UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Mecanismo de prevenção de ataque DDoS em redes SDN

Dissertação de Mestrado

Alfredo Menezes Vieira

São Cristóvão – Sergipe

2021

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Alfredo Menezes Vieira

Mecanismo de prevenção de ataque DDoS em redes SDN

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Admilson de Ribamar Lima Ribeiro
Coorientador(a): Rubens de Souza Matos Junior

São Cristóvão – Sergipe

2021

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Vieira, Alfredo Menezes
B658m Mecanismo de prevenção de ataque DDoS em redes SDN / Alfredo Menezes Vieira ; orientador Admilson de Ribamar Lima Ribeiro. - São Cristóvão, 2021.
81 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2021.

1. Computação. 2. Rede definida por software (Tecnologia de rede de computador). 3. Proteção de dados. I. Ribeiro, Admilson de Ribamar Lima orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: ALFREDO MENEZES VIEIRA

Em 23 dias do mês de setembro do ano de dois mil e vinte um, com início às 9h, realizou-se na Sala virtual https://meet.google.com/dxz-oupo-kqe?authuser=8&hl=pt_BR. A Sessão Pública de Defesa de Dissertação de Mestrado do candidato **ALFREDO MENEZES VIEIRA**, que desenvolveu o trabalho intitulado: **"Mecanismo de prevenção de ataque DDoS em redes SDN",** sob a orientação do Prof. Dr. **Admilson de Ribamar Lima Ribeiro**. A Sessão foi presidida pelo Prof. Dr. **Admilson de Ribamar Lima Ribeiro** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Wanderson Roger Azevedo Dias** (IFS) e em seguida, o Prof. Dr. **Jean Carlos Teixeira de Araujo** (PROCC/UFS) e o coorientador o Prof. Dr. **Rubens de Souza Matos Júnior** (PROCC/UFS) . Após as discussões, a Banca Examinadora reuniu-se e considerou o(a) mestrando(a) Aprovado "*(aprovado/reprovado)"*. Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), Resolução nº 25/2014/CONEPE e da Portaria nº 413 de 27 de maio de 2020 (Banca por videoconferência) que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 23 de setembro de 2021.

Prof. Dr. Admilson de Ribamar Lima Ribeiro
(PROCC/UFS)
Presidente

Prof. Dr. Rubens de Souza Matos Júnior
(PROCC/UFS)
coorientador

Prof. Dr.Wanderson Roger Azevedo Dias
(IFS)
Examinador Externo

Prof. Dr. Jean Carlos Teixeira de Araujo
(PROCC/UFS)
Examinador Interno

Alfredo Menezes Vieira
Candidato

Agradecimentos

Primeiramente agradeço a Deus, que me deu saúde e forças para superar todos os obstáculos nessa caminhada, de ter materializado "Anjos" no meu caminho, viabilizando troca de experiências e conhecimentos, essenciais na minha formação e fundamentais na elaboração dessa dissertação.

A minha esposa Clésia, pelo seu amor e dedicação em compartilhar meus sonhos, dividir minhas inseguranças, incertezas e por todo seu apoio, paciência e compreensão.

Ao meu filho Felipão, pelo seu amor incondicional, compreendendo minhas ausências e inseguranças, ter sido sempre um ouvinte atento e crítico em todas as etapas do projeto, encorajando-me através dos seus beijos e abraços carinhosos como forma de incentivo e motivação.

Aos “Meus pais, Aparecido e Celina", pelo exemplo de vida, fé, amor, respeito e ética. Valores estes, que moldaram minha identidade pessoal e profissional. Ao meu irmão e amigo Lorenço, por caminhar sempre ao meu lado e não permitir que eu desanimasse diante das incertezas e dos obstáculos.

Um agradecimento especial ao meu orientador, Professor Admilson, a quem divido a parceria desse trabalho, por ter compartilhado o tema de pesquisa, suas experiências, seus conhecimentos e suas orientações.

Para muitos, os Anjos possuem asas, porém para mim eles utilizam um computador com o sistema operacional Linux Ubuntu. Agradeço, ao meu amigo Anjo, ex-aluno Alex, colega de trabalho pela parceria e apoio nos momentos difíceis. Ao amigo Anjo Professor Gilson, pelo incentivo e toda colaboração na socialização de todas ferramentas que fizeram a diferença neste trabalho.

Agradeço ao amigo Anjo Professor Rubens Coordenador e Coorientador, por ter me apresentado a pesquisa científica, pelos conselhos e orientações e em dividir os acertos e os erros durante o processo. Enfim, a todos os meus alunos, colegas de profissão e funcionários do DCOMP/UFS e a secretária Elaine por terem contribuído nesta conquista pessoal e profissional. Minha eterna gratidão.

Sonho que se sonha só,
é só um sonho que se sonha só,
mas um sonho que se Sonha junto é Realidade.
(Raul Seixas)

Resumo

A Rede Definida por *Software* (SDN) oferece benefícios como escalabilidade, flexibilidade, monitoramento e facilidade de inovação, pela sua característica principal de separar o plano de dados do plano de controle. A comunicação entre o controlador e o plano de dados é realizada por meio do protocolo *OpenFlow*, permitindo o envio e o recebimento de mensagens de um *switch* com suporte deste protocolo. Desse modo, permite que o controlador SDN envie instruções por meio de códigos desenvolvidos em diversas linguagens de programação para os dispositivos de rede. Devido a sua estrutura logicamente centralizada e controlada por *software*, o controlador se torna um alvo estratégico na realização de ataques. Dentre as diversas ameaças existentes, o ataque distribuído de negação de serviço (DDoS) possui um efeito destrutivo em redes SDN. O principal objetivo deste ataque cibernético é que os usuários legítimos sejam prejudicados devido à negação de serviço. A realização do ataque possui fases distintas e conta com dispositivos infectados os quais são chamados de *bot*, formando-se um exército conhecido como *botnet*. A prevenção contra o ataque DDoS envolve métodos que tem como objetivo evitar que a infraestrutura de rede seja uma vítima desta forma de ataque. Diante dos resultados observados por meio de um mapeamento sistemático, resolvemos neste trabalho propor e analisar um mecanismo de prevenção de ataques DDoS em redes SDN que atue na primeira fase do ataque, na proteção do controlador SDN. Dos dois tipos de varreduras existentes (horizontal e vertical), foram observados a partir dos experimentos que o mecanismo obtém de 98,64% a 99,37% de acurácia, 63,89% a 82,76% de precisão e 77,97% a 84,62% *F1-Score* para varredura vertical e 99,73% a 100% de acurácia, 99,46% a 100% de precisão e 99,73% a 100% *F1-Score* para varredura horizontal. Pode ser útil para administradores de redes SDN no contexto de defesa desse tipo de infraestrutura.

Palavras-chave: Ataque Distribuído de Negação de Serviço. DDoS. Rede Definida por Software. SDN. Prevenção.

Abstract

The Software Defined Network (SDN) offers benefits such as scalability, flexibility, monitoring and ease of innovation, due to its main characteristic of separating the data plane from the control plane. Communication between the controller and the data plane is carried out through the OpenFlow protocol, allowing the sending and receiving of messages from a switch that supports this protocol. In this way, it allows the SDN controller to send instructions through codes developed in various programming languages to the network devices. Due to its logically centralized and software-controlled structure, the controller becomes a strategic target in carrying out attacks. Among the many existing threats, the distributed denial of service (DDoS) attack has a destructive effect on SDN networks. The main objective of this cyber attack is for legitimate users to be harmed due to denial of service. The execution of the attack has distinct phases and counts on infected devices which are called bots, forming an army known as botnet. DDoS attack prevention involves methods that aim to prevent the network infrastructure from falling victim to this form of attack. Given the results observed through a systematic mapping, we decided in this work to propose and analyze a mechanism for preventing DDoS attacks in SDN networks that acts in the first phase of the attack, protecting the SDN controller. Of the two types of existing scans (horizontal and vertical), it was observed from the experiments that the engine obtains from 98.64% to 99.37% accuracy, 63.89% to 82.76% accuracy and 77.97% to 84.62% F1-Score for vertical scanning and 99.73% to 100% accuracy, 99.46% to 100% precision and 99.73% to 100% F1-Score for horizontal scanning. It can be useful for SDN network administrators in the context of defending this type of infrastructure.

Keywords: Distributed Denial of Service Attack. DDoS. Software Defined Network. SDN. Prevention.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

ACL	*Access Control List*
ARC	*Attack-aware recovery Controller Link-switch*
ARP	*Address Resolution Protocol*
BDSS	*Blocking DDoS Scan on Sdn*
CSV	*Comma Separated Values*
DCOMP	Departamento de Computação
DDoS	Ataque Distribuído de Negação de Serviço
DNS	*Domain Name System*
DoS	Ataque de Negação de Serviço
FPGA	*Field Programmable Gate Array*
GPU	*Graphics Processing Unit*
GUI	*Graphical User Interface*
HTTP	*Hypertext Transfer Protocol*
IDPS	*Intrusion Detection and Prevention System*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
IRC	*Internet Relay Chat*
KNN	*K - Nearest Neighbor*s
MAC	*Media Access Control*
MLP	*Multi-layer Perceptron*
NFV	*Network Functions Virtualization*
RAM	*Random Access Memory*
SSD	*Solid State Drives*
OvS	*OpenvSwitch*

PSO	*Particle Swarm Optimization*
P2P	*Peer-to-peer*
SDN	*Software Defined Networking*
SDWSN	*Software-Defined Wireless Sensor Network*
SMA	Sistemas Multiagentes
SVM	*Support Vector Machine*
TLL	*Time to Live*
UFS	Universidade Federal de Sergipe

Sumário

1 Introdução

A Rede Definida por *Software* (SDN - *Software Defined Networking*) é um paradigma de rede de computadores que separa o plano de controle do plano de dados, sendo que as políticas são definidas em um controlador SDN. Enquanto no plano de dados, os elementos de roteamento e comutação operam como encaminhadores de pacotes, atendendo às políticas informadas pelo controlador. Esta separação desempenha um papel crítico proporcionando um desempenho superior em larga escala e alta velocidade nos sistemas de computação (RAHMAN; QURAISHI; LUNG, 2019).

Essa reorganização arquitetural coloca o controlador com uma visão centralizada e geral da rede e permite a gerência dos sistemas subjacentes (*switches* e roteadores) como o plano de dados deve tratar o tráfego da rede. A comunicação entre o plano de controle e o plano de dados pode ser realizada pelo protocolo *OpenFlow*, o que permite que os comutadores executem o controle no nível de fluxo (YAN et al., 2015).

No entanto, pela sua característica da centralização do controle e por ser programável, a rede SDN também é exposta a uma variedade de novas ameaças à segurança. Segundo Sahoo et al. (2018), o ataque distribuído de negação de serviço (DDoS - *Distributed Denial of Service*) é um ataque de rápido crescimento que representa uma grande ameaça e pode ser catastrófico em uma rede SDN, sobrecarregando os recursos gerenciados com o *OpenFlow* se a rede não estiver adequadamente protegida.

Diversos autores apresentam métodos e técnicas para a prevenção de ataques DDoS em redes SDN, como Sahoo et al. (2018), Huang e Yu (2019) que utilizaram *Machine Learning* em seus trabalhos. Rahman, Quraishi e Lung (2019), Pande et al. (2018) apresentam em seus trabalhos métodos de defesa que atuam na segunda fase do ataque, que consiste na formação da *botnet*. No entanto, com base na prevenção de ataques DDoS, a maioria dos trabalhos tem como foco em atuar na terceira fase onde o ataque já está em execução, apresentando impactos nos serviços da rede. Ainda assim, na literatura há uma escassez de trabalhos que abordem a

prevenção na primeira fase do ataque, na qual o objetivo é reunir o uma quantidade significativa de *hosts* zumbis (Choi et al., 2010).

Pelo exposto acima o trabalho realizado nesta dissertação propõe um mecanismo de prevenção que atue diretamente na primeira fase do ataque DDoS, no momento em que a rede SDN está sendo analisada com o objetivo de encontrar *hosts* com vulnerabilidades que possam tornar-se zumbis, também conhecidos como *bots*, ou seja, agentes integrantes de um futuro ataque DDoS. O mecanismo proposto baseia-se na rápida detecção de ações de varredura dos *hosts* em uma rede SDN e serve também como base para a utilização do método *Honeypot*.

1.1 Problemática e Hipótese

As mudanças na área de tecnologia da informação proporcionaram o surgimento de diversas novas aplicações e evoluções arquiteturais para a construção e implantação de aplicações distribuídas cada vez mais flexíveis. Em contrapartida, as redes de computadores permaneceram por muito tempo com suas características inalteradas, e pouca adaptabilidade a cenários dinâmicos. Neste contexto, surgiu o paradigma de SDN (*Software Defined Network*), que permite a rede fisicamente distribuída ter a sua estrutura logicamente centralizada e controlada por *software*.

No entanto, pelo mesmo motivo que em que a SDN demonstra-se ser um futuro próximo nas redes de computadores, também fica suscetível às novas ameaças. Sendo que, o controlador SDN passa a ser um alvo em potencial na rede, visto que ele é o responsável pelo fluxo de dados que trafegam na rede. Esta é uma vulnerabilidade que pode ser utilizada por um ataque DDoS, causando a interrupção dos serviços da rede e como também a sua total falha.

Os últimos relatórios de ameaças gerados pelos laboratórios de segurança demonstram com clareza que os ataques DDoS são uma ameaça em potencial para a segurança da rede. Além disso em 2018, o crescimento do ataque DDoS foi classificado como terceiro maior e a sua frequência vem aumentando a cada ano (SAHARAN; GUPTA, 2019).

Diante dessa situação, a solução aqui proposta a ser investigada é a viabilidade da detecção de um *host* infectado na fase inicial do ataque, para a análise e criação de políticas de acesso com o objetivo de isolar o *host* infectado.

1.2 Justificativa

Apesar da motivação exposta na Seção 1.1, o ataque DDoS tem se demonstrado uma grande ameaça nas redes definidas por software. Uma vez que, a SDN não possua um nível de segurança confiável, estará sujeita a este e outras formas de ameaças que visam comprometer o controlador SDN.

Da mesma maneira em que a SDN está na vanguarda, torna-se alvo de novas ameaças à

segurança. Considera-se que um dos ataques que tem um efeito mais devastador em uma rede SDN é o ataque distribuído de negação de serviço (RAHMAN; QURAISHI; LUNG, 2019).

Em uma investigação científica previamente realizada para esta dissertação, não foram encontrados muitos trabalhos que abordem especificamente a prevenção de ataques DDoS em redes SDN. Portanto, não há evidências na preocupação com a forma de prevenir, no entanto, ficou nítido a atenção dos pesquisadores com a detecção e mitigação do ataque. Não apenas isso, a maior parte dos trabalhos encontrados somente realizaram simulações utilizando *Mininet* [1]. Portanto, poucas são as análises baseadas em ambientes experimentais, mesmo que em escala reduzida .

1.3 Objetivos

O principal objetivo do presente trabalho é desenvolver um mecanismo de prevenção de ataques DDoS com alvo aos controladores de uma rede SDN. Bem como, para identificar *hosts* na rede possivelmente infectados com a realização de um experimento em laboratório (prototipação).

Para que o objetivo principal seja alcançado outros objetivos específicos devem ser concluídos, são eles:

- Identificar as principais técnicas de prevenção de ataques DDoS em redes SDN;
- Desenvolver métodos de detecção de varredura de rede;
- Verificar, por meio de um experimento, o desempenho do mecanismo de prevenção.

1.4 Metodologia

Este seção tem objetivo de detalhar os critérios utilizados para a elaboração deste trabalho. A Subseção 1.4.1, apresenta a classificação da natureza da pesquisa realizada. Por fim, a Subseção 1.4.2, descreve as etapas da dissertação.

1.4.1 Classificação da Pesquisa

Esta dissertação está classificada como uma pesquisa aplicada, visando encontrar soluções aos problemas apresentados na realidade, realizando um estudo bibliográfico e experimental.

[1] É um emulador de rede que cria uma rede de *hosts* virtuais, *switches*, controladores e links. Os *hosts Mininet* executam por padrão o *Linux* e seus *switches* oferecem suporte a *OpenFlow* para roteamento personalizado altamente flexível e rede definida por *software*.

1.4.2 Etapas de Pesquisa

Esta dissertação está dividida em três fases. A primeira fase compreende uma revisão bibliográfica para obter uma ideia precisa do estado da arte proporcionando um embasamento teórico necessário para a realização do trabalho, como também identificar lacunas de pesquisas.

A fase seguinte consiste em uma revisão sistemática da literatura para identificar métodos e técnicas de prevenção de ataques distribuídos de negação de serviço em redes definidas por software. Será investigado como são realizados os ataques DDoS, os métodos, técnicas e algoritmos utilizados para a prevenção de ataques DDoS em redes SDN e as ferramentas utilizadas para a simulação das redes SDN.

Na terceira fase será proposto um mecanismo de prevenção para a criação de políticas de segurança para o bloqueio e isolamento do *host* infectado dentro da rede SDN e a proteção do controlador SDN. Serão utilizadas métricas para validar a eficácia do mecanismo através de experimento.

1.5 Organização do Trabalho

A estrutura do documento está organizado em seis capítulos. Os tópicos a seguir descrevem o conteúdo de cada um deles:

- Capítulo 2 - Fundamentação Teórica: Apresenta o referencial teórico desta dissertação e uma visão geral sobre ataques DDoS em redes SDN;
- Capítulo 3 - Trabalhos Relacionados: Expõe os trabalhos selecionados por meio do mapeamento sistemático relacionados à prevenção de ataques distribuídos de negação de serviço em redes definidas por *software*;
- Capítulo 4 - BDSS - Blocking DDoS Scan on SDN: Apresenta o mecanismo para a prevenção de ataques DDoS em redes SDN;
- Capítulo 5 - Estudo de caso: Este capítulo apresenta a implementação do mecanismo para a prevenção de ataques DDoS em redes SDN;
- Capítulo 6 - Conclusão: Apresenta as considerações finais deste estudo, suas contribuições e trabalhos futuros.

2

Fundamentação Teórica

2.1 Redes Definidas por *Software* (SDN)

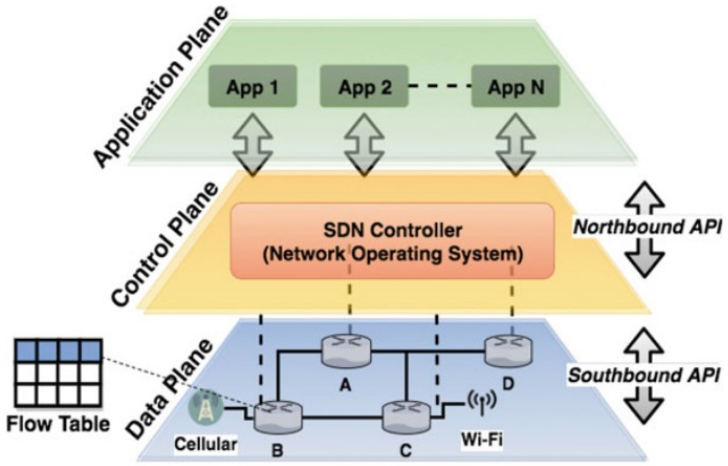
As redes definidas por *software* (SDN) são um paradigma de redes de computadores que define o encaminhamento de pacotes de forma diferente para superar as limitações das redes tradicionais. Neste paradigma consiste na separação do plano de controle do plano de dados tornando os elementos comutadores (*switches* ou roteadores) como simples encaminhadores de pacotes. Assim, com a inserção de um novo elemento de rede - o controlador - fornece uma política distribuída de forma consistente em toda a rede com a utilização de um protocolo padrão o *Openflow* (SAHOO et al., 2018). A proposta de separar o plano de controle do plano de dados é abordado nos trabalhos de Singh, Khan e Agrawal (2015), Yan et al. (2015), Trung et al. (2015), Yan et al. (2015), Singh, Khan e Agrawal (2015), Hussein et al. (2016), Alparslan et al. (2017), Zhu e Chang (2019), Rahman, Quraishi e Lung (2019), fornecem aos administradores de rede a facilidade de monitorar, controlar, gerenciar e configurar recursos de rede por meio do *software* em execução no controlador que possui um controle centralizado, uma visão global da rede e atualização dinâmica das regras de encaminhamento.

2.1.1 Arquitetura

A SDN possui três planos, conforme é ilustrado na Figura 1. O plano de aplicação que fica localizado no topo da arquitetura, é composto por diversos aplicativos de redes que por sua vez tem o objetivo de gerenciar a SDN. Por possuir a capacidade de programação viabiliza o desenvolvimento de novos aplicativos, tais como: roteamento, *firewall*, monitoramento, balanceamento de carga e entre outros (RAHMAN; QURAISHI; LUNG, 2019). No plano de controle que pode conter um conjunto de controladores baseados em *software* que encaminha regras e políticas utilizando as interfaces de comunicação *northbound* e *southbound* para interação com os demais planos (YAN et al., 2015). O plano de dados, também conhecido como camada

de infraestrutura é composto de equipamentos de rede, como *switches* e roteadores. No entanto, estes equipamentos atuam como encaminhadores simples de pacotes e se comunicam através do protocolo *OpenFlow* (SHAGHAGHI et al., 2020).

