origem]+int64(tb)
147         }
148         //insere um novo BIN com valor 0
149         bins[ip_origem] = append(bins[ip_origem],0)
150         //atualiza o nba
```

```
151           nba = len(bins[ip_origem])
152         }
153         // incrementa o ultimo BIN
154         bins[ip_origem][nba-1] = bins[ip_origem][nba-1]+1
155
156
157       }else{
158         //se o conectionMoment nao    maior, ele incrementa o
              ultimo BIN
159         bins[ip_origem][nba-1] = bins[ip_origem][nba-1]+1
160       }
161     }
162 }
```

APÊNDICE B – RECOMENDAÇÕES DE USO

Como prova de conceito, o *plugin* está estruturado para funcionar dentro do Kong API Gateway, mas o mesmo pode ser adaptado para funcionar em qualquer outro, basta seguir o padrão de uso para o API Gateway escolhido, sendo necessárias algumas alterações no código, já que o mesmo utiliza **GO-PDK**, o qual é uma biblioteca de comunicação do *plugin* com o Kong.

Para a utilização do *plugin* no Kong API Gateway basta que o mesmo seja compilado com *'go build -buildmode plugin <nome>.go'*. Juntamente com ele, também deve ser compilado o go-pluginserver disponível no site (KONG, 2022). Ambos devem ser copiados para seus respectivos diretórios de acesso a *plugins* configurados no Kong API Gateway. Após isso, basta realizar uma configuração de *plugin* comum no próprio Kong. Lembrando que os únicos parâmetros configuráveis pelo administrador são: tamanho da amostra (CollSize) e intervalo (TB) de amostra para avaliação.