Figura 1 – Arquitetura SDN



Fonte: Shaghaghi et al. (2020)

A interface *Southbound* é um dos componentes mais críticos em uma arquitetura SDN, pois realiza a comunicação entre os dispositivos de encaminhamento com o plano de controle (SHAGHAGHI et al., 2020). Essa comunicação utiliza-se da interface *Northbound*, que possibilita a desenvolver diversos aplicativos de rede (BADOTRA; PANDA, 2020).

De acordo com Ahuja e Singal (2019), o plano de controle é responsável por tomar todas as decisões da rede, podendo ser visto como o cérebro no corpo humano. O controlador SDN possui a responsabilidade de manter, distribuir e atualizar as políticas e instruções para os dispositivos de rede. Sendo que, a comunicação entre o controlador e os equipamentos do plano de dados (*switches* e roteadores) é realizado pelo protocolo *Openflow*, baseado em fluxo (LEE; CHANG; SYU, 2020).

Segundo Shaghaghi et al. (2020), o plano de dados é composto de equipamentos de rede, como *switches* e roteadores. Eles ressaltam, que tais equipamentos atuam apenas como simples encaminhadores de pacotes em uma SDN.

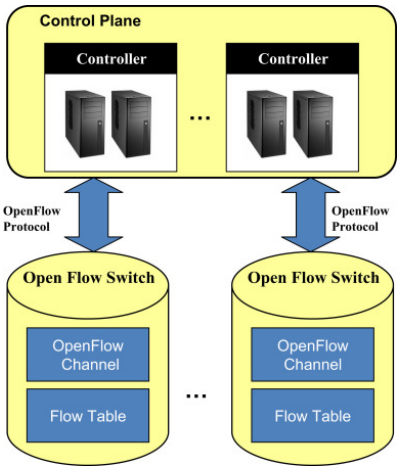
2.1.2 Protocolo *OpenFlow*

Segundo Trung et al. (2015), o *OpenFlow* chegou como um protocolo promissor utilizado para a comunicação entre o controlador e o plano de dados permitindo que o controlador envie e receba mensagens de um *switch OpenFlow*. Nas redes tradicionais os *switchs* possuem seus específicos *softwares* para a execução de tarefas exclusivas. Em uma rede definida por *software* é utilizado o protocolo *OpenFlow* que atua na interface *southbound* permitindo que o controlador

envie instruções para os equipamentos de rede. Desta forma, o controlador possui códigos em *software*, podendo ser desenvolvido em diversas linguagens de programação. O protocolo *OpenFlow* utiliza uma tabela de fluxo para lidar com o tráfego de rede rapidamente, portanto os comandos *Openflow* são capazes de configurar os *switches* quando possuírem essa tecnologia.

Conforme ilustrado na Figura 2, o *OpenFlow* possui três componentes principais: o *switch OpenFlow*, o canal *OpenFlow* e o controlador *OpenFlow*.

Figura 2 – Componentes e comunicação básica do protocolo *OpenFlow*



Fonte: Li, Meng e Kwok (2016)

Esses *switches* são gerenciados por controladores *OpenFlow* em um canal seguro utilizando o protocolo *OpenFlow*. Um *switch* geralmente consiste em uma ou mais tabelas de fluxo que realizam pesquisa e encaminhamento de pacotes (LI; MENG; KWOK, 2016).

O canal *OpenFlow* atua como uma interface para a conexão entre os *switches* e controladores *OpenFlow*. Por meio dessa interface, o controlador configura e gerencia o *switch*, recebe eventos e envia pacotes para fora do *switch* (LI; MENG; KWOK, 2016).

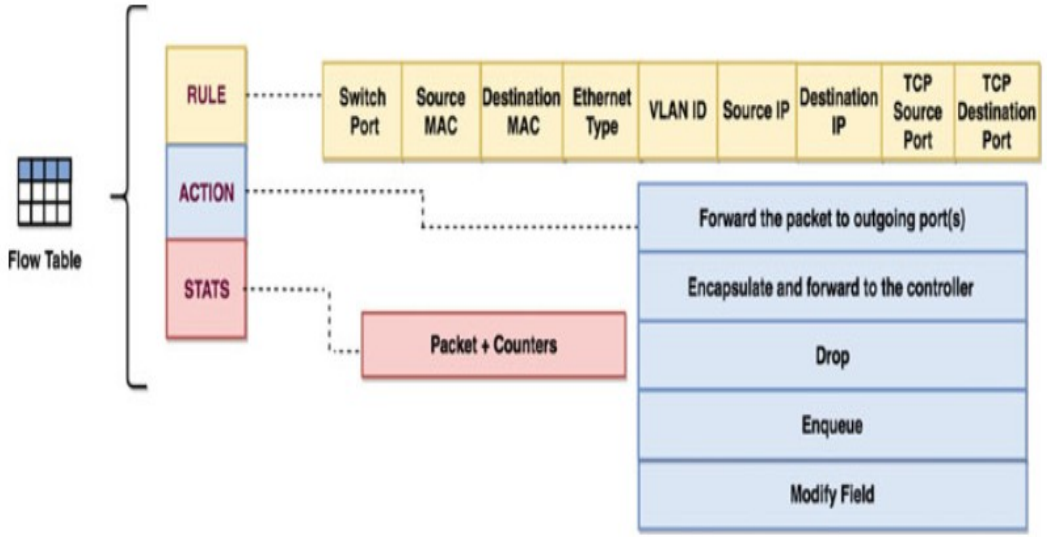
O controlador é responsável por manter, distribuir e atualizar as políticas e instruções para os dispositivos de rede. Ele pode determinar como tratar com os pacotes sem entradas de fluxo válidas e pode gerenciar a tabela de fluxo do *swicth* adicionando ou removendo entradas de fluxo no canal seguro (LI; MENG; KWOK, 2016).

2.1.3 Tabela de Fluxo

Em um dispositivo de encaminhamento habilitado para utilização do protocolo *OpenFlow* possui uma tabela de fluxo, a qual é composta por três partes: (1) correspondência da regra; (2) ações a serem executadas para os pacotes correspondentes; e (3) contadores para estatísticas de pacotes correspondentes (SAHARAN; GUPTA, 2019).

Conforme ilustrado na Figura 3, os campos de correspondência de regra incluem: porta do *switch*, *MAC* de origem, *MAC* de destino, tipo de *Ethernet*, ID de *Vlan*, IP de origem, IP de destino, porta de origem TCP e porta de destino TCP. As ações mais comuns são: encaminhar o pacote para a(s) porta(s) de saída; encapsular e encaminhar para o controlador; rejeita; enfileira; e modifica o campo.

Figura 3 – Tabela de fluxo



Fonte: Shaghaghi et al. (2020)

2.1.4 Controladores

O controlador é uma unidade central (mais importante), executa todas as tomadas de decisões em uma SDN, assim como recebe todo o tráfego enviado pelo *Switch OpenFlow*. Podendo determinar como lidar com os pacotes sem entradas de fluxo válidas e pode gerenciar a tabela de fluxo do *switch* adicionando ou removendo entradas de fluxo no canal seguro (LI; MENG; KWOK, 2016).

O controlador SDN pode ser implementado em um computador simples, o qual assume o papel de um servidor da rede. Existem diversos controladores, os quais se diferenciam pelas linguagens de programação ou o tipo de aplicação. A Tabela 1 apresenta algumas características dos principais controladores.

Tabela 1 – Características dos controladores

Controlador	***OpenSource***	**Linguagem**
OpenDayLight	Sim	JAVA
NOX	Sim	C++
POX	Sim	Python
Floodlight	Sim	JAVA
Ryu	Sim	Python

Fonte: Autor.

2.1.5 *Switch OpenFlow*

O objetivo principal da SDN é a separação entre o plano de dados o qual encaminha os pacotes, do plano de controle responsável pela configuração do plano de dados. O controlador realiza a comunicação com o *switch* para instruí-lo sobre como configurar o plano de dados, enviando comandos de modificação de fluxo que colocam regras na tabela de fluxo do *switch* (KUŹNIAR et al., 2018).

Para que sejam considerados *switches OpenFlow* é necessário que tenha suporte ao protocolo *OpenFlow*, podendo ser físico ou serem executados via *software* em simuladores, emuladores ou máquina virtual.

2.1.6 Aplicações SDN

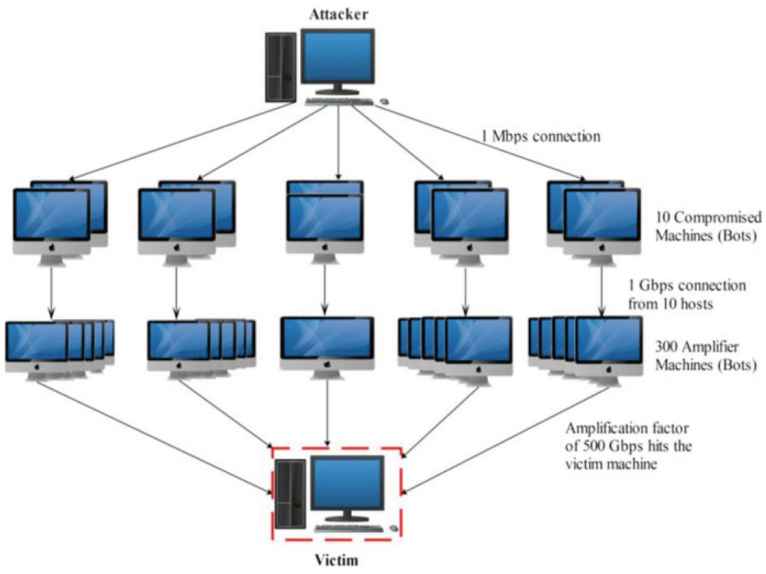
Segundo Hussein et al. (2016), a SDN é uma rede fisicamente distribuída, mas logicamente estruturada e controlada, oferecendo recursos ilimitados. A flexibilidade proporcionada pela sua arquitetura permite que ela seja aplicada em qualquer ambiente. Oferecendo assim, diversas funcionalidades podendo ser aplicadas em diversos setores.

As aplicações de rede podem ser divididas em cinco categorias: **Engenharia de tráfego** tem como objetivos: a diminuição do consumo de energia, aumentar a utilização da rede global e um balanceamento de carga otimizado; **Mobilidade e *Wireless*** proporciona mais facilidade no gerenciamento de diferentes tipos de redes sem fio, alocação de recursos de rádio, gerenciamento de entrega e execução do balanceamento de carga eficiente entre células; **Medições e Monitoramento** são aplicações que fornecem novas funcionalidades para outros serviços da rede e propostas que melhoram as características das redes SDN baseadas em *OpenFlow* como, reduzir a sobrecarga do plano de controle devido à coleta de estatísticas; **Segurança e Confiança**, são aplicativos que permitem aplicações de políticas de segurança realizadas no primeiro ponto de entrada da rede; **Redes de Data Center** permitem implantação rápida de redes de desenvolvimento para redes de produção, melhor gerenciamento da rede, otimização da rede, mecanismo e serviços de aplicação de políticas de segurança (KREUTZ et al., 2014).

2.2 Ataque Distribuído de Negação de Serviço (DDoS)

O ataque distribuído de negação de serviço (DDoS), é um ataque cibernético no qual são enviados pelo atacante excessivas requisições falsas para um servidor. Fazendo que os usuários legítimos obtenham a negação de serviço. Os ataques DDoS podem atacar a rede da vítima de diversas maneiras: TCP, UDP, inundação ICMP, inundação aleatória de IP e o uso de *botnets* (RAHMAN; QURAISHI; LUNG, 2019). A Figura 4 ilustra a utilização dos *bots* pelo atacante para a amplificação do ataque DDoS(UBALE; JAIN, 2020).

Figura 4 – Amplificação do ataque DDoS



Fonte: Ubale e Jain (2020)

2.2.1 Ataque de Esgotamento de Recursos

O objetivo deste ataque é causar o colapso dos principais recursos do sistema. Existem duas formas diferentes de implementação do ataque: na primeira a rede, os protocolos de camada de transporte e da aplicação são explorados pelo atacante; na segunda forma, são utilizados os pacotes malformados para a execução dos ataques.

2.2.2 Ataque de Esgotamento de Largura de Banda

O ataque de esgotamento de largura de banda tem como objetivo de consumir toda a largura de banda utilizando um exército para a realização do ataque. Assim, a vítima nega serviço para os seus usuários legítimos por um pequeno a grande período de tempo até que o ataque seja mitigado.

2.2.3 Ataque de Infraestrutura

É o tipo de ataque DDoS mais catastrófico, visto que seu objetivo é causar danos aos elementos cruciais da Internet. Sendo que ele segmenta a largura de banda de rede, também como os recursos do sistema de destino. Um exemplo de ataque à infraestrutura é o que visa o *Domain Name System* (DNS), especialmente os DNSs raiz que fornecem serviços para todos os usuários de Internet em todo mundo (MAHJABIN et al., 2017).

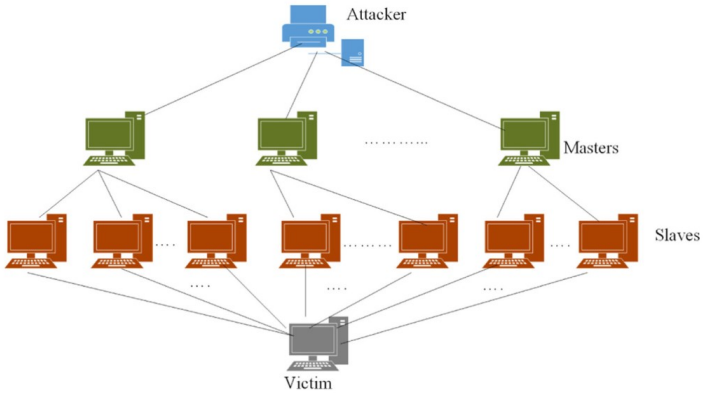
2.2.4 Ataque de Dia Zero

É conhecido por ataque de dia zero porque as vulnerabilidades dos sistemas são conhecidas no primeiro dia após o ataque. Visto que, estas vulnerabilidades de segurança eram desconhecidas, ou seja, uma ameaça que não havia sido corrigida e não tinha conhecimento público.

2.3 Fases do Ataque DDoS

O ataque DDoS possui uma estrutura básica apresentada na Figura 5. Apresenta três fases distintas e quatro componente diferentes. Os componentes são conhecidos como atacante, o *master*, o agente ou zumbis e uma máquina vítima ou alvo. Cada componente possui determinada função em cada fase do ataque são elas: intrusão em massa, instalação de *software* DDoS e disparando o ataque (MAHJABIN et al., 2017).

Figura 5 – Estrutura do ataque DDoS



Fonte: Mahjabin et al. (2017)

2.3.1 Fase 1 - Intrusão em Massa

Na primeira fase, é exigido muito tempo dos atacantes para reunir uma quantidade significativa de PCs zumbis. Os quais são chamados de *master* à medida que apontam e controlam outras máquinas. Com os *masters* reunidos, os agentes devem ser distribuídos para

os *host* vulneráveis. A distribuição dos agentes pode ocorrer de diversas formas, utilizando vulnerabilidades dos PCs agentes, como também por uma página da web. O método de distribuição mais popular é ludibriar o usuário para que realize o *download* do agente via página web ou uma rede P2P. Sendo que, este método impede a detecção e prevenção do recrutamento do agente, visto que o processo de *download* é uma atividade legítima (Choi et al., 2010).

De acordo com Mahjabin et al. (2017), existem diversas técnicas para a infecção de um *host*. As principais são: varredura aleatória - são *hosts* já infectados que verificam endereços IP com o objetivo de infectar novos *hosts*, varredura de lista de alvos - o atacante cria uma lista de *hosts* consideráveis potencialmente vulneráveis, além disso quando há sucesso na infecção de um *host* é propagada a metade da lista para este *host* infectado, varredura de permutação - é uma técnica inteligente em que a auto coordenação é introduzida para interromper várias sondagens do mesmo endereço IP, varredura topológica - nesse processo quando um *host* é infectado ele seleciona um alvo a partir de informações contidas no *host* e nesta técnica não há lista pré-produzida, no entanto o *bot* pode criar a sua própria lista e a varredura de sub-rede local - com um *host* já infectado, o mesmo pesquisa novos alvos na sua própria sub-rede local.

As varreduras de portas podem ser executadas de forma horizontal e vertical. Sendo que, a varredura vertical se caracteriza pela testagem de uma faixa ou por todas as portas de um determinado *host*. Em contrapartida, a varredura horizontal realiza a testagem na mesma porta em diferentes *hosts* da rede.

2.3.2 Fase 2 - Instalação de *Software* DDoS

Nesta fase, inicia-se com um número significante de dispositivos infectados se unindo a um exército comprometido o qual é conhecido como *botnet* e sendo controlado pelo invasor. No entanto, este controle utiliza-se de protocolos de aplicativos legítimos como HTTP, IRC, P2P, etc (Choi et al., 2010).

Existem três mecanismos básicos para a propagação do ataque que são: propagação de fonte central neste modelo o código de ataque se propaga de um servidor central para o *host* infectado, no encadeamento reverso é realizado o *download* do *host* infectado, sendo que (durante esta fase ocorre uma conexão entre o atacante e o *host* comprometido) e na propagação autônoma todos os códigos referentes ao ataque são enviados automaticamente para o sistema infectado durante o tempo de exploração (MAHJABIN et al., 2017).

2.3.3 Fase 3 - Disparando o Ataque

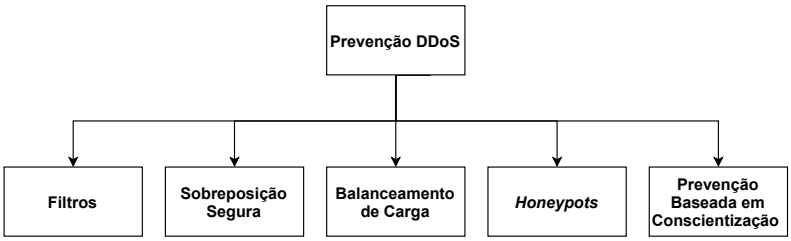
Na fase final é o momento onde o atacante comanda seu exército de *bots* (*botnet*) para iniciar e executar os ataques. Desta forma, a vítima é atacada de forma distribuída, inundando o sistema ou os principais recursos com um grande fluxo de pacotes. Além disso, o esgotamento das larguras de banda da rede não afetam apenas a vítima, mas também outros usuários e sistemas

que dependem desse alvo do ataque. Portanto, afetando de forma contundente a rede e os sistemas conectados a essa rede (MAHJABIN et al., 2017).

2.4 Técnicas de Prevenção de Ataques DDoS

A prevenção é uma técnica de defesa, com o objetivo de evitar que a infraestrutura de rede se torne uma vítima de ataque DDoS, sendo assim garantindo que o tráfego do ataque não seja bem-sucedido. De acordo com a pesquisa realizada por Mahjabin et al. (2017) existem métodos de prevenção como mostrado na Figura 6.

Figura 6 – Métodos de prevenção



Fonte: Mahjabin et al. (2017)

2.4.1 Filtros

A utilização de filtros que são aplicados no tráfego de rede, que são utilizados nos roteadores proporcionam um tráfego mais seguro na rede. De acordo com Mahjabin et al. (2017) existem diversas técnicas de filtragem, nas quais são apresentadas o sucesso e o fracasso na prevenção de DDoS.

A filtragem de entrada/saída é uma técnica que impede o tráfego de IP's falsificados. Portanto, se um invasor utilizar um IP falsificado que não corresponde a rede local, é descartado nos roteadores. Esta técnica garante uma prevenção considerável em ataques DDoS, quando utilizado a falsificação de um endereço IP. Porém, a filtragem de tráfego não tem sucesso nos casos em que o *bot* se utiliza de endereços válidos no momento do ataque.

2.4.2 Sobreposição Segura

É um método de prevenção que utiliza software para criar camadas de redes virtualizadas e separadas na parte superior da rede física. No entanto, com as camadas adicionais de software pode gerar uma sobrecarga no desempenho.

A rede de sobreposição é o ponto de entrada para a rede externa estabelecer uma comunicação com a rede protegida. Supõe-se que o isolamento possa ser alcançado ocultando seus endereços IP ou utilizando um *firewall* distribuído. Esse *firewall* garante que apenas o tráfego confiável dos *hosts* da rede de sobreposição (MAHJABIN et al., 2017).

2.4.3 *Honeypots*

Um ataque à uma rede pode ser detectado utilizando o *honeypot*, sendo um recurso de segurança desenvolvido para ser investigado, atacado e comprometido (Sembiring, 2016). Sendo um sistema ou dispositivo menos seguro utilizado para atrair atacantes. O *honeypot* tem o objetivo de parecer um item real dentro do ambiente. Além de ser uma armadilha, também é possível extrair informações importantes do atacante, como: registros de atividade de ataque; ferramentas e softwares utilizados para o ataque.

A técnica de *honeypot* pode reunir dados sobre os atacantes e suas atividades e auxiliar no reconhecimento das vulnerabilidades da rede. Além disso, os dados possibilitam descobrir solicitações de IP que devem ser bloqueadas (Rahman; Roy; Yousuf, 2019).

2.4.4 Balanceamento de Carga

Esta tecnologia tem como finalidade equilibrar a quantidade de trabalho de diferentes sistemas, a fim de evitar que o sistema fique sobrecarregado. O balanceamento de carga garante a resiliência à medida que redireciona o tráfego para outros servidores ativos e não atacados.

Segundo Gangadhara, Hasyagar e Damotharan (2019), o balanceamento de carga é crucial para o crescente tráfego da rede, para uma utilização otimizada de energia e recursos. As formas existentes para alcançar o balanceamento de carga são: através de protocolos, *software* especializado, *gateways* e divisores especializados.

2.4.5 Prevenção Baseada na Conscientização

Alguns dos ataques podem ser evitados com a conscientização dos usuários, adotando medidas preventivas em seu próprio sistema. Desta forma, garante a segurança de não ser atacado e não ser um *bot* em uma *botnet*. Algumas iniciativas adotadas pelo usuário podem ser eficazes para evitar um ataque DDoS como: a mudança de endereço IP, desabilitando serviços incomuns e a aplicação de *patches* de segurança.

2.5 Desafios na Segurança em Redes SDN

Nesta seção, destacamos os desafios relacionados a segurança em redes definidas por *software*, apresentados por diversos autores.

Segundo Li, Meng e Kwok (2016), ao desacoplar o plano de controle em um aplicativo centralizado, o próprio controlador SDN provavelmente se tornará o principal gargalo e um alvo principal para vários ataques, como os ataques de inundação e negação de serviço (DoS). Assim, ao atacar o plano de controle pode fazer com que ele fique indisponível para responder às solicitações do plano de dados.

A arquitetura SDN permite a implementação de diversos aplicativos. Por outro lado, cria falhas de segurança como: o controle de acesso não autorizado / não autenticado, violação e contradição das regras de fluxo. No entanto, a natureza inerente da operação SDN torna vulnerável, de modo que um atacante pode simplesmente tornar o controlador indisponível (RAHMAN; QURAISHI; LUNG, 2019).

De acordo com Sharma (2017), uma das formas mais comuns de ataques é a falsificação de mensagens para as interfaces *northbound* e *southbound*. Outras formas de ataques incluem o consumo de recursos do controlador, como o ataque DDoS, o ataque de *smurf* por inundação de pacotes e entre outros.

Em uma rede com uma grande quantidade de comutadores requer uma significativa quantidade de troca de mensagens entre o controlador e o comutador. Além disso, todo o tráfego excepcional de cada *switch* pode ser encaminhado diretamente ao controlador para a sua análise e a utilização de regras correspondentes. Tornando o controlador vulnerável, devido ao aumento de tráfego com novos tipos de pacotes fluindo na rede, como exemplo o ataque DoS (HUSSEIN et al., 2016).

Portanto, diante das características arquiteturais das redes SDN, a preocupação com a segurança do controlador é clara diante das diversas formas de ataques que este tipo de rede está exposta. Os autores (SHARMA, 2017), (LI; MENG; KWOK, 2016) e (HUSSEIN et al., 2016), apontam os ataques DoS e DDoS como um dos principais desafios com a segurança da SDN, visto que este tipo de ataque pode tornar o controlador indisponível. Assim, como o ataque de negação de serviço e o ataque distribuído de negação de serviço geram um grande tráfego de pacotes, os quais direcionados ao controlador SDN ocasionaria o esgotamento de recursos.

2.6 Considerações Finais

Este capítulo apresentou o referencial teórico que forneceu embasamento para este trabalho de mestrado. Foram abordados os conceitos de redes definidas por *software*, ataque distribuído de negação de serviço e técnicas de prevenção de ataques DDoS. Em seguida, foram apresentados os principais desafios relacionados à segurança da SDN, com destaque em ataques DDoS.

3

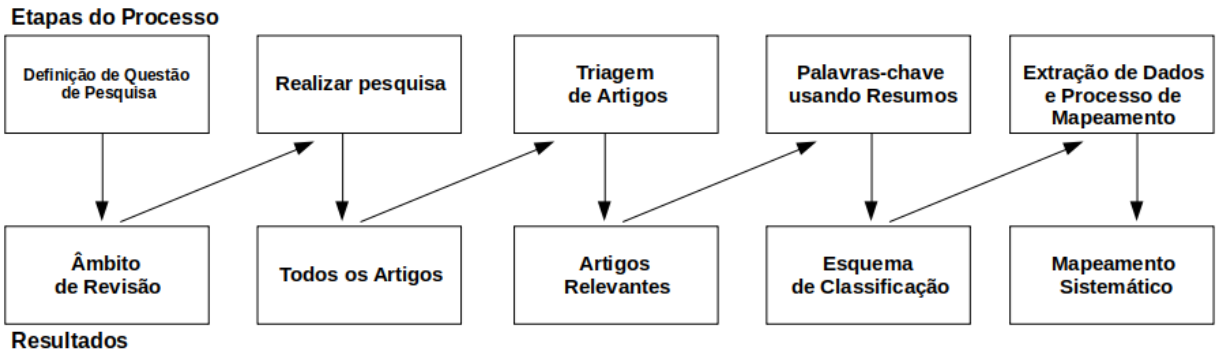
Trabalhos Relacionados

Neste capítulo, foram identificados e discutidos trabalhos de pesquisas em ambientes SDN, relacionados aos ataques DDoS com o intuito de identificar quais são as técnicas e os mecanismos que vem sendo propostos para evitar este tipo de ataque. Para a seleção do trabalho, foram selecionados os artigos que abordam prevenção de ataques DDoS em SDN.

3.1 Mapeamento Sistemático

Com o objetivo de mapear os estudos sobre as técnicas e métodos de prevenção de ataques DDoS em redes SDN foram realizadas por etapas essenciais no processo deste estudo. Segundo Petersen et al. (2008), no qual é necessária a definição de perguntas de pesquisas, busca de artigos relevantes, a triagem dos artigos, palavras-chave e extração e mapeamento dos dados, como apresentado na Figura 7.

Figura 7 – Processo mapeamento sistemático



Fonte: Adaptada de Petersen et al. (2008)

Como primeira etapa no processo serão definidas as questões de pesquisa, seguida da estratégia de busca onde serão utilizados os critérios para a seleção dos artigos.

3.1.1 Questões de Pesquisa

O objetivo deste estudo é mapear os artigos publicados nos últimos seis anos completos, compreendendo o período de janeiro de 2015 à dezembro de 2020, que abordem a prevenção de ataques de negação de serviço distribuído (DDoS) em redes definidas por software (SDN). Assim, foi definida a seguinte questão principal de pesquisa que norteia o estudo: "Quais técnicas e métodos de prevenção de ataques DDoS são úteis no contexto das SDNs?". Para responder está questão, foram extraídas cinco outras questões apresentas a seguir: Q1) Como foram realizados os ataques DDoS em redes SDN nos artigos selecionados? Q2) Quais os métodos estão sendo utilizados para a prevenção de ataques DDoS em redes SDN? Q3) Quais os algoritmos foram utilizados nas pesquisas para a prevenção de ataques DDoS em redes SDN? Q4) Como foi realizado a pesquisa (Simulação/Experimento/Medição/Modelos Analíticos)? Q5) Quais ferramentas foram utilizadas para a simulação da rede SDN?

3.1.2 Estratégia de Busca de Seleção

As fontes de dados foram selecionadas a partir da indicação de pesquisadores na área, no Quinquênio 2015-2020, que tratem da temática segurança, dos repositórios e indexadores de artigos e periódicos de relevância para área de redes de computadores como o *IEEE Xplore Digital Library*, *ACM Digital Library*, *Scopus*, *SpringerLink* e *Science@Direct*, Para a realizão da busca foi utilizado o portal de periódicos da Capes (https://www.periodicos.capes.gov.br) para obter acesso aos artigos disponíveis nessas bases.

As palavras chaves e seus sinônimos definidos para a *string* de busca foram: **DDoS** (*Distributed Denial of Service / DDoS Attack / Distributed DoS Attack*), **SDN** (*Software Defined Networking / Software-Defined-Networking / Software-Defined Networking*) e **Prevent** (*Prevention / Attack Prevetion*). Desta maneira, a *string* de é apresentada no Quadro 1 .

Quadro 1 – *String* de busca

(("DISTRIBUTED DENIAL OF SERVICE" OR "DDOS ATTACK" OR "DISTRIBUTED DOS ATTACK") AND ("SOFTWARE DEFINED NETWORKING" OR "SDN" OR “SOFTWARE-DEFINED-NETWORKING” OR “SOFTWARE-DEFINED NETWORKING”) AND ("TO PREVENT" OR "PREVENTION" OR "ATTACK PREVENTION"))

Fonte: Autor.

3.1.3 Critérios de Seleção

Foram definidos alguns critérios de inclusões, nos quais tem por objetivo filtrar os artigos encontrados e selecionar apenas os que possuem relevância à pesquisa, são eles: CI-1) Publicações com o foco em detecção e prevenção de ataques DDoS em redes SDN. CI-2) Publicações com o foco em mitigação e prevenção de ataques DDoS em redes SDN. CI-3) Publicações com o foco a prevenção de ataques DDoS em redes SDN. Estes critérios foram utilizados baseando-se no resumo (*abstract*) dos artigos encontrados.

Para excluir os trabalhos que não são relevantes a pesquisa, definiu-se os seguintes critérios de exclusão: CE-1) Publicações em idioma diferente do inglês e português. CE-2) Publicações cuja a versão completa esteja indisponível. CE-3) Publicações com o foco em detecção de ataques DDoS em redes SDN. CE-4) Publicações com o foco em mitigação de ataques DDoS em redes SDN. CE-5) Publicações com o foco em mitigação e detecção de ataques DDoS em redes SDN. CE-6) Publicações que não abordem o tema da pesquisa.

3.1.4 Análise de Resultados

A fim de obter estudos relacionados ao interesse da pesquisa, a *string* de busca foi utilizada nas consultas nas bases de dados. Foram identificados 294 trabalhos, conforme detalhamento na Tabela 2.

Tabela 2 – Resultado da pesquisa

Base	**Resultado da Pesquisa**
IEEE Digital Library	42
Scopus	66
ACM Digital Library	49
SpringerLink	288
Science@Direct	03
Total	**448**

Fonte: Autor.

Dentre os 448 artigos encontrados, após a leitura dos *abstracts*, 334 foram excluídos, 95 artigos estão duplicados e 19 foram selecionados utilizando os critérios de inclusão para a realização deste mapeamento sistemático. Os trabalhos aceitos e passaram por leitura completa. Sendo que, todos os trabalhos de pesquisa atenderam aos critérios de inclusão formalizados no protocolo do mapeamento sistemático.

Na Tabela 3 apresenta o resultado do critério de exclusão por base pesquisada.

Tabela 3 – Selecionados pelo critério de exclusão

Base	**Critério de Exclusão**	**Trabalhos Duplicados**
IEEE Digital Library	12	25
Scopus	33	24
ACM Digital Library	44	04
SpringerLink	242	42
Science@Direct	03	00
Total	**334**	**95**

3.2 Trabalhos de Pesquisas Selecionados

3.2.1 Trabalhos sobre Prevenção

Trung et al. (2015), propõe uma arquitetura de rede para a prevenção de ataques DDoS com base em parâmetros estatísticos do tráfego de entrada *on-the-fly* e um algoritmo de lógica difusa. O aplicativo é executado no controlador SDN e aplica o algoritmo híbrido (os dados estatísticos e a lógica difusa) esses dados são analisados e realizam a verificação se o servidor está sob um ataque DDoS ou está em seu estado normal. Embora exista uma variedade de algoritmos para a prevenção dos ataques, os mesmos dependem principalmente das características específicas dos ataques DDoS. Nas soluções que utilizam mapas auto-organizáveis (SOM) e técnicas de aprendizado de máquina para prevenção de ataques DDoS apresentou horas e uma enorme matriz de cálculos para concluir o treinamento SOM, bem como a tabela de treinamento para a técnica de aprendizado de máquina. Foram analisados os aquivos de log do tráfego de rede de um ISP vietnamita no qual ambos tráfegos normais e de anomalia. Foram encontrados três parâmetros que possa ser critério para a detecção de um ataque DDoS: 1) a taxa de pacotes em relação entre a hora de chegada (*IAT*) de (0 - 0,2 X e-3) segundos; 2) a taxa de vazões que possuem apenas um pacote e 3) número de fluxos para um servidor. O autor propõe um mecanismo de prevenção utilizando um esquema de prevenção híbrido, baseado nos critérios identificados no trabalho e um sistema de inferência difusa para detectar o risco de ataque.

A proposta criada por Zhu e Chang (2019) foi um mecanismo de controle e encaminhamento baseado na identificação criptografada da SDN, onde todos os pacotes são encapsulados com uma identificação criptografada e assinadas por chaves privadas com base na identificação da criptografia. Essa identificação é gerada com base na informação do dispositivo como a identidade do usuário, atributo do arquivo ou conteúdo comercial, e o encaminhamento de rede é definido com base na identificação da criptografia. Com a utilização dos *switches* SDN para a verificação das fontes dos dados, se os pacotes enviados para o controlador são razoáveis, reduzindo a probabilidade de ataques DDoS. A verificação da assinatura na entrada e saída da rede tem como finalidade assegurar que os pacotes verdadeiros sejam entregues nos dispositivos de destino. Os resultados do experimento mostraram que o mecanismo garante a disponibilidade da rede e como trabalho futuro o autor pretende aprimorar a verificação do fluxo para melhorar o

desempenho do encaminhamento do fluxo.

O mecanismo de prevenção proposto por Singh, Khan e Agrawal (2015), foi desenvolvido para evitar os ataques DoS ou DDoS em seus estágios inciais prejudiquem a SDN. O trabalho aborda um tipo de ataque DoS, baseado em uma área geográfica específica e utilizando-se de um conjunto de diretrizes e políticas para evitar este tipo de ataque antes de danificar a rede. A estratégia utilizada pelo autor foi a criação de etapas, onde a primeira etapa foi de identificar o endereço proveniente de um tráfego pesado e utilizar o *buffer* para manter os pacotes em uma fila. Caso a política anterior não surta efeito, será iniciada a segunda etapa onde o tráfego será identificado com a hora, podendo ser bloqueado por algum intervalo de tempo e a terceira etapa é gerado um pacote de solicitação *eco* para a fonte IP para diminuir a taxa de transferência de envio. Nos três cenários testados as políticas aplicadas como mecanismo de prevenção obteveram sucesso.

A arquitetura proposta por Ngo, Pham-Quoc e Thinh (2019), é baseada em *hardware* para os dispositivos seguros de encaminhamento SDN no plano de dados. Utilizando a computação paralela como FPGA e GPU, para a criação de um *switch OpenFlow*, com quatro métodos de segurança para a proteção do sistema contra ataque múltiplos. A função *OpenFlow* realiza pesquisa de fluxos para novos pacotes recebidos. O pré-scanner inclui núcleos de varreduras leves e focado na prevenção de ataques DDoS. O *F-NIDS* é o sistema de detecção de intrusão rede e o *F-ANIDS* que tem o sistema de detecção de intrusão de rede baseado em anomalias com modelos de redes neurais treinados. A arquitetura proposta é composta por diversos blocos com suas respectivas funcionalidades. Com a utilização das quatro funções principais do sistema foi possível ter um aumento na segurança, mas também fornecer a capacidade de comutação de alta velocidade baseada no padrão *OpenFlow*.

O mecanismo desenvolvido por Bose et al. (2019), preveni os ataques DDoS no nível do *switch*, incorporando uma segurança utilizando *blockchain* para os canais de interação entre o plano de dados e plano de controle. No trabalho proposto é implementado um controlador SDN virtual, o qual gerencia todas as tarefas de transmissão e gerenciamento as entradas de fluxo nas tabelas. O controlador principal fica responsável por gerar as regras de fluxo e atualizar a tabela mestre. O *blockchain* cuida dos blocos gerados em cada mudança para atualizar a tabela *OpenFlow* do controlador com base nas regras de fluxo assim gerado. Cada bloco consiste em uma lista de informações valiosas, utilizando o *hash* gerado do bloco anterior com o seu vetor de inicialização. Assim, fazendo uma interconexão e, consequentemente, uma cadeia de blocos. Conclui-se, que o *blockchain* fornece uma maneira estável e eficiente de proteção dos *switches* aos invasores da arquitetura SDN.

3.2.2 Trabalhos sobre Detecção

O modelo de detecção proposto Bae et al. (2015), tem o objetivo a identificação de PC DDoS e zumbis em uma rede local SDN durante o processo de ataque. Através do aplicativo

instalado no controlador SDN é possível a detecção e registro de pacotes falsificados em um banco de dados, assim assume-se que este é um pacote de ataque DDoS, cuja origem, 'zumbi'. No banco de dados deve possuir uma tabela para o armazenamento de determinadas informações recebidas pelos comutadores SDN. Fundamentalmente, a tabela deve ter cinco informações do comutador: número da porta de entrada, endereço de destino, endereço de origem, controlador e o identificador do *switch* que enviou o pacote para o controlador. Será realizado uma análise sobre a existência de pacotes com a mesmo endereço de destino. Porém com outras informações diferentes. Segundo o autor, esses pacotes podem ser de um ataque DDoS e os *host* responsáveis pelo envio dos mesmos controlados pelo mesmo atacante. O autor conclui, que pode ser reduzido e bloqueado o tráfego de ataques, melhorando toda a segurança da rede local através da detecção de PCs zumbis.

O autor Kim et al. (2017) propõe, uma nova estrutura de segurança utilizando a rede definida por *software*, para o armazenamento do histórico de consultas DNS como evidência para distinguir respostas DNS normais de pacotes de ataques DDoS. O esquema proposto utiliza o método estrito um a um entre as solicitações e respostas DNS. O esquema pode detectar essas respostas, verificando se existe uma solicitação DNS que corresponda a uma determinada resposta DNS. O sistema possui dois componentes principais: *switch* e o controlador SDN. Sendo que, no *switch* são realizadas diversas verificações com os pacotes com o objetivo de identificar, analisar e armazenar as informações referentes aos pacotes DNS. Caso o comutador, não possua capacidade de armazenamento disponível localmente; os pacotes são redirecionados para o controlador onde são realizados os procedimentos de identificação, analise dos pacotes e pode ter essas informações das solicitações DNS armazenadas em um servidor externo de banco de dados com maior capacidade de armazenamento. O autor conclui, que o esquema proposto fornece um armazenamento escalonável e centralizado para as solicitações DNS utilizando a rede definida por *software*, podendo remover a possibilidade de pacotes falsos positivos. Sendo assim, pode-se impedir completamente ataques de amplificação DNS sem acarretar atrasos significativos entre as comunicações do *host* e o controlador SDN.

3.2.3 Trabalhos sobre Mitigação

A pesquisa desenvolvida por Sharma (2017) apresenta um sistema auto-mitigante baseado em sistemas multiagentes (SMA) para SDN. A arquitetura do agente se encontra na camada de controle que se comunica com a camada de dados baseados nos serviços exigidos. O módulo gerenciador de topologia é utilizado para extrair a visão topológica da rede. O mecanismo de raciocínio utiliza fluxo de pacotes e um mecanismo baseado em limites para as regras que tendem a mudar continuamente as políticas no controlador. Outro componente crucial é uma linguagem formal para representar o conhecimento e o mecanismo de raciocínio para as entidades da rede. O comportamento dos agentes são muitos afetados pela topologia, dependendo dos eventos gerados e o estado da rede. A base do agente é implantada no plano acima do controlador da rede o

qual apresenta uma visão topológica, ajuda na criação do conhecimento do domínio, detecta o ambiente da rede, o fluxo de tabelas e o de pacotes. O trabalho prova que a arquitetura pode detectar de forma rápida a ameaça e fornecer a migração e recuperação com baixo tempo de inatividade e degradação do desempenho da rede.

3.2.4 Trabalhos sobre Detecção e Mitigação

Apesar do foco do trabalho de Rahman, Quraishi e Lung (2019) ser a detecção e mitigação de ataques DDoS na SDN, o autor avaliou algumas técnicas de aprendizado de máquina para impedir o ataque DDoS em redes SDN. O autor descreve o ataque DDoS sendo muito difícil de ser detectado pela sua semelhança com o tráfego normal da rede. No entanto, todos os ataques DDoS possuem alguns recursos comuns, os pacotes com o tamanho diferente em relação ao tráfego normal e pacotes maliciosos com o mesmo endereço de destino e porta. Uma das abordagens mais populares entre os pesquisadores para fornecer um sistema de detecção de intrusão (IDS) eficiente é o uso de técnicas de aprendizado de máquina em SDN. Neste trabalho foi investigado os seguintes algoritmos de aprendizado de máquina: SVM, floresta aleatória, J48 e K-NN. Os resultados obtidos no experimento realizado pelo autor mostra que o algoritmo J48 foi o mais adequado para a prevenção de ataques DDoS.

3.2.5 Trabalhos sobre Prevenção e Detecção

O trabalho Sahoo et al. (2018) apresenta um mecanismo robusto de detecção de DDoS, para evitar ataques ao plano de controle da SDN. Foram utilizados sete técnicas de Machine Learning. Baseado em dados históricos de diferentes tipos de ataques DDoS, como o Smurf, inundação UDP e inundação HTTP, foi possível prever e classificar com precisão estes ataques. Os algoritmos utilizados no trabalho são : K-Nearest Neighbor (KNN); Naive Bayes (NB); Support Vector Machine (SVM); Random Forest (RF); Linear Regression (LR); Desicion Tree (DT); Artificial Neural; Network (ANN). O experimento realizado no trabalho demonstra que os algoritmos de ML, podem auxiliar e prever o ataque com precisão. O algoritmo de LR apresentou uma média de precisão de 98,652%. Sendo que, o algoritmo RF obteve 98,409% de precisão com menos tempo de duração em relação ao algoritmo LR. Como trabalho futuro o autor (SAHOO et al., 2018) realizará testes para Smurf e UDP-Flood, sendo comparados com outras técnicas de ML.

A pesquisa Xing et al. (2018) propõe um novo esquema de sistema de detecção e prevenção de intrusões (IDPS) com dimensionamento automático para a proteção da nuvem contra ataques DoS e DDoS. Na primeira etapa é detectado em tempo real o tráfego anormal com base no fluxo de estatísticas coletadas nos comutadores SDN. Posteriormente, o tráfego anormal é redirecionado pelo controlador SDN para um contêiner utilizando o *Docker* onde é limpo pelo *Snort* e direciona o tráfego legítimo para os seus destinos originais. Com base nos resultados o sistema executa algumas ações específicas, como criar regras para o bloqueio do

tráfego dos atacantes. Foi utilizada a ferramenta *Hping3* para a execução dos ataques DoS e DDoS e a ferramenta *Scapy* para a simulação do tráfego legítimo. Com base na implementação do protótipo, foi validada a eficácia do sistema na defesa do ataque DDoS.

Os autores de Pande et al. (2018), modelaram um mecanismo de prevenção de ataques DDoS que ocorrem em iguais e diferentes domínios com o auxílio do controlador SDN. Os controladores são programados para identificar vítimas e atacantes aplicando o mecanismo de defesa. O mecanismo identifica os *hosts* que são vítimas e os atacantes na rede e o plano de mitigação é executado pelo controlador em seu respectivo domínio e leva em consideração se a tentativa maliciosa de invasão está ocorrendo na mesma topologia. O tráfego da rede é monitorado pelo controlador, onde são solicitas estatísticas de fluxo de cada *switch* para a verificação do estado do ataque. Se o pacote possuir o TLL inválido, ou se o pacote for LLDP ou se do tamanho o pacote for superior que o aceitável ele será descartado. Da mesma forma, se o endereço IP e o MAC do pacote recebido não forem encontrado em uma tabela, o pacote será descartado. Senão o pacote é verificado na tabela ARP e caso o endereço IP e o MAC se encontre na tabela, o endereço IP e o MAC é retornado. O mecanismo garantiu uma alta taxa de detecção de ataques DDoS reduzindo o número de falsos positivos.

O trabalho proposto por Huang e Yu (2019) é uma extensão do *FuzzyGuard* para prevenção de ataques DDoS em rede de sensores sem fio definidas por software (SDWSN), incluindo componentes de construção da rede de controle, detecção de ataques difusos e supressão probabilística de fluxo. O componente de construção de rede gera regras de controle e regras de dados e as distribui pelos sensores, o componente de detecção de ataque difuso é implementado por um sistema de inferência difusa que consiste em quatro módulos: difusão, inferência difusa, *defuzzificação* e base de regras e o componente de supressão probabilística do fluxo adota uma maneira não discriminatória de restringir os fluxos de entrada de pacotes, classificando o estado da rede de sensores da seguinte forma: probabilidade zero, probabilidade baixa e probabilidade de penalidade. Os resultado mostram que o experimento pode detectar efetivamente ataques de saturação em diferentes modos protegendo o encaminhamento normal dos fluxos de dados sob ataques.

O artigo desenvolvido por Alparslan et al. (2017) propõe uma arquitetura que melhora a resiliência contra ataques DDoS, aproveitando as funções virtualizadas de rede (NFV) e as redes definidas por *software* (SDN). A arquitetura calcula o posicionamento das NFV's e as rotas dos serviços que são determinados de acordo com a lista de serviços e desempenho, configurando a rede de acordo com a otimização, que é chamado de modo regular. Sendo que, a arquitetura possui o modo de proteção onde os caminhos secundários de todos os serviços são filtrados por uma VNF antes de seguir para outra função virtualizada de rede, a fim de impedir a sua saturação de requisição devido ao tráfego de um ataque DDoS. Após a detecção do ataque o controlador SDN altera os caminhos dos serviços para as rotas secundárias previamente calculadas, fazendo que o ataque DDoS seja impedido imediatamente sem que seja preciso o cálculo de novas rotas

ou movimentação de NVF's. Os resultados obtidos pelo experimento revelam que a arquitetura melhora a taxa de absorção do tráfego DDoS.

A abordagem levantada por Hussein et al. (2016), propõe uma arquitetura de segurança em SDN que permite um bom equilíbrio entre o desempenho e os recursos de segurança da rede, utilizada para impedir e rastrear a origem dos ataques DDoS no controlador ou em diferentes *hosts* da rede. A arquitetura possui um terceiro plano, denominado plano de segurança além dos planos de dados e controle da SDN. O plano de segurança estabelece uma conexão entre os comutadores e o controlador por meio de um agente, sendo realizada por uma porta exclusiva e diferente da qual é utilizada pelo plano de controle. A técnica visa detectar ataques DDoS de *hosts* ou *bots* maliciosos para outros *hosts* ou ao controlador, analisando e rastreando sua contagem de pacotes visando um específico endereço IP para ser comparado com um limite; mantendo um banco de dados para mapear a porta do *switch* MAC-IP com o objetivo de detectar IP's e pacotes falsificados; um alerta é enviado com o IP de origem do atacante para ser bloqueado e em caso de falsificação IP é utilizado um procedimento de rastreio para identificar o *switch* e a porta para ser bloqueado. A técnica de prevenção foi implementada e testada em diferentes cenários para provar a eficácia da proposta. Os autores propõem como trabalhos futuros implementar uma dinâmica completa para ser testado em um *hardware* de médio porte e poderia evoluir para uma interface gráfica Web para mostrar os fluxos de tráfego e o uso da largura de banda da rede em tempo real.

O esquema proposto por Qin et al. (2019), é o posicionamento de um controlador de *backup* com o reconhecimento de ataques, tendo como objetivo a redução de custos no planejamento da rede e bem como no fornecimento ininterrupto de serviços para a SDN. A abordagem é realizada através de uma combinação híbrida de estratégia heurística para a geração de uma variedade de combinações em um curto intervalo de tempo em comparação com o método de combinação K. Como técnica de prevenção foi proposto o método de posicionamento *attack-aware recovery controller link-switch* (ARC) o controlador de *backup* será conectado a um controlador fixo em um nó para facilmente ser substituído por um mau funcionamento ou devido a um ataque. O ARC utiliza um algoritmo para a geração de uma matriz de combinações distintas e heurísticas do controlador. A recuperação da rede é planejada, antecipando os ataques e falhas em qualquer nó e a substituição dos controladores que estão sob ataque pelos controladores de *backup* adicionais. O esquema ARC é capaz de produzir matriz de combinação de controladores em menor período de tempo em comparação com o método de combinação k.

Ahuja e Singal (2019) propõe, uma contramedida de detecção e prevenção conhecida como vigilância contínua, analisando continuamente as estatísticas de tráfego no *switch* com o objetivo de detecção do ataque de algum nó malicioso. Esta detecção e prevenção é realizada em duas fases, sendo que a primeira analisa a mudança na taxa de pacotes e largura da banda, sendo que na segunda fase o ataque é evitado alterando a lógica de encaminhamento do *host* na tabela de fluxo. Como trabalhos futuros o autor propõe a utilização dos *logs* de trafégo e algoritmos de aprendizagem de máquinas para a detecção e prevenção de ataques DDoS.

Lee, Chang e Syu (2020), apresenta um sistema de detecção e prevenção de intrusão para evitar ataques de foça bruta de *shell* seguro (SSH) e ataques distribuídos de negação de serviço em SDN. O autor realiza a comparação de quatro modelos de aprendizagem profundo, incluindo *Multilayer Perceptron* (MLP), *Convolutional Neural Network* (CNN), *Long Short-Term Memory* (LSTM) e *Stacked Auto Encoder* SAE para a avaliação do mecanismo de IDPS proposto. Para a identificação do ataque DDoS é utilizado um conjunto de dados DDoS baseado em fluxos que contém a sequência de comprimento do pacote em segundo para cada pacote de entrada com base na mesma conexão com diferentes pacotes e o comprimento máximo da sequência é defina como 32. O modelo MLP obteve uma precisão de 99,9% com o tempo de prevenção de 38,4 ms para a detecção de ataque de força bruta SSH e para o ataque DDoS foi de 98,3% com o tempo de predição de 27,2 ms.

3.2.6 Trabalhos sobre Prevenção e Mitigação

Saharan e Gupta (2019), pesquisou a utilização da SDN para tornar redes subjacentes inteligentes o suficiente para impedir ataques DDoS baseados em sistema de nomes de domínios (DNS). As duas hipóteses pesquisadas são: supondo que a rede principal da Internet esteja habilitada para a SDN, neste caso os algoritmos de roteamento da camada de rede são modificados; supondo que a rede principal da Internet não esteja habilitada para a SDN, neste caso uma barreira separada será criada utilizando a SDN pelo qual passará todo o tráfego DNS. Os resultados mostram a viabilidade das hipóteses apresentadas, uma solução para a falsificação DNS em uma rede SDN, foi a implementação de uma estrutura que insere uma inteligência apropriada na própria rede capaz de detectar qualquer atividade e tomar ações corretivas na camada de rede para anular esse comportamento. O mecanismo *KarmaNet* forçou a rede subjacente encaminhar o pacote de resposta DNS para o atacante DDoS para se auto-paralisar; na atenuação de ataques de amplificação de DNS foi utilizado um conjunto de roteadores SDN geograficamente distribuídos para criar uma barreira por onde passa todo o tráfego de entrada e saída dos usuários, sendo que todo o tráfego de ataque proveniente do tráfego de entrada é bloqueado e o restante é enviado para o destino.

3.3 Resultados e Discussões

A Tabela 4 apresenta as respostas obtidas para as questões de pesquisa. A ferramenta *Hping3* foi utilizada em três artigos para simular o ataque DDoS, mas na maioria dos trabalhos nenhuma ferramenta específica foi mencionada ou os autores desenvolveram suas próprias ferramentas para esse fim.

Dentre os trabalhos selecionados, 37,5% implementaram algoritmos de aprendizagem de máquina em suas pesquisas. Os algoritmos utilizados por Sahoo et al. (2018), Trung et al. (2015), Rahman, Quraishi e Lung (2019), Huang e Yu (2019), Sharma (2017) e Ngo, Pham-Quoc e Thinh

(2019) foram: *K-Nearest Neighbor (KNN), Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF)*, Regressão Linear (RL), Combinação de Limite, Sistema de inferência Fuzzy e J48.

Tabela 4 – Questões da pesquisa.

	Q1 - Como foram realizados os ataques DDoS ?	Q2 - Quais os métodos de prevenção foram utilizados ?	Q3 - Quais os algoritmos foram utilizados para a prevenção ?	Q4 - Como foi realizado a pesquisa ?	Q5 - Quais as ferramentas foram utilizadas para a simulação ?
(SAHOO et al., 2018)	Foi utilizado um conjunto de dados desenvolvido por Mouhammd et al.	*Machine Learning*	K-Nearest Neighbor (KNN) Naive Bayes (NB) Support Vector Machine (SVM) Random Forest (RF) Regressão Linear (LR)	Modelos Analíticos	- Sistema Operacional Ubuntu 14.04 - Linguagem de Programação Python
(TRUNG et al., 2015)	Ataques reais.	*Machine Learning* / *Análise de fluxo de Tráfego*	Combinação de Limite Sistema de inferência Fuzzy	Medição	ISP Vietnamita
(XING et al., 2018)	Criado pelo o autor um cenário de envio de 10 solicitações por segundo ao Servidor WEB.	Estatística de Fluxo Sistema de Detecção e Prevenção de Intrusão (IDPS)	Não Mencionado.	Simulação	Docker e Snort
(RAHMAN; QURAISHI; LUNG, 2019)	*Hping3*	*Machine Learning*	K-Nearest Neighbor (KNN) Random Florest (RF) Suport Vector Machine (SVM) J48	Simulação	- Sistema Operacional Ubuntu 18.10 - Mininet 2.3 - Controlador RYU 4.3 - Weka 3.9.3 - Tshark - Linguagem de programação Python.
(PANDE et al., 2018)	Não mencionado.	Estatística de Fluxo	Combinação de Limite	Simulação	- Mininet - Controlador RYU - Linguagem de progamação Python.
(HUANG; YU, 2019)	Realizado pelo autor com a alteração de pacote.	*Machine Learning*	Sistema de Inferência Fuzzy	Simulação	- Sistema Operacional 12.04 - VMWare Player 12 - Contiki 3.0 - Linguagem de progamação Python.
(ALPARSLAN et al., 2017)	Não mencionado.	Análise de fluxo de Tráfego /Encaminhamento de Tráfego.	Não Mencionado.	Simulação	Não Mencionado
(SHARMA, 2017)	Não mencionado.	Machine Learning / Multi-Agentes	Mecanismo de inferência	Simulação	- Controlador POX
(ZHU; CHANG, 2019)	Não foi realizado.	Controle e encaminhamento baseado na identificação cifrada.	Algoritmos de Criptografia	Experimento	- OpenDaylight - OVS - Linguagem de programação C++.
(SAHARAN; GUPTA, 2019)	Não foi realizado.	Análise de fluxo de Tráfego	Algoritmos de roteamento	Modelos Analíticos	Não Houve
(SINGH; KHAN; AGRAWAL, 2015)	Criado pelo autor.	Análise de fluxo de Tráfego / Conjunto de Diretrizes e Políticas.	Não Mencionado.	Simulação	Mininet 2.2.0 - Ubuntu-14.10 - Virtual Box - openVswitch - Wireshark - Xming - Controlador NOX
(HUSSEIN et al., 2016)	*Hping3 e hulk.*	Análise de fluxo de Tráfego / Criação de um novo Plano	Não Mencionado.	Simulação	- Mininet - Ubuntu 14.04.2 - VirtualBox 3.2.10 - openVswitch - Controlador POX - Linguagem de Programação Python
(QIN et al., 2019)	Não foi realizado.	Backup de Controlador SDN	Algoritmo de Combinações	Experimento	- Windows 10
(NGO; PHAM-QUOC; THINH, 2019)	*Hping3*	*Machine Learning* Redes Neurais Artificiais	Não Mencionado	Experimento	- NetFPGA-10G -GTX Geforce 1080 G1 -Linux Mint 17 Qiana -Ubuntu 16.04 -Ubuntu 14.04 -*Switch* Cisco Catalyst 2960-s
(BAE et al., 2015)	Não Mencionado	Análise de Fluxo Identificação de PCs Zumbis	Não Mencionado	Simulação	Não Mencionado
(KIM et al., 2017)	Criado pelo autor.	Análise do histórico de consultas DNS	Não Mencionado	Simulação	- Mininet 2.3.0 - Uuntu 12.04.3 - VMware Workstation 12.5.4 - Open vSwitch 1.10.0 - POX 0.2.0
(AHUJA; SINGAL, 2019)	Não Mencionado	Análise de Fluxo de Dados	Não Mencionado	Simulação	- Mininet - Controlaldor Ryu
(LEE; CHANG; SYU, 2020)	Hping3	*Deep Learning*	*Multilayer Perceptron* (MLP) *Convolutional Neural Network* (CNN) *Long Short-Term Memory* (LSTM) *Stacked Auto Encoder* SAE	Experimento	- Controlador Ryu - Open vSwitch - VMware ESXi - Ubuntu 14.04
(BOSE et al., 2019)	Não Mencionado	*Blockchain*	Não Mencionado	Simulação	- Controlador Pox

Com a categorização apresentada na Tabela 4, foi possível identificar que 68,75 % dos trabalhos selecionados utilizavam ferramentas de simulação para estudar os ataques em infraestruturas de redes SDN. Adicionalmente, notamos que 7 trabalhos utilizaram o sistema operacional Ubuntu (GNU / Linux) e apenas 1 utilizou o sistema operacional Windows 10. A tabela 4 também apresenta 5 trabalhos que usaram Python como linguagem de programação para simulação SDN e o controlador POX foi o mais utilizado com 3 trabalhos seguido pelo controlador RYU com 2 trabalhos e os controladores NOX e Opendaylight com apenas 1 trabalho para cada controlador.

De acordo com a análise dos artigos revisados mostrados na Tabela 5, os trabalhos de Sahoo et al. (2018), Trung et al. (2015), Xing et al. (2018), Huang e Yu (2019), Alparslan et al. (2017), Sharma (2017), Zhu e Chang (2019), Saharan e Gupta (2019), Singh, Khan e Agrawal (2015), Hussein et al. (2016), Qin et al. (2019), Bae et al. (2015), Kim et al. (2017) e Ngo, Pham-Quoc e Thinh (2019) apresentam mecanismos de defesa contra ataques DDoS que atuam na terceira fase do ataque: quando o exército de bots (botnet) já está formado e executando o ataque. Nesse estágio, os controladores SDN ou serviços na rede podem já ter sido o alvo do ataque e já podem estar inoperantes. Apenas os trabalhos em Rahman, Quraishi e Lung (2019) e Pande et al. (2018) focaram na detecção de ataques na segunda fase, que consiste em instalar software DDoS quando os dispositivos são infectados e ingressam no botnet.

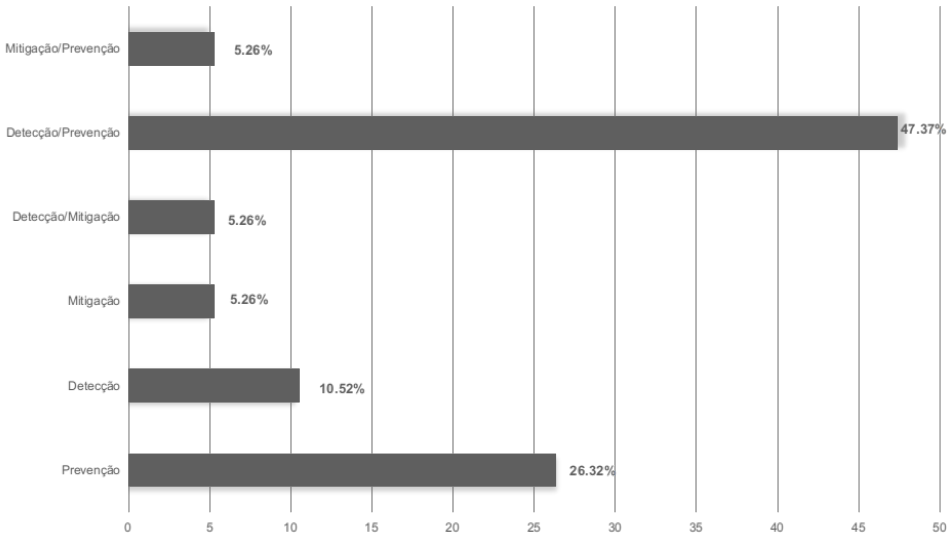
Dentre os artigos que abordaram técnicas de prevenção, o uso de filtros foi teve o maior predomínio, conforme também observado na Tabela 5.

Tabela 5 – Comparação entre os trabalhos selecionados.

Autor/Ano	Método de Defesa	Fases do Ataque	Técnicas de Prevenção	Avaliação
[01] Sahoo et al. (2018)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Filtro	Utilização de algoritmos Machine Learning que podem auxiliar a prever o ataque com precisão.
[02] Trung et al. (2015)	Prevenção	Fase 03 - Disparando o Ataque	Filtro	Aplicação de algoritmo híbrido (dados estatísticos e lógica difusa).
[03] Xing et al. (2018)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Filtro / Balanceamento de Carga	Utilização de um contêiner Docker para a limpeza do tráfego anormal.
[04] Rahman, Quraishi e Lung (2019)	Detecção / Mitigação	Fase 02 - Instalação de Software DDoS		Comparativo de técnicas de aprendizado de máquina para encontrar o mais adequado para prevenção.
[05] Pande et al. (2018)	Detecção / Prevenção	Fase 02 - Instalação de Software DDoS	Filtro	Identificação de host infectados pela estatísticas de fluxo.
[06] Huang e Yu (2019)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Filtro	Criação e distribuição de regras para rede de sensores com a utilização de Machine Learning.
[07] Alparslan et al. (2017)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Sobreposição Segura	Alteração do tráfego para caminhos secundários utilizando funções virtualização de rede.
[08] Sharma (2017)	Mitgação	Fase 03 - Disparando o Ataque		Sistema auto-mitigante que utiliza fluxo de pacotes e um mecanismo baseado em limites.
[09] Zhu e Chang (2019)	Prevenção	Fase 03 - Disparando o Ataque	Filtro	Criação de uma identificação criptografada e assinadas por chaves privadas dos pacotes.
[10] Saharan e Gupta (2019)	Mitigação / Prevenção	Fase 03 - Disparando o Ataque	Filtro	Utilização de SDN para impedir ataques DDoS baseados em DNS.
[11] Singh, Khan e Agrawal (2015)	Prevenção	Fase 03 - Disparando o Ataque	Filtro	Identificação e armazenamento dos pacotes em um buffer para posteriormente bloquear o IP atacante.
[12] Hussein et al. (2016)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Sobreposição Segura	Criação do plano de segurança na arquitetura SDN.
[13] Qin et al. (2019)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Filtro	Utilização de matriz de combinações para o posicionamento de um controlador backup.
[14] Bae et al. (2015)	Detecção	Fase 03 - Disparando o Ataque	Filtro	Identificação de host na rede local SDN durante o ataque.
[15] Kim et al. (2017)	Detecção	Fase 03 - Disparando o Ataque	Filtro	Armazenamento de históricos de consultas DNS para distinguir entre pacotes normais e de ataques.
[16] Ngo, Pham-Quoc e Thinh (2019)	Prevenção	Fase 03 - Disparando o Ataque	Filtro	Utilização de computação paralela para a criação de um switch OpenFlow.
[17] Ahuja e Singal (2019)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Filtro	Cálculo da taxa de pacotes e largura de banda.
[18] Lee, Chang e Syu (2020)	Detecção / Prevenção	Fase 03 - Disparando o Ataque	Filtro	Identificar ataque através do cabeçalho e comprimento dos pacotes.
[19] Bose et al. (2019)	Prevenção	Fase 03 - Disparando o Ataque	Filtro	Utilização de *Blockchain* para atualizar a tabela *OpenFlow* do controlador.

A classificação do método de defesa utilizada nos trabalhos selecionados possibilitou a construção do gráfico da Figura 8, do qual se pode concluir que o uso conjunto de técnicas de detecção e prevenção são as mais utilizadas para a defesa contra ataques DDoS.

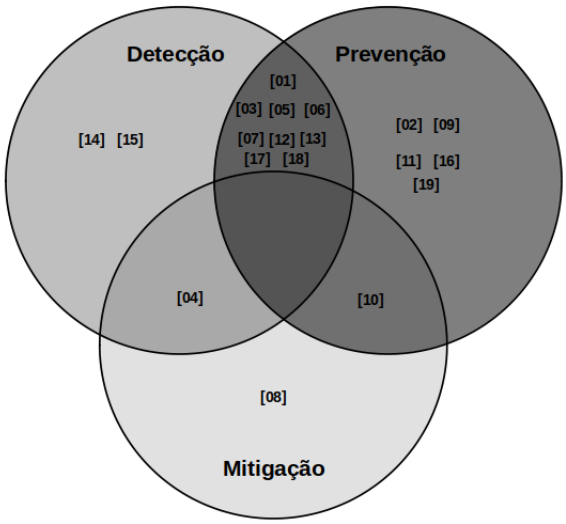
Figura 8 – Porcentagem de cada técnicas de defesa entre artigos selecionados.



Fonte: Autor.

Nove estudos entre os selecionados tiveram abordagem de detecção e prevenção, representando 43,75% dos estudos. Na Figura 9 é possível verificar esses nove artigos nos círculos de intersecção de detecção e prevenção.

Figura 9 – Técnicas de defesa contra ataques DDoS.



Fonte: Autor.

O trabalho em Sahoo et al. (2018), apresenta um mecanismo de detecção que é utilizado para prevenir o ataque DDoS, utilizando diversos algoritmos de aprendizagem de máquina e regressão linear. Xing et al. (2018), propõem um sistema de detecção e prevenção de intrusão (IDPS), onde *Docker* com *Snort* foi usado para limpar o tráfego anormal. O mecanismo (PANDE et al., 2018) identifica *hosts* e atacantes na rede, monitorando o tráfego por meio do controlador, que solicita estatísticas de fluxo e verifica a tabela ARP para encontrar alterações nos pacotes. O *Fuzzy Guarde xtension* apresentado por Huang e Yu (2019) tem como objetivo prevenir ataques DDoS em redes de sensores sem fio definidas por software. A arquitetura proposta por Alparslan et al. (2017), demonstrou uma melhora na taxa de absorção do tráfego DDoS com o uso de NFV e SDN, alterando os caminhos de serviço para rotas secundárias. Além dos planos de controle de dados e SDN, o trabalho em Hussein et al. (2016), propõem a criação de um plano de segurança para detectar IPs e pacotes falsificados. Com a utilização de um controlador de backup e um algoritmo para geração de uma mistura de combinações. O esquema de Qin et al. (2019), visam reduzir os custos de planejamento da rede e fornecer serviço interrompido ao SDN.

O segundo maior grupo de artigos são aqueles que usa apenas prevenção como técnica de defesa contra os ataques DDoS. Os trabalhos de Trung et al. (2015), Zhu e Chang (2019), Singh, Khan e Agrawal (2015), Bose et al. (2019) e Ngo, Pham-Quoc e Thinh (2019), pertencem a esse grupo. Conforme ilustrado na Figura 9; ressaltam-se também a falta de artigos que abordem as três técnicas de defesa simultaneamente. Os resultados desse mapeamento destacam a necessidade de um maior enfoque na prevenção nos estágios iniciais do ataque DDoS (ou seja, Fase 1 e Fase 2). O panorama da literatura também destaca a importância da fusão de diferentes técnicas, sem sobrecarregar os recursos dos dispositivos da rede e, ao mesmo tempo, acompanhar a evolução dos métodos de ataque. A predominância de estudos de simulação é outro aspecto importante, que pode ser explicado pelos altos custos para aquisição de infraestrutura para experimentos SDN. Outro aspecto importante a ser considerado é a adaptação fácil e rápida de ambientes SDN simulados para avaliar configurações distintas é outro aspecto importante a ser considerado.

Diante disto, este trabalho de pesquisa apresenta um mecanismo de prevenção de ataques DDoS em redes SDN com foco em atuar nas lacunas encontradas na revisão sistemática. Atuando na primeira fase do ataque e utilizando as técnicas de prevenção: filtro e *honeypot*.

3.4 Considerações Finais

Foi apresentado um mapeamento sistemático da literatura sobre a prevenção de ataques DDoS em SDN. Essa análise, foi realizada com artigos encontrados em bancos de dados relevantes de publicações da ciência da computação. Os dados de 19 artigos, foram coletados e analisados após o uso cuidadoso do protocolo de seleção pré-definido. Os resultados mostram a escassez de trabalhos propondo técnicas ou mecanismos de prevenção que atuem nas primeiras fases do ataque DDoS em redes SDN.

O objetivo deste mapeamento foi compreender e analisar como os pesquisadores estão abordando a segurança em redes SDN em relação aos ataques DDoS. Ataques estes que continuaram a crescer nos últimos anos. Sabemos que nesta arquitetura os dispositivos de rede são controlados por um controlador SDN, onde é possível monitorar, gerenciar e configurar estes dispositivos através da execução de software no controlador. Assim, o controlador passa a ser um ponto vulnerável na rede e com um ataque bem-sucedido afetará toda a rede.

O mapeamento concluiu que a segurança das redes definidas por *software* ainda precisa de melhorias em relação aos ataques DDoS, principalmente quando o ataque visar o controlador SDN.

4

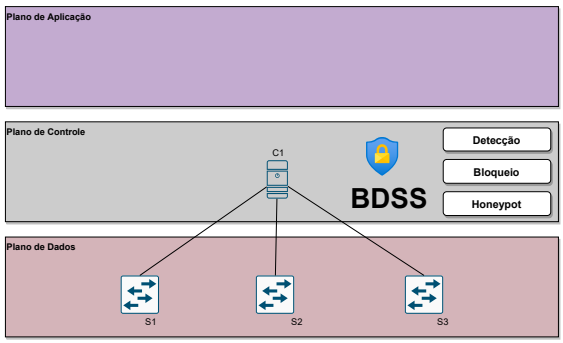
BDSS -Blocking DDoS Scan on SDN

De acordo com Rahman, Quraishi e Lung (2019), o ataque DDoS revelou ter um efeito devastador em uma SDN, quando o ataque for realizado contra o controlador. Sendo assim, o objetivo do mecanismo de prevenção será de proteger o controlador SDN dos próprios *hosts* infectados dentro da SDN.

Assis et al. (2018), destacam que os ataques DDoS são frequentemente precedidos por ataques de varredura de porta, no qual o invasor varre as portas do servidor para encontrar uma abertura para um processo de intrusão.

A fim de contribuir para a segurança das redes definidas por *software*, o presente trabalho propõe o mecanismo BDSS - *Blocking DDoS Scan on SDN* definido em camadas, conforme a sua arquitetura é ilustrada na Figura 10.

Figura 10 – Arquitetura SDN com mecanismo BDSS.

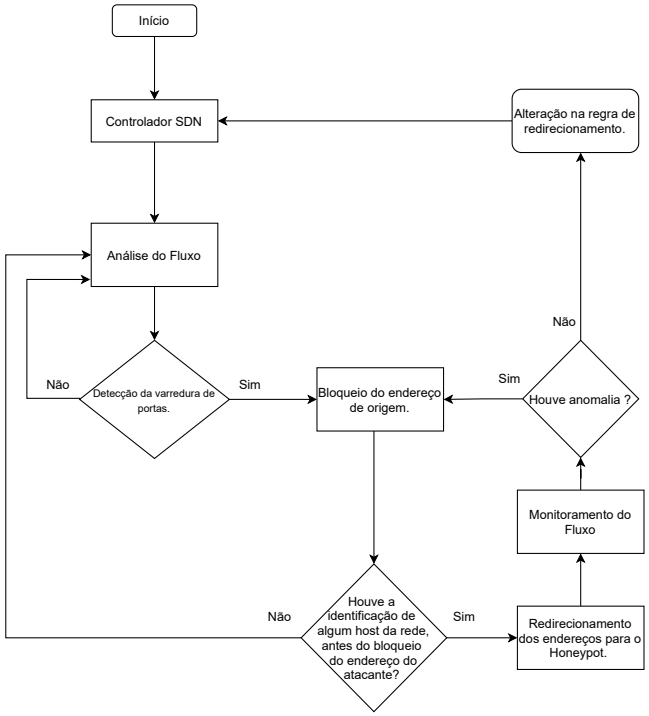


Fonte: Autor.

As camadas são definidas nas subseções: 4.1 camada de detecção; 4.2 camada de bloqueio e a 4.3 camada *honeypot*.

O mecanismo BDSS tem como objetivo de prevenir que a SDN seja alvo de um ataque DDoS. Para atingir esse objetivo, será utilizada uma análise de fluxo de IP, para a caracterizar o comportamento anormal da rede e posteriormente, detectar e bloquear o ataque DDoS na sua fase inicial, conforme representado na Figura 11.

Figura 11 – Fluxograma do mecanismo BDSS.



Fonte: Autor.

4.1 Camada de Detecção

Nesta camada é realizada a detecção da varredura de portas por algum host da SDN, para identificar este ataque, foram coletados dados dos fluxos *OpenFlow*, utilizando as características do fluxo de dados baseadas no trabalho de (ASSIS et al., 2018). Os diferentes recursos de fluxo de IP são: bits/s, pacotes/s, endereços IP de origem/destino, portas de origem/destino e *flag* TCP.

Segundo Neu et al. (2018), o intervalo de coleta de estatísticas é um ponto crítico a ser considerado. No entanto, se a coleta dos dados for realizada em um intervalo de tempo maior, haverá um atraso na detecção do ataque. Por isso, optou-se neste trabalho por um intervalo de três segundos para a coleta dos dados, levando em consideração os testes realizados no trabalho de (NEU et al., 2018).

Os códigos 1 e 2 foram utilizados para identificar a varredura de portas. O mecanismo solicita ao *swicth* a cada três segundos a estatística dos últimos cinco minutos do tráfego da rede.

Sendo assim, verificado se a quantidade de fluxos originados de um único *host* é menor ou igual a cinco, pelo protocolo TCP possuindo o *Flags* SYN, SYN / ACK e ACK *(three-way handshake)* e o protocolo UDP com o flag igual a *null*.

Código 1 – Detecção de varreduras na rede.

```
1 if (isCandidateFlow(dFlowStats)) { // Ate 5 fluxos e porta diferente
     de 0.
2        // Protocolo TCP com flags SYN, ACK e SYN/ACK. + Protocolo UDP
            com flag null
3             if (isToAnalyzeTCP(dFlowStats) ||
                 isToAnalyzeUDP(dFlowStats)) {
4               newRow(dmatch, key.getValue().toString());
5             }
6           }
```

Código 2 – Parâmetros do mecanismo.

```
1  // Ativacao do mecanismo.
2  module.active=true
3  // Estatisticas dos 5 ultimos minutos do trafego da rede.
4  analysis_time_in_minutes=5
5  // Numero de eventos para a varredura vertical .
6  vscan.amount_of_events=3
7  // Numero de eventos para a varredura horizontal.
8  hscan.amount_of_events=3
9  // Tempo de monitoramento do trafego a cada 3 segundos.
10 schedule.monitor_statistics=3
11 // Tempo de monitoramento para a identificacao da varredura.
12 schedule.monitor_port_scan_control=3
13 // Qauntidade de fluxos.
14 analysis.qtd_pkt_flow=5
15 // Flags elegiveis para serem analisados como varredura de rede.
16 analysis.flags_tcp=2,16,18
```

4.2 Camada de Bloqueio

A camada possibilita o bloqueio do endereço IP do atacante (*host* infectado), incluindo uma regra no controlador SDN descartando os fluxos provenientes do atacante. Além disso, o mecanismo analisa se houve alguma conexão bem-sucedida no intervalo entre a detecção e bloqueio do atacante. Portanto, o tráfego destes *hosts* serão redirecionados ao servidor *Honeypot*.

Para realização do bloqueio dos *hosts* que realizaram a varredura na rede foi desenvolvido o Código 3, o qual ativa o *firewall* nativo do controlador *foodlight* e realiza o bloqueio do endereço IP.

Código 3 – Bloqueio dos *hosts*.

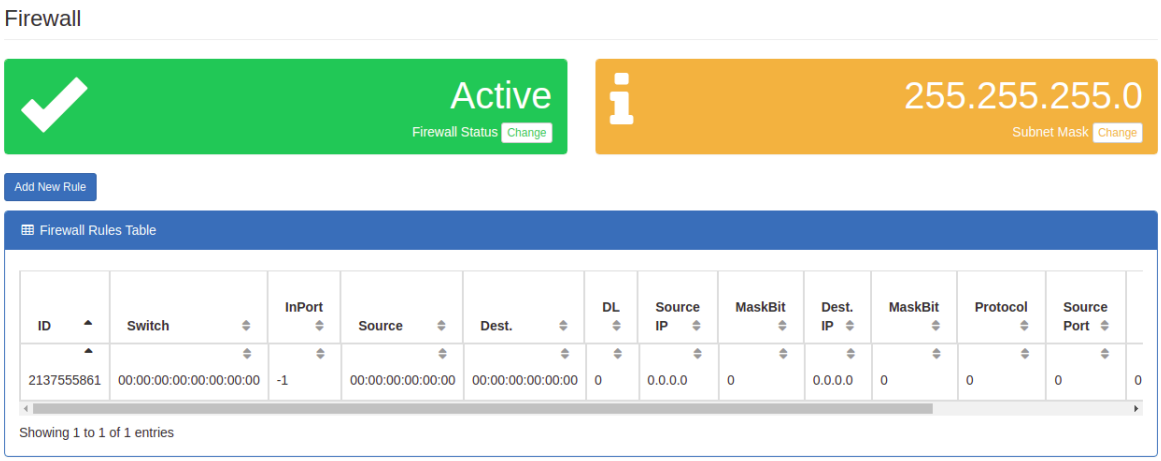
```
// Habilita o Firewall do Controlador
// Por padrao, o firewall nega todo o trafego, a menos que uma regra
   ALLOW (regra_zero) seja criada.
if(!firewallService.isEnabled()) {
  FirewallRule regra_zero = new FirewallRule();
  regra_zero.ruleid = 000000001;
  regra_zero.priority = 1;
  regra_zero.action = FirewallAction.ALLOW;
  irewallService.enableFirewall(true);
  firewallService.addRule(regra_zero);
}
/*Criacao da regra de bloqueio*/
BigInteger bigInt = new BigInteger(ipSrc.getBytes());
ACLRule rule = new ACLRule();
rule.setId(bigInt.intValue());
rule.setNw_src(ipSrc + "/32");
rule.setNw_src_prefix(IPv4.toIPv4Address(ipSrc));
rule.setNw_src_maskbits(32);
rule.setAction(rule.getAction().DENY);

List<ACLRule> regras = aclService.getRules();
boolean existeRegra = false;

for (Iterator iterator = regras.iterator(); iterator.hasNext();) {
  ACLRule aclRule = (ACLRule) iterator.next();
    if(aclRule.getId() == rule.getId()) {
      existeRegra = true;
    }
}
if(!existeRegra) {
    aclService.addRule(rule);
    log.info("Regra adicionada com sucesso!");
}
```

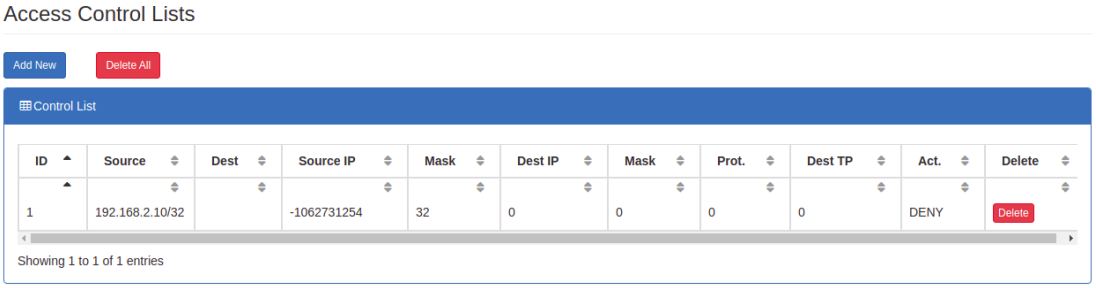
Na criação da ACL (*Access Control List*) de bloqueio é necessário a ativação do *firewall* do *foodlight*, sendo necessária a criação de uma regra para habilitar todo o tráfego na rede, visto que sempre que ativado é criada uma regra padrão a qual bloqueia todo o tráfego da rede, ilustrada pela Figura 12. O bloqueio é realizado com a criação de uma ACL com o IP do *host* que efetuou a varredura da rede, conforme a Figura 13.

Figura 12 – Firewall Floodlight.



Fonte: Autor.

Figura 13 – Lista de controle de acesso Floodlight.



Fonte: Autor.

4.3 Camada *Honeypot*

Segundo Gangadhara, Hasyagar e Damotharan (2019), *Honeypots* são aplicativos de segurança de redes avançados. Ressaltam, que devido a SDN ser baseada na camada de plano de controle, ela garante que a comunicação permaneça transparente redirecionando a comunicação entre o *honeypot* para futuras investigações. Visto que, nas redes tradicionais os mecanismos de transferência de conexão TCP são vulneráveis aos ataques (como varreduras de portas, que podem ser facilmente visível para os *hackers*). Nesta camada, os endereços serão investigados com a finalidade de encontrar alguma anomalia na comunicação destes *hosts*.

De acordo com Du e Wang (2020), os honeypots possui algumas vantagens na capacidade de proteção e de utilização de recursos. Por exemplo, como um IDS pode ter que monitorar um grande número de atividades de rede com trilhões de bytes por segundo, seu *cache* se esgotará

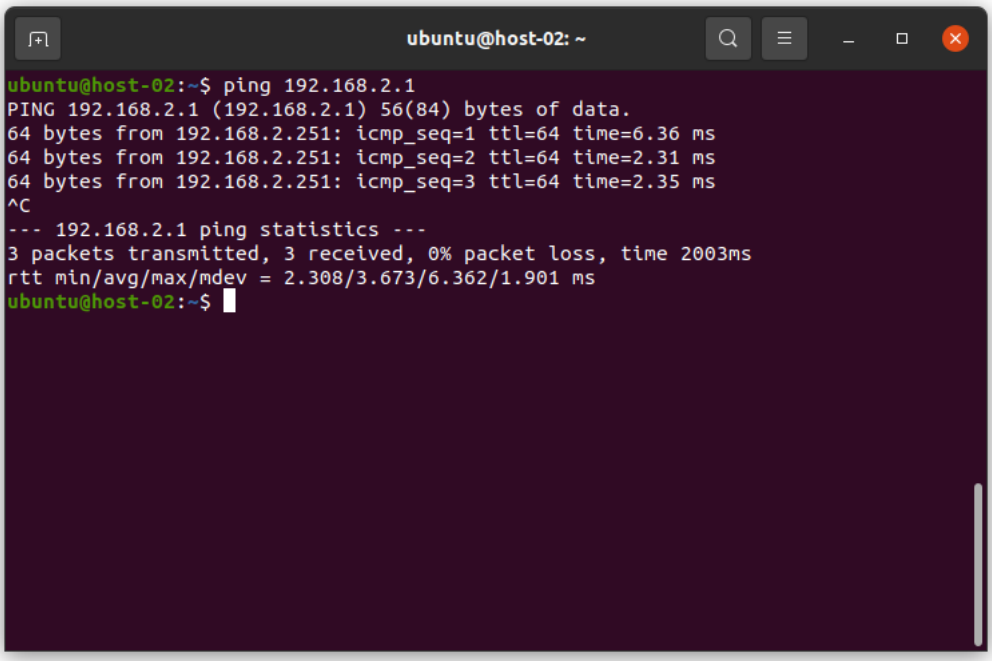
rapidamente. Ao contrário, os honeypots capturam e monitoram apenas uma pequena parte das atividades da rede, sem o problema de esgotamento de recursos.

O intervalo entre a detecção e o bloqueio do atacante poderia ter fornecido informações sobre alguns *hosts* da rede. Sendo que, estes deverão ter algumas ações monitoradas como exemplo: tentativas de conexões SSH, conexão remota do *Windows*, tráfego FTP, tentativas de encontrar novos *hosts* na rede utilizando o *ping*, etc.

O mecanismo realiza a identificação e redirecionamento dos possíveis *hosts* infectados para o servidor *honeypot*. É realizada uma verificação dos *hosts* da rede, os quais retornaram o *flag TCP RST+ACK*. Com o objetivo de proteger o controlador de um possível ataque, todos os fluxos originados destes *hosts* com destino ao controlador serão redirecionados para o servidor *honeypot*.

A Figura 14 apresenta o teste da primeira etapa do redirecionamento, com um *ping* do *host* a ser investigado para o controlador, onde fica claro o redirecionamento dos fluxos. Nesta etapa o mecanismo cria a primeira regra, conforme Figura 15.

Figura 14 – Exemplo da primeira etapa do redirecionamento.



Fonte: Autor.

Figura 15 – Regra da primeira etapa do redirecionamento.

Static Flow Pusher
(00:00:00:e0:4c:36:00:1d)

Add New | Delete All

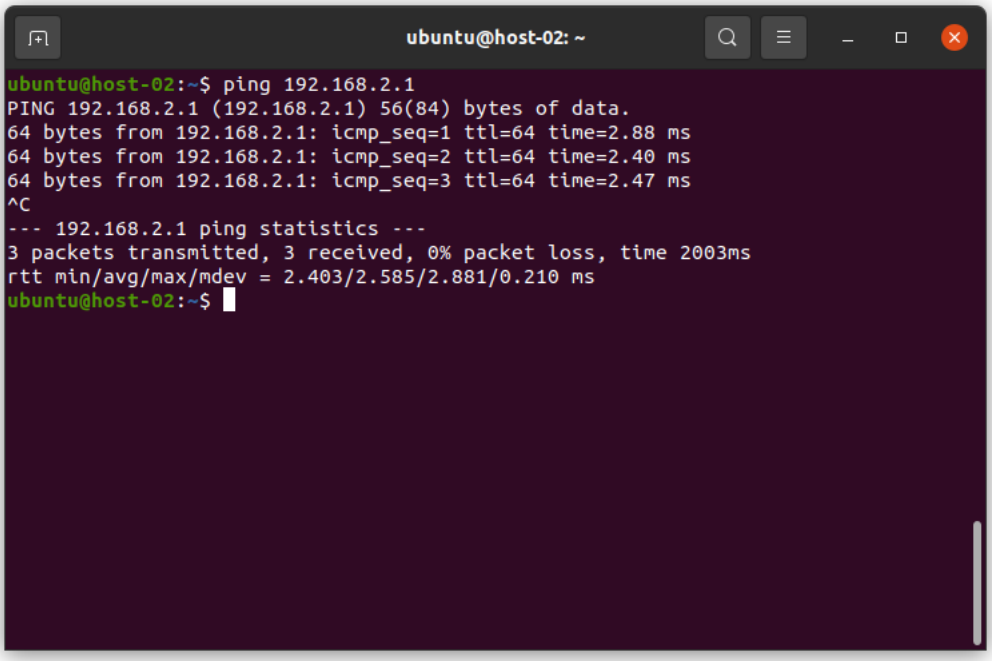
Flows

Name	Command	Cookie	Priority	idleTimeoutSec	hardTimeoutSec	Out Port	flags	In port	Command	Delete
IDA-HONEYPOT	ADD	49539595486004096	32768	0	0	any	1	1	{"instruction_apply_actions":{"actions":"set_field=eth_dst->08:00:27:6d:d9:63,set_field=ipv4_dst->192.168.2.251,output=2"}}	Delete

Fonte: Autor.

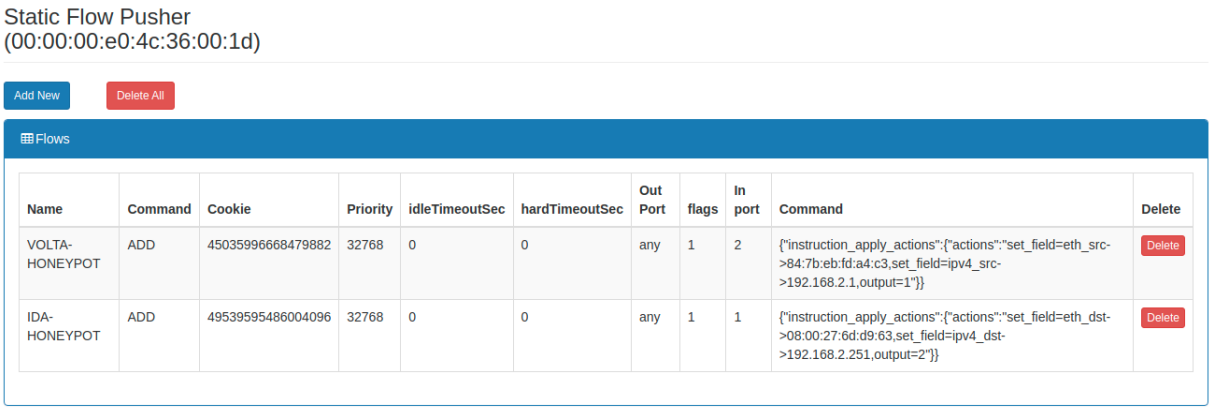
Com a criação da segunda regra do redirecionamento concluída o procedimento torna-se totalmente transparente para o usuário, conforme ilustrado na Figura 16. A Figura 17 apresenta esta regra implementada no controlador.

Figura 16 – Exemplo com a segunda etapa do redirecionamento.



Fonte: Autor.

Figura 17 – Regra da segunda etapa do redirecionamento.



Fonte: Autor.

5
Estudo de Caso

Este capítulo apresenta o estudo de caso que demonstram a implementação prática do mecanismo BDSS. O estudo de caso exibe a prevenção de um ataque DDoS em sua fase inicial, momento em que é realizada a varredura na rede. A prevenção tem por objetivo buscar detectar a varredura na rede, bloquear o endereço IP do ataque e redirecionar possíveis endereços identificados pela varredura para um servidor *honeypot*.

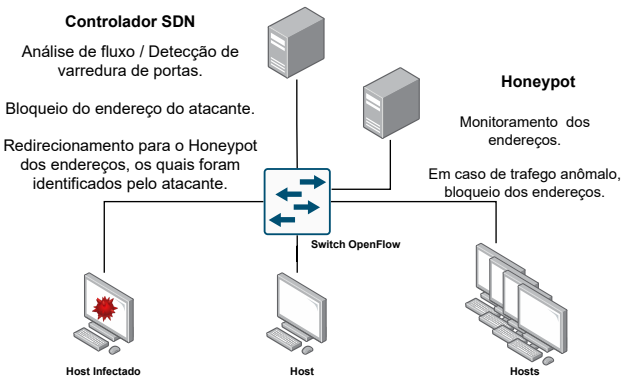
5.1 Metodologia Experimental

Nesta seção são apresentados as tecnologias adotadas à implementação prática do mecanismo de prevenção de ataques DDoS em redes SDN, incluindo uma descrição do ambiente utilizado para implementação do mecanismo.

5.1.1 Ambiente de Implementação

A primeira etapa para criação do cenário foi a montagem do ambiente para a realização do experimento, conforme apresentado na Figura 18. A seguir serão apresentados os componentes de *hardware* e *software* utilizados neste ambiente. A Figura 19 apresenta a prototipagem utilizada no experimento.

Figura 18 – Topologia de rede do mecanismo BDSS.



Fonte: Autor.

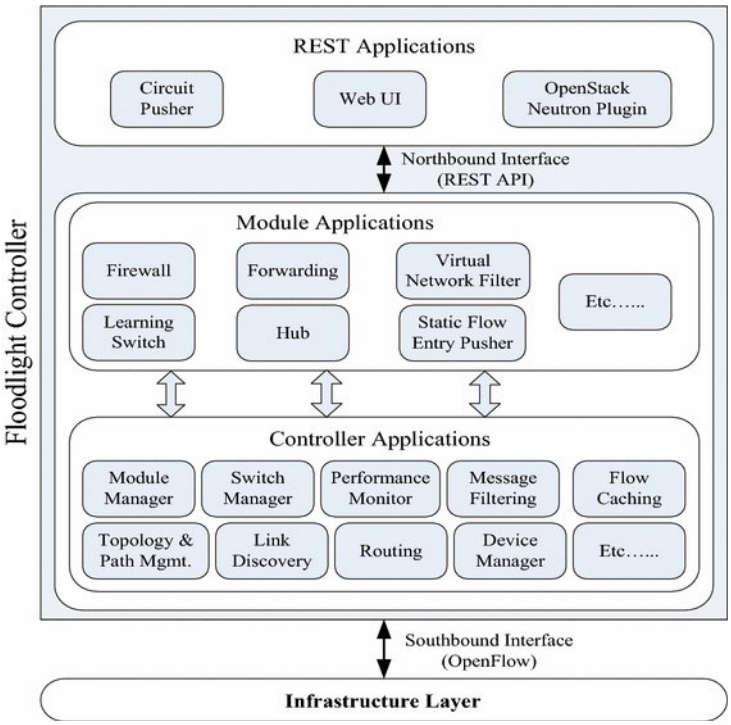
Figura 19 – Prototipagem do experimento.

Fonte: Autor.

5.1.2 Controlador SDN

Neste trabalho foi utilizado o *OpenFlow Floodlight*, é um controlador SDN de código aberto baseado na linguagem *Java*. O *Floodlight* possui algumas características relevantes: oferece um sistema modular que simplifica a extensão e o aprimoramento, facilita na configuração com dependências mínimas e possui suporte a uma ampla variedade de *switches* físicos e virtuais habilitados para o *OpenFlow*. Além disso, o *Floodlight* possui um módulo de aplicativos os quais estão disponíveis na parte superior do controlador, conforme sua arquitetura é apresentada na Figura 20 (BHOLEBAWA; DALAL, 2018).

Figura 20 – Arquitetura interna do controlador do *Floodlight* e suas interfaces.



Fonte: (BHOLEBAWA; DALAL, 2018).

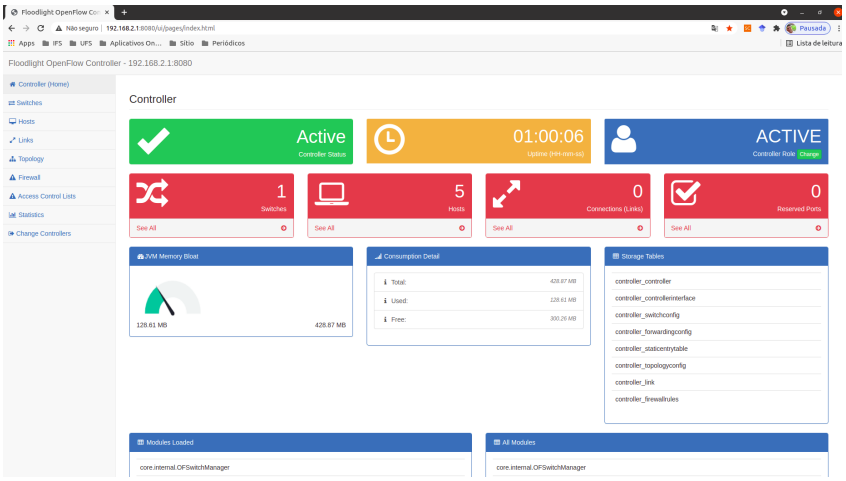
A instalação do controlador *OpenFlow Floodlight* foi realizada em um computador com processador Intel(R) Core(TM) i5-7200U CPU @ 2.50Ghz, 8GB de memória RAM, SSD de 254GB e com o sistema operacional Ubuntu 20.04.2 LTS - 64Bits. Foi realizado a instalação do *OpenJDK 8* [1] e do *Eclipse* [2].

O controlador *OpenFlow Floodlight*, possui uma interface GUI (Java *Application*) baseada na web é acessada pelo navegador *web* utilizando o protocolo HTTP, sendo utilizado o endereço da porta 8080. Portanto, neste experimento, o endereço de acesso a interface WEB ficou: http://192.168.2.1:8080/ui/pages/index.html, conforme apresentado na Figura 21.

[1] É uma implementação open source do Java que provê um Java Development Kit totalmente baseado em software livre.

[2] É uma IDE para desenvolvimento Java.

Figura 21 – Interface GUI do controlador *Floodlight*.



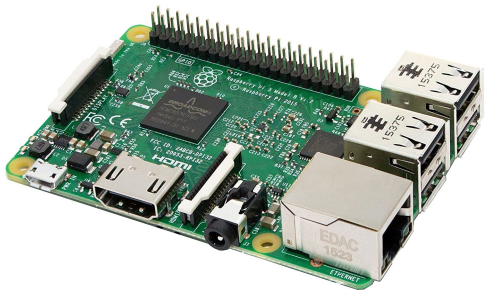
Fonte: Autor.

5.1.3 *Hosts*

Como *hosts* da SDN foi utilizado o modelo Raspberry Pi 3 Model B Rev 1.2 64Bits, com o processador Quad Core 1.2GHz Broadcom BCM2837 64bit CPU e 1GB RAM, como ilustrado na Figura 22.

Essas placas foram interligadas à rede do experimento, por meio de conexão *Fast Ethernet*. Em todas as placas, estava instalado o sistema operacional Ubuntu 20.04.1 LTS, kernel Linux 5.4.0-1041-raspi, arquitetura arm64 e os *softwares* necessários para emulação do tráfego de rede (*scapy*) e para varredura da rede (*Nmap*).

Figura 22 – Raspberry Pi 3 Model B



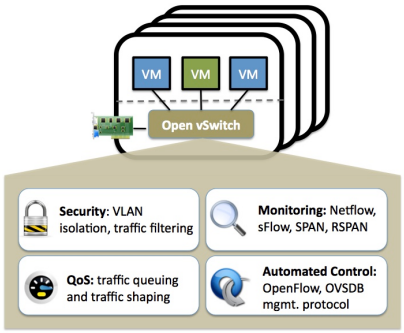
Fonte: Autor.

5.1.4 *Software Switch OpenFlow*

O protocolo *OpenFlow* continua sendo o principal utilizado pela indústria, apresentado diversas ferramentas *Openflow*, como *software switch*, controlador e plugin do *wireshark* [3] (FERNANDES; ROTHENBERG, 2014). Permitindo que o controlador receba e envie mensagens para o *Switch OpenFlow*, como também envie configurações para os dispositivos da rede.

O *Open vSwitch* (OvS) é um *software* de código aberto o qual implementa um *switch* virtual de multicamadas. Possui suporte a encaminhamento de mensagens *OpenFlow* e integração com vários sistemas de gerenciamento virtual. Sua arquitetura é apresentada na Figura 23.

Figura 23 – Arquitetura *Open vSwitch*



Fonte: https://www.openvswitch.org/

Devido ao custo muito elevado do *switch OpenFlow* será utilizado neste experimento o minicomputador Raspberry Pi 3, com o sistema operacional Ubuntu 18.04.4 LTS e *Open vSwitch* 2.9.8. Para a conexão com os demais *hosts* foram utilizados quatro adaptadores de rede USB-RJ45, ilustrado na Figura 24.

Figura 24 – Adaptador USB-RJ45

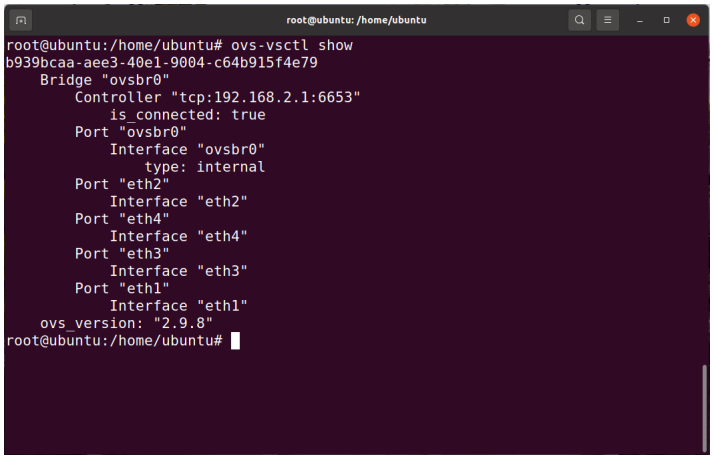
Fonte: https://www.multilaser.com.br/

A Figura 25 apresenta o comando ovs-vsctl é um utilitário para consultar e configurar o *Open vSwitch*. Com a utilização do parâmetro *show* é possível ter uma visão geral das portas e

[3] O Wireshark é um programa que analisa o tráfego de rede, e o organiza por protocolos.

interfaces, onde cada porta (*eth1, eth2, eth3 e eth4*) são representados pelos adaptadores de rede USB-RJ45.

Figura 25 – Lista de portas e *bridges*



Fonte: Autor.

Com a finalidade de permitir o acesso a Internet dos *hosts* do exeprimento, foi criado um código que realiza o roteamento entre a rede *wireless* e a rede cabeada, conforme apresentado no Código 4. Sendo assim, o *gateway* da rede do experimento é definido com endereço IP configurado na interface LAN.

Código 4 – Roteamento para os hosts.

```
1  #!/bin/bash
2  echo "Reiniciando Interfaces"
3  #/etc/init.d/networking restart
4  echo "Habilitando o Roteamento do Kernel"
5  echo 1 > /proc/sys/net/ipv4/ip_forward
6  echo "Limpando as Regras Iptables"
7  iptables -F
8  echo "Habilitando o Roteamento Iptables"
9  iptables -P FORWARD ACCEPT
10 echo "Habilitando o Roteamento da Wlan"
11 iptables -t nat -A POSTROUTING -o wlp2s0 -j MASQUERADE
```

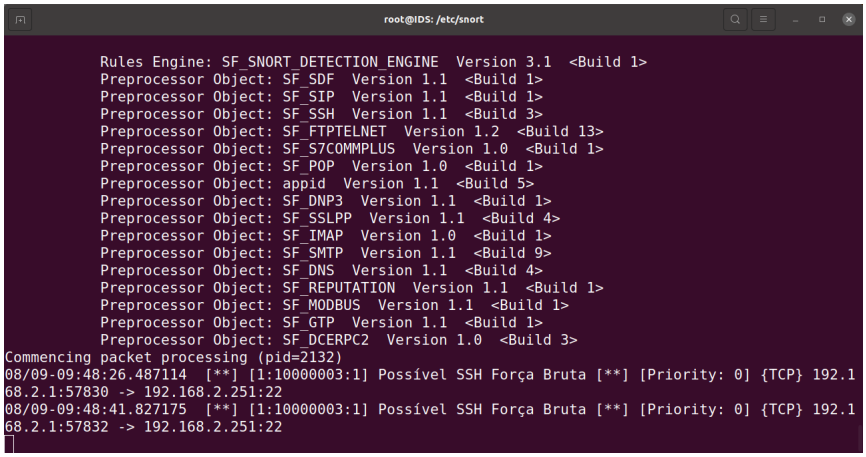
5.1.5 Honeypot

Neste experimento foi instalado e configurado o *Snort* [4] em um computador com processador Intel Core i7-2630QM, 4GB de memória RAM, Hd de 1TB e com o sistema operacional Ubuntu 20.04.2 LTS. Na Figura 26 podemos ver os *logs* do Snort em ação com um

[4] O Snort é um Sistema de Prevenção de Intrusão (IPS) de código aberto.

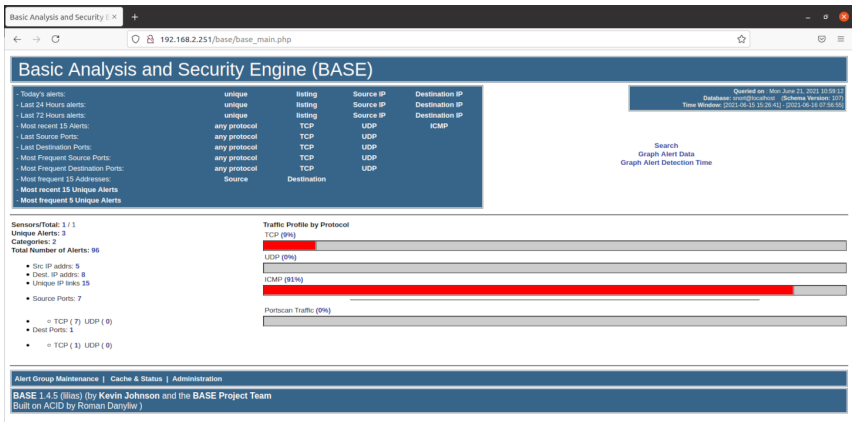
exemplo de um possível ataque de força bruta SSH. Neste computador também foi instalado o BASE – *(Basic Analysis and Security Engine)* apresentado na Figura 27, para fornecer um *front-end* da web para consultar e analisar os alertas provenientes do *Snort*.

Figura 26 – Alerta de ataque força bruta SSH - Snort



Fonte: Autor.

Figura 27 – BASE – *(Basic Analysis and Security Engine)*



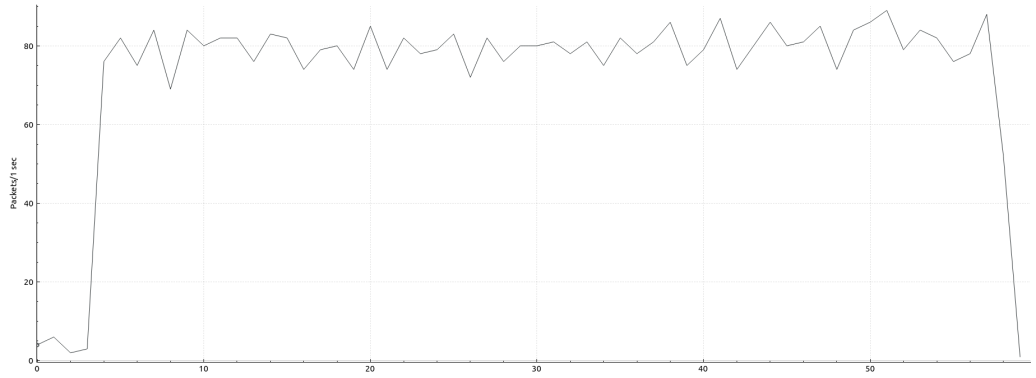
Fonte: Autor.

5.2 Experimento

Neste trabalho foi analisado a fase inicial do ataque DDoS em uma rede SDN, realizando uma varredura de portas e vulnerabilidades, com o objetivo de obter acesso privilegiado as máquinas da rede e possibilitando a criação de uma lista de endereços IP's das máquinas invadidas, que posteriormente serão utilizadas no ataque.

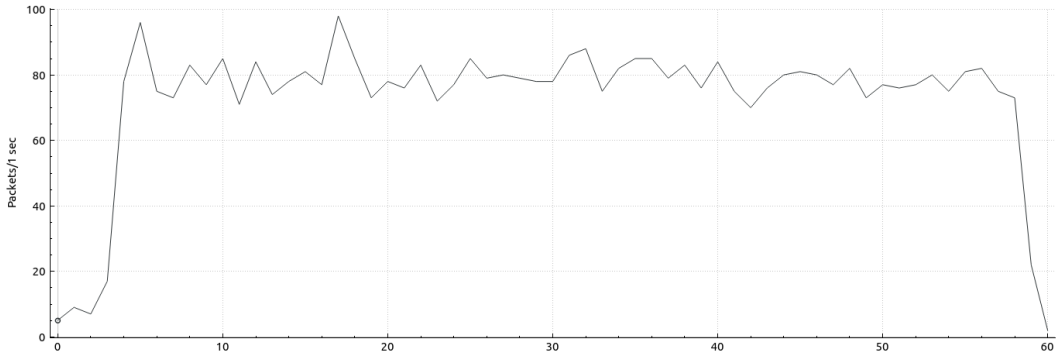
Na realização deste experimento foi utilizada a ferramenta *Scapy*[5] para a geração do tráfego normal, usando diferentes protocolos, endereços IP's e portas. Na realização da varredura horizontal e vertical foi utilizada a ferramenta *Nmap*[6]. As Figura 28, Figura 29 e Figura 30 apontam o tráfego normal, tráfego com a varredura vertical e o tráfego com a varredura horizontal.

Figura 28 – Tráfego normal.



Fonte: Autor.

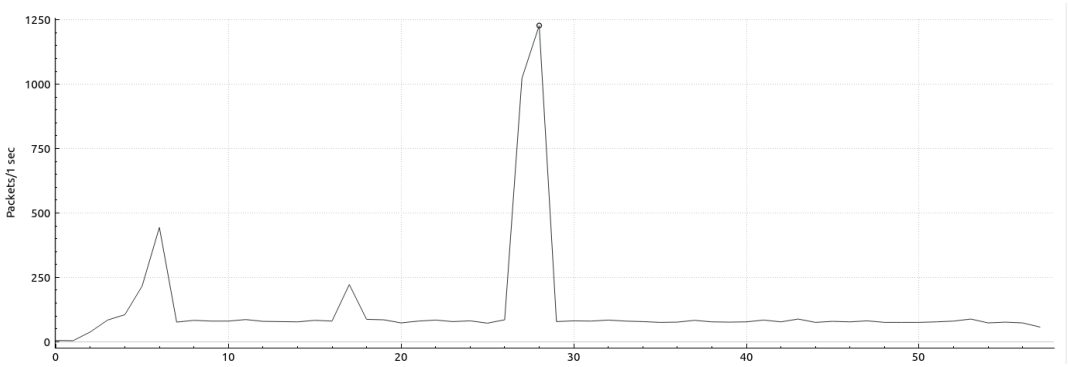
Figura 29 – Tráfego com a varredura vertical.



Fonte: Autor.

[5] É uma ferramenta de manipulação de pacotes para redes de computadores, escrita em Python.
[6] É um software livre que realiza port scan.

Figura 30 – Tráfego com a varredura horizontal.



Fonte: Autor.

5.2.1 Cenários

No experimento do estudo de caso, foram realizados dois cenários, conforme Tabela 6. Sendo que, os dois utilizam a mesma topologia da rede apresentada na seção 5.1.1 e na Tabela 7 e com a mesma situação hipotética, na qual um dos *hosts* da rede foi infectado e está executando a varredura na rede. No cenário I é observado a varredura horizontal, sendo identificados 5 *hosts*, com a porta 22/tcp e 3 com a porta 80/tcp aberta, demonstrado na Figura 31 e no cenário II a varredura vertical foi realizada nas 20 principais portas de apenas um *host*, identificando apenas as portas 22/tcp e 80/tcp abertas, como demonstrado na Figura 32.

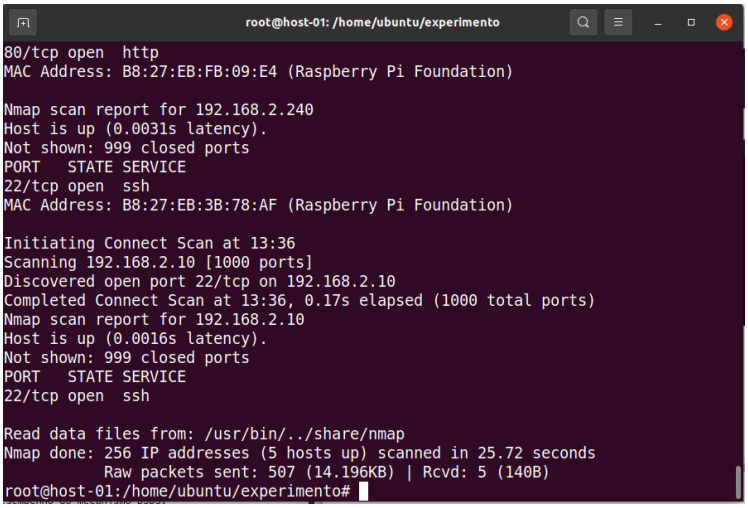
Tabela 6 – Cenários do estudo de caso

CENÁRIO	**DESCRIÇÃO**
I	Utiliza a mesma topologia de redes apresentada na Figura 18; O HOST-01 é o atacante e executa a varredura horizontal; Comando utilizado para geração do ataque: nmap -v -sT 192.168.2.0/24
II	Utiliza a mesma topologia de redes apresentada na Figura 18; O HOST-01 é o atacante e executa a varredura vertical; Comando utilizado para geração do ataque: nmap -v -sS –top-ports 20 192.168.2.11

Tabela 7 – Tabela de endereçamento IP

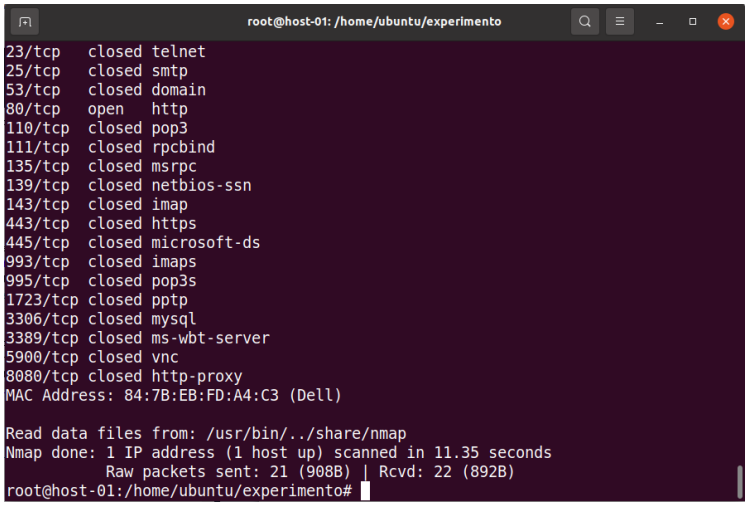
HOSTS	ENDEREÇO IP	DESCRIÇÃO
Host-01	192.168.2.10/24	Atacante
Host-02	192.168.2.11/24	Geração do tráfego normal.
Host-03	192.168.2.240/24	Host da rede.
Host-04	192.168.2.251/24	IDS - Snort.
Controlador SDN	192.168.2.1/24	Floodligth
Comutador	192.168.2.5/24	OpenVswitch

Figura 31 – Varredura horizontal.



Fonte: Autor.

Figura 32 – Varredura vertical.



Fonte: Autor.

Neu et al. (2018), destacam que a varredura de portas consiste no envio de vários tipos de pacotes para saber mais sobre um *host* ou rede de destino. Categoriza a varredura em: vertical quando um *host* de origem varre várias portas distintas do mesmo *host* de destino; e horizontal quando um *host* de origem varre uma porta específica em diferentes *hosts* de destino.

5.3 Metodologia de Avaliação

Nesta seção são apresentadas as métricas utilizadas, as quais possibilitaram avaliar o desempenho do mecanismo BSDS.

5.3.1 Métricas de Desempenho

As métricas de desempenho são os indicadores utilizados na avaliação dos resultados do experimento. Para escolhas das métricas, nos baseamos no trabalho de (RIBEIRO; SANTOS; NASCIMENTO, 2021).

Assim, nesta dissertação foram utilizadas para avaliar o mecanismo de prevenção de ataques DDoS em redes SDN as seguintes métricas: (1) Total de Instâncias; (2) Número de Falso Positivo; (3) Número de Verdadeiro Negativos; (4) Número de Verdadeiro Positivos; (5) Número de Falso Negativo; (6) Acurácia; (7) *Recall*; (8) Precisão e; (9) *F1-Score*. Está métricas são apresentadas na Tabela 8 e foram utilizadas em todos os cenários.

Tabela 8 – Descrição das métricas de desempenho.

Métrica	**Descrição**
TI - Total de Instâncias	Total do dados simulados dos tráfegos legítimos e maliciosos.
FP - Falso Positivo	Tráfego legítimo identificado como malicioso.
VN - Verdadeiro Negativo	Tráfego legitimo identificado como legítimo.
VP - Verdadeiro Positivo	Tráfego malicioso identificado como malicioso.
FN - Falso Negativo	Tráfego malicioso identificado como legítimo.
Acurácia = (VP+VN)/(VN+VP+FN+FP)	Combinação de acerto de detecção, considerando todos os verdadeiros positivos e negativos em relação a população total.
Recall = VP/(VP+FN)	Percentual do tráfego malicioso identificado como malicioso.
Precisão = VP/(VP+FP)	Avalia a capacidade de sucesso na abordagem.
F1-Score = (*recall**2*precisão) / (*recall*+precisão)	Média harmônica entre precisão e *recall*.

5.3.2 Resultados

Para a realização dos testes foram gerados dois tipos de tráfegos de rede: o tráfego normal e o tráfego com a varredura de rede. O tráfego normal foi gerado a partir de uma coleta de dados da rede da Universidade Del Cauca, Popayán, Colômbia. As coletas dos pacotes foram realizadas em diferentes horários durante o período da manhã e da tarde, em diferentes dias entre abril e junho de 2019 (KAGGLE, 2019).

A geração do tráfego normal foi por meio de um script em *Python* utilizando a ferramenta *Scapy*, onde foi definido os 4 endereços IP's dos *hosts* utilizados no experimento como endereço

de origem e para os campos endereço de destino, porta de destino, tamanho do pacote e a duração do pacote foi utilizado os campos do dataset (KAGGLE, 2019).

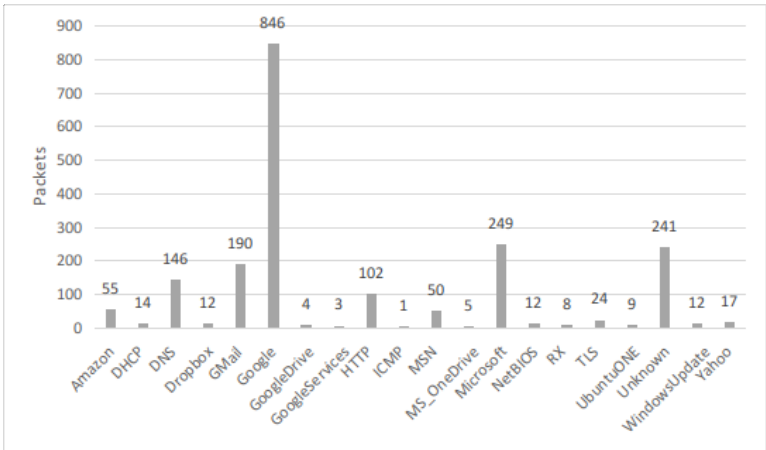
Para este experimento foram retiradas 4 amostras aleatórias, possuindo 2000 linhas do arquivo CSV deste *dataset*. Sendo assim, possível a simulação do tráfego em diferentes dias e horários, conforme apresentado na Tabela 9.

Tabela 9 – Quantidade de pacotes enviados por amostra.

	Amostra - 01	**Amostra - 02**	**Amostra - 03**	**Amostra - 04**
TCP	19971	16743	117942	51115
UDP	3357	3442	2329	4651
ICMP	1	24	0	24
Tempo de Execução	597s	896s	2501s	2249s

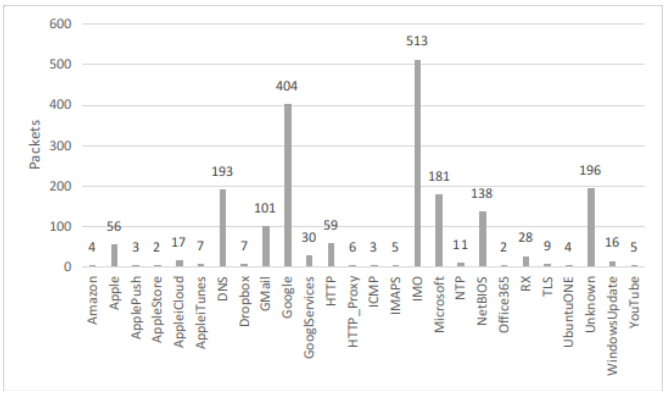
Nas Figuras 33, 34, 35 e 36 são apresentados os serviços que foram utilizados nas amostras, demonstrando a heterogeneidade nos pacotes trafegados na rede do experimento.

Figura 33 – Serviços utilizados na amostra 01.



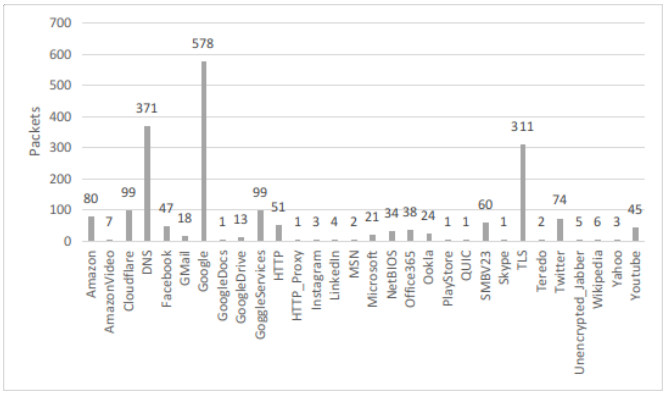
Fonte: Autor.

Figura 34 – Serviços utilizados na amostra 02.



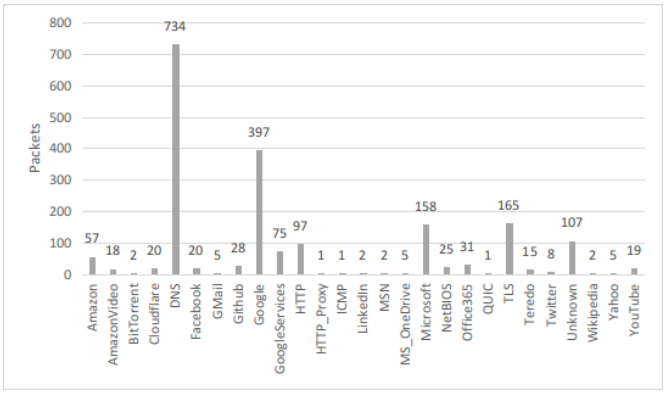
Fonte: Autor.

Figura 35 – Serviços utilizados na amostra 03.



Fonte: Autor.

Figura 36 – Serviços utilizados na amostra 04.



Fonte: Autor.

Para validar o mecanismo, foi gerado o tráfego normal da rede e simultaneamente realizado o ataque. Para cada amostra foram realizados dois experimentos, o primeiro utilizando a varredura horizontal e o segundo a varredura vertical. As Tabelas 10, 11, 12 e 13 apresentam os resultados obtidos após a realização dos testes nas diferentes amostras nos dois cenários.

Tabela 10 – Valores obtidos com as varreduras na amostra 01.

Descrição	**Varredura Vertical**	**Varredura Horizontal**
Total de Instâncias	588	1842
Número de Falso Positivo	8	0
Número de Verdadeiro Negativos	558	950
Número de Verdadeiro Positivos	22	892
Número de Falso Negativo	0	0

Tabela 11 – Valores obtidos com as varreduras na amostra 02.

Descrição	**Varredura Vertical**	**Varredura Horizontal**
Total de Instâncias	799	2151
Número de Falso Positivo	5	4
Número de Verdadeiro Negativos	770	1113
Número de Verdadeiro Positivos	24	1034
Número de Falso Negativo	0	0

Tabela 12 – Valores obtidos com as varreduras na amostra 03.

Descrição	**Varredura Vertical**	**Varredura Horizontal**
Total de Instâncias	1491	1874
Número de Falso Positivo	13	5
Número de Verdadeiro Negativos	1455	955
Número de Verdadeiro Positivos	23	914
Número de Falso Negativo	0	0

Tabela 13 – Valores obtidos com as varreduras na amostra 04.

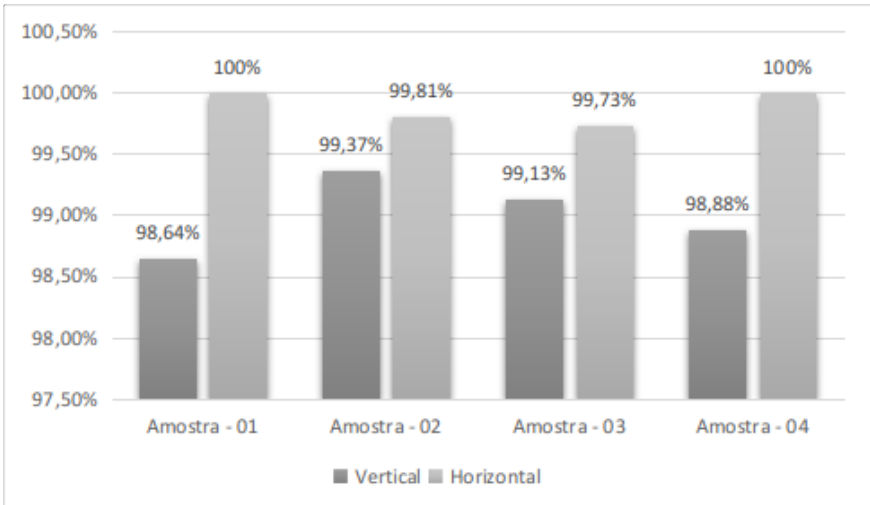
Descrição	**Varredura Vertical**	**Varredura Horizontal**
Total de Instâncias	1076	1979
Número de Falso Positivo	12	0
Número de Verdadeiro Negativos	1034	1027
Número de Verdadeiro Positivos	30	952
Número de Falso Negativo	0	0

A diferença entre o total de instâncias entre os cenários, se dá pelo fato que na varredura vertical o número de pacotes enviados é menor, por se tratar de uma varredura somente de um *host*, em contrapartida na varredura horizontal é realizada em vários *hosts*.

A ausência de falso negativo está relacionada, ao bloqueio do *hosts* infectado após o término da varredura, impossibilitando outro ataque. Portanto, todos os fluxos maliciosos gerados pelos ataques são definidos como verdadeiros positivos.

Na Figura 37 é apresentada os valores obtidos em relação acurácia das amostras, sendo a indicação geral do desempenho do mecanismo. Na varredura horizontal a métrica variou entre 99,73% e 100%. Nas amostra A e B obteve-se 100%, pois não houve nenhum falso positivo nas amostras. Na varredura vertical houve uma variação de 98,64% e 99,37%.

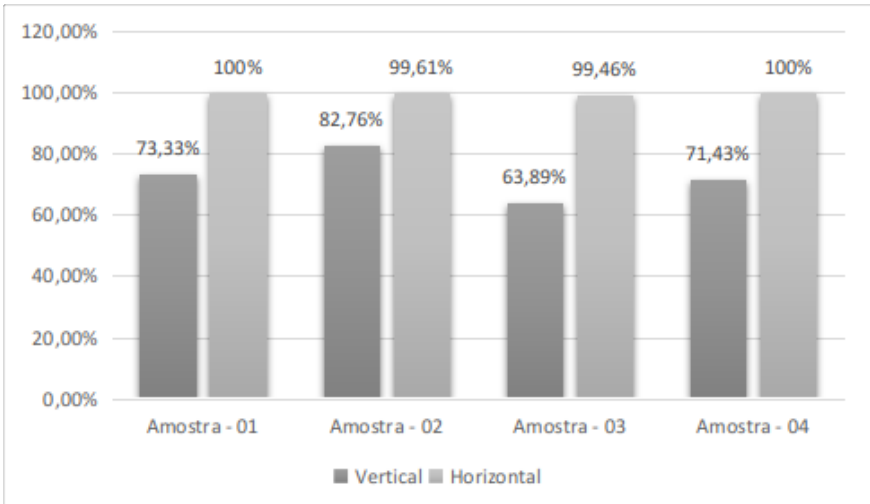
Figura 37 – Acurácia.



Fonte: Autor.

A métrica precisão da varredura vertical em relação a varredura vertical apresentou-se menor, devido aos números de falsos positivos encontrados na varredura horizontal, conforme apresentando na Figura 38.

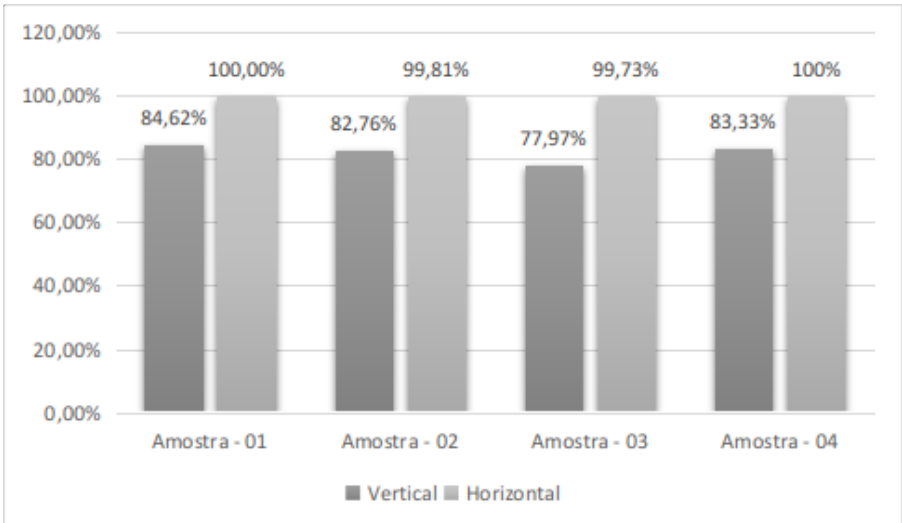
Figura 38 – Precisão.



Fonte: Autor.

Foi utilizada a métrica *F1-Score* para aferir a qualidade do experimento, sendo uma forma de combinar a precisão e o *recall*. Em todas as amostras foi obtido o valor de 100% de *recall*. Como apresentado na Figura 39, a varredura horizontal teve melhor desempenho em relação a varredura vertical. Os valores altos são um indicativo de que a precisão ou o *recall* está alto.

Figura 39 – *F1 Score*.



Fonte: Autor.

O valor obtido na precisão no cenário I é pode ser explicado pela limitação de *hosts* utilizados no experimento. O programa utilizado para a geração de tráfego normal realiza o sorteio de 4 *hots* da rede, sendo que pode acontecer de um desses *hosts* serem sorteados sucessivamente. Portanto se enquadrando nas características de bloqueio e consequentemente gerando um falso positivo.

No trabalho desenvolvido por Neu et al. (2018), foram realizados testes para a detecção de falsos positivos no *script* gerador de tráfego normal da rede. Sendo que, neste trabalho foi utilizados dados reais para a geração do tráfego normal da rede.

Conforme apresentado no Capítulo 3, não houveram trabalhos que atuem na primeira fase do ataque DDoS. Dentre os trabalhos que atuam na terceira fase e utilizam aprendizado de máquina para a detecção do ataque DDoS, o mecanismo desenvolvido por Sahoo et al. (2018), utilizam os algoritmos *Random Forest* e o *Linear Regression*, os quais obtiveram a precisão de 98,65% e 98,40% respectivamente. A arquitetura apresentada por Ngo, Pham-Quoc e Thinh (2019) possui três mecanismos de detecção DDoS, utiliza placas FPGA e com o modelo de rede neural implementada atinge 99,01% de precisão. O mecanismo BDSS atua na primeira fase do ataque DDoS, é implementado diretamente no controlador permitindo uma rápida resposta ao ataque. Além disso, por não utilizar algoritmos de aprendizado de máquina na detecção o mecanismo não requer muito processamento do controlador.

Apesar dos trabalhos de Sahoo et al. (2018) e Ngo, Pham-Quoc e Thinh (2019), possuírem uma abordagem diferente na detecção e na fase do ataque DDoS. O mecanismo BDSS foi mais eficiente, obtendo a média de 99,76% de precisão na detecção da varredura horizontal.

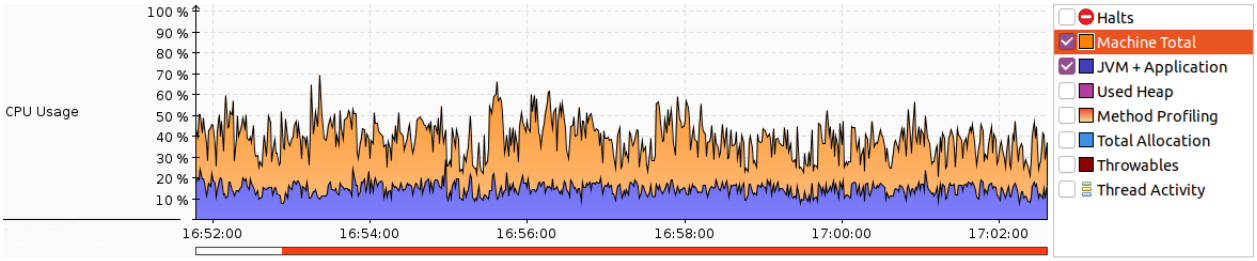
Um olhar mais atento em nossos conjuntos de dados revelou que o mecanismo obteve sucesso em todas as suas abordagens em um tempo satisfatório, realizando a detecção e bloqueio de todas as varreduras nos dois cenários.

Na varredura horizontal foi possível detectar que, os *hosts* identificados na varredura enviam um *flag* TCP com o valor 20 (RST+ACK). Portanto, uma característica importante para a realização do redirecionamento destes possíveis *hosts* infectados para um monitoramento em um servidor *honeypot*.

O tráfego malicioso pode ser caracterizado por outros fatores além das varreduras de rede. A ocorrência de um número elevado de requisições DNS em um pequeno intervalo de tempo, pode ocasionar muitos falsos positivos. Tornando-se um inconveniente no decorrer do tempo.

A Figura 40 apresenta a utilização de CPU durante 10 minutos, foram realizadas 5 varreduras horizontais e verticais para verificar a taxa de utilização da CPU, a qual não houve nenhuma alteração significativa durante o monitoramento. O gráfico de utilização de CPU nos dá indícios de que o mecanismo proposto não impacta de forma significativa o desempenho do controlador SDN.

Figura 40 – Utilização da CPU.



Fonte: Autor.

6
Conclusão

Este trabalho teve como objetivo desenvolver um mecanismo de prevenção de ataques DDoS com alvo aos controladores de uma rede SDN. Para alcançar o objetivo o mecanismo deve identificou e bloqueou as varreduras de rede.

Após a realização de um mapeamento sistemático, foram levantados dados que indicaram a ausência de trabalhos que atuem na primeira fase do ataque DDoS. Assim conclui, que as redes SDN possuem vulnerabilidades em relação aos ataques DDoS, apesar do risco evidente de um ataque visando o controlador SDN. A partir deste fato, foi proposto um mecanismo para prevenir ataques DDoS em sua primeira fase em redes SDN. Aproveitando os diferentes recursos de IP, onde foi possível detectar e bloquear *hots* infectados.

Dentre os artigos selecionados 37,5% utilizaram aprendizado de máquina para a detecção do ataque DDoS. O mecanismo implementado neste trabalho, demonstrou-se mais leve por ser executado diretamente no controlador. Portanto, consumindo menos recursos computacionais.

Com esse mecanismo, foi possível detectar e bloquear os dois tipos de varreduras, que são usados para encontrar vulnerabilidades na infraestrutura SDN. O mecanismo foi implementado diretamente na camada de controle, o que permitiu uma resposta mais rápida e eficiente na detecção e bloqueio do tráfego malicioso.

Resultados experimentais demonstram que o mecanismo pode ajudar a prevenir ataques com precisão. De acordo com o resultado da avaliação, o mecanismo é capaz de detectar e bloquear uma varredura vertical com cerca de 72,85% de precisão e na varredura horizontal essa taxa foi de 99,76%.

A principal contribuição deste trabalho foi: a criação de um mecanismo de prevenção de ataques DDoS em redes SDN, o qual demonstrou-se ser eficiente em um ambiente experimental, sendo assim, podendo ser utilizado em uma rede real.

Por fim, o desenvolvimento do módulo de detecção de varreduras de rede diretamente no

controlador *Floodlight*, permitiu a criação de ACL's para o bloqueio dos *hosts* infectados, como também o redirecionamento de possíveis *hosts* infectados.

6.1 Dificuldades e Limitações

No decorrer deste trabalho, algumas dificuldades e limitações foram encontradas. Devido à pandemia da Covid 19, a Universidade Federal de Sergipe e o Instituto Federal de Sergipe aderiram ao *home office*, impossibilitando a captura do tráfego de rede para a utilização no ambiente experimental.

Devido a falta de dados do tráfego de redes, foi realizada uma pesquisa para encontrar uma forma de geração de tráfego que se aproximava-se mais de uma rede real, não sendo possível a realização dos testes e consequentemente, proporcionou atrasos em relação ao cronograma desta dissertação.

Devido a limitação de somente quatro entradas USB do Raspberry Pi 3 Model B, o experimento ficou restrito a cinco *hots* com a utilização de 4 adaptadores USB-RJ45 e a porta *ethernet* do dispositivo. Além disso, o programa gerador de tráfego utilizou apenas 4 endereços válidos na rede, o que pode ter gerado falsos negativos.

6.2 Trabalhos Futuros

Nesta seção são levantadas algumas possibilidades de trabalhos futuros para aprimorar os resultados obtidos:

- Criação automática de ACL's no controlador SDN, dos eventos maliciosos identificados por um servidor *honeypot*;
- Realização de testes em ambientes SDN reais de empresas e outras organizações, para obter resultados mais precisos com a realidade de uma rede real;
- Utilização de algoritmos de aprendizado de máquinas para a detecção de varreduras de redes.

6.3 Publicações Relacionadas à Dissertação

Nesta seção são apresentadas as contribuições relacionadas a dissertação.

6.3.1 Artigo Aprovados

Systematic Mapping on Prevention of DDoS Attacks on Software Defined Networks - In 15th Annual IEEE International Systems Conference (IEEE SYSCON) April 15-May 15, 2021 - Qualis A4.

6.3.2 Artigo Submetido

Network Scan Detection in SDNs for prevention of DDoS Botnet Formation - Computing Journal - Qualis A2

Referências

AHUJA, N.; SINGAL, G. Ddos attack detection & prevention in sdn using openflow statistics. In: IEEE. *2019 IEEE 9th International Conference on Advanced Computing (IACC)*. [S.l.], 2019. p. 147–152. Citado 4 vezes nas páginas 23, 41, 44 e 46.

ALPARSLAN, O. et al. Improving resiliency against ddos attacks by sdn and multipath orchestration of vnf services. In: IEEE. *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. [S.l.], 2017. p. 1–3. Citado 6 vezes nas páginas 22, 40, 44, 45, 46 e 48.

ASSIS, M. V. D. et al. Fast defense system against attacks in software defined networks. *IEEE Access*, IEEE, v. 6, p. 69620–69639, 2018. Citado 2 vezes nas páginas 50 e 51.

BADOTRA, S.; PANDA, S. Software-defined networking: A novel approach to networks. In: *Handbook of Computer Networks and Cyber Security*. [S.l.]: Springer, 2020. p. 313–339. Citado na página 23.

BAE, H.-B. et al. Zombie pc detection and treatment model on software-defined network. In: *Computer Science and its Applications*. [S.l.]: Springer, 2015. p. 837–843. Citado 4 vezes nas páginas 37, 44, 45 e 46.

BHOLEBAWA, I. Z.; DALAL, U. D. Performance analysis of sdn/openflow controllers: Pox versus floodlight. *Wireless Personal Communications*, Springer, v. 98, n. 2, p. 1679–1699, 2018. Citado 2 vezes nas páginas 59 e 60.

BOSE, A. et al. Blockchain as a service for software defined networks: A denial of service attack perspective. In: IEEE. *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. [S.l.], 2019. p. 901–906. Citado 4 vezes nas páginas 37, 44, 46 e 48.

Choi, Y. et al. Integrated ddos attack defense infrastructure for effective attack prevention. In: *2010 2nd International Conference on Information Technology Convergence and Services*. [S.l.: s.n.], 2010. p. 1–6. Citado 2 vezes nas páginas 19 e 29.

DU, M.; WANG, K. An sdn-enabled pseudo-honeypot strategy for distributed denial of service attacks in industrial internet of things. *IEEE Transactions on Industrial Informatics*, v. 16, n. 1, p. 648–657, 2020. Citado na página 54.

FERNANDES, E. L.; ROTHENBERG, C. E. Openflow 1.3 software switch. *Salao de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos SBRC*, p. 1021–1028, 2014. Citado na página 62.

GANGADHARA, S.; HASYAGAR, S. N.; DAMOTHARAN, U. Deployable sdn architecture for network applications: an investigative survey. In: IEEE. *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. [S.l.], 2019. p. 43–49. Citado 2 vezes nas páginas 31 e 54.

HUANG, M.; YU, B. Fuzzyguard: A ddos attack prevention extension in software-defined wireless sensor networks. *KSII Transactions on Internet & Information Systems*, v. 13, n. 7, 2019. Citado 7 vezes nas páginas 18, 40, 42, 44, 45, 46 e 48.

HUSSEIN, A. et al. Sdn security plane: An architecture for resilient security services. In: IEEE. *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*. [S.l.], 2016. p. 54–59. Citado 8 vezes nas páginas 22, 26, 32, 41, 44, 45, 46 e 48.

KAGGLE. *https://www.kaggle.com/jsrojas/labeled-network-traffic-flows-114-applications*. 2019. Citado 2 vezes nas páginas 68 e 69.

KIM, S. et al. Preventing dns amplification attacks using the history of dns queries with sdn. In: SPRINGER. *European Symposium on Research in Computer Security*. [S.l.], 2017. p. 135–152. Citado 4 vezes nas páginas 38, 44, 45 e 46.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, Ieee, v. 103, n. 1, p. 14–76, 2014. Citado na página 26.

KUŹNIAR, M. et al. Methodology, measurement and analysis of flow table update characteristics in hardware openflow switches. *Computer Networks*, Elsevier, v. 136, p. 22–36, 2018. Citado na página 26.

LEE, T.-H.; CHANG, L.-H.; SYU, C.-W. Deep learning enabled intrusion detection and prevention system over sdn networks. In: IEEE. *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. [S.l.], 2020. p. 1–6. Citado 4 vezes nas páginas 23, 42, 44 e 46.

LI, W.; MENG, W.; KWOK, L. F. A survey on openflow-based software defined networks: Security challenges and countermeasures. *Journal of Network and Computer Applications*, Elsevier, v. 68, p. 126–139, 2016. Citado 3 vezes nas páginas 24, 25 e 32.

MAHJABIN, T. et al. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, SAGE Publications Sage UK: London, England, v. 13, n. 12, p. 1550147717741463, 2017. Citado 4 vezes nas páginas 28, 29, 30 e 31.

NEU, C. V. et al. Lightweight ips for port scan in openflow sdn networks. In: IEEE. *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. [S.l.], 2018. p. 1–6. Citado 3 vezes nas páginas 51, 68 e 73.

NGO, D.-M.; PHAM-QUOC, C.; THINH, T. N. Heterogeneous hardware-based network intrusion detection system with multiple approaches for sdn. *Mobile Networks and Applications*, Springer, p. 1–15, 2019. Citado 8 vezes nas páginas 37, 43, 44, 45, 46, 48, 73 e 74.

PANDE, B. et al. Detection and mitigation of ddos in sdn. In: IEEE. *2018 Eleventh International Conference on Contemporary Computing (IC3)*. [S.l.], 2018. p. 1–3. Citado 6 vezes nas páginas 18, 40, 44, 45, 46 e 48.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. [S.l.: s.n.], 2008. p. 1–10. Citado na página 33.

QIN, C. C. et al. Attack-aware recovery controller-switch-link cost minimization placement algorithm in software-defined networking. In: SPRINGER. *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. [S.l.], 2019. p. 297–308. Citado 5 vezes nas páginas 41, 44, 45, 46 e 48.

Rahman, M. M.; Roy, S.; Yousuf, M. A. Ddos mitigation and intrusion prevention in content delivery networks using distributed virtual honeypots. In: *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. [S.l.: s.n.], 2019. p. 1–6. Citado na página 31.

RAHMAN, O.; QURAISHI, M. A. G.; LUNG, C.-H. Ddos attacks detection and mitigation in sdn using machine learning. In: IEEE. *2019 IEEE World Congress on Services (SERVICES)*. [S.l.], 2019. v. 2642, p. 184–189. Citado 11 vezes nas páginas 18, 20, 22, 27, 32, 39, 42, 44, 45, 46 e 50.

RIBEIRO, A. d. R. L.; SANTOS, R. Y. C.; NASCIMENTO, A. C. A. Anomaly detection technique for intrusion detection in sdn environment using continuous data stream machine learning algorithms. In: IEEE. *2021 IEEE International Systems Conference (SysCon)*. [S.l.], 2021. p. 1–7. Citado na página 68.

SAHARAN, S.; GUPTA, V. Prevention and mitigation of dns based ddos attacks in sdn environment. In: IEEE. *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*. [S.l.], 2019. p. 571–573. Citado 6 vezes nas páginas 19, 24, 42, 44, 45 e 46.

SAHOO, K. S. et al. A machine learning approach for predicting ddos traffic in software defined networks. In: IEEE. *2018 International Conference on Information Technology (ICIT)*. [S.l.], 2018. p. 199–203. Citado 10 vezes nas páginas 18, 22, 39, 42, 44, 45, 46, 48, 73 e 74.

Sembiring, I. Implementation of honeypot to detect and prevent distributed denial of service attack. In: *2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. [S.l.: s.n.], 2016. p. 345–350. Citado na página 31.

SHAGHAGHI, A. et al. Software-defined network (sdn) data plane security: Issues, solutions, and future directions. In: *Handbook of Computer Networks and Cyber Security*. [S.l.]: Springer, 2020. p. 341–387. Citado 2 vezes nas páginas 23 e 25.

SHARMA, V. Multi-agent based intrusion prevention and mitigation architecture for software defined networks. In: IEEE. *2017 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.], 2017. p. 686–692. Citado 6 vezes nas páginas 32, 38, 42, 44, 45 e 46.

SINGH, S.; KHAN, R.; AGRAWAL, A. Prevention mechanism for infrastructure based denial-of-service attack over software defined network. In: IEEE. *International Conference on Computing, Communication & Automation*. [S.l.], 2015. p. 348–353. Citado 6 vezes nas páginas 22, 37, 44, 45, 46 e 48.

TRUNG, P. V. et al. A multi-criteria-based ddos-attack prevention solution using software defined networking. In: IEEE. *2015 International Conference on Advanced Technologies for Communications (ATC)*. [S.l.], 2015. p. 308–313. Citado 8 vezes nas páginas 22, 23, 36, 42, 44, 45, 46 e 48.

UBALE, T.; JAIN, A. K. Survey on ddos attack techniques and solutions in software-defined network. In: *Handbook of Computer Networks and Cyber Security*. [S.l.]: Springer, 2020. p. 389–419. Citado na página 27.

XING, J. et al. Asidps: Auto-scaling intrusion detection and prevention system for cloud. In: IEEE. *2018 25th International Conference on Telecommunications (ICT)*. [S.l.], 2018. p. 207–212. Citado 5 vezes nas páginas 39, 44, 45, 46 e 48.

YAN, Q. et al. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE communications surveys & tutorials*, IEEE, v. 18, n. 1, p. 602–622, 2015. Citado 2 vezes nas páginas 18 e 22.

ZHU, X.-w.; CHANG, C.-w. Packet access control mechanism based on cipher identification in software-defined network. In: *Proceedings of the 2019 2nd International Conference on Information Management and Management Sciences*. [S.l.: s.n.], 2019. p. 90–95. Citado 6 vezes nas páginas 22, 36, 44, 45, 46 e 48.

Apêndices

APÊNDICE A – *Script* Geração de Tráfego Normal.

Código 5 – *Script* geração de tráfego

```
from scapy.all import *
from random import randint
import ipaddress
import csv
import pandas as pd

def ip_origem():
  j = randint(0,3)
  if j == 0 :
    return '192.168.2.251'
  elif j == 1:
    return '192.168.2.11'
  elif j == 2:
    return '192.168.2.250'
  elif j ==  3:
    return '192.168.2.240'

dataframe = pd.read_csv('/home/ubuntu/experimento/dataset_01.csv')

for i in range(2000):
  src_ip = dataframe.loc[i,'src_ip']
  dst_ip = dataframe.loc[i,'dst_ip']
  pkt_total = dataframe.loc[i,'pktTotalCount']
  proto = dataframe.loc[i,'proto']
  dst_port = dataframe.loc[i,'dst_port']
  duration = dataframe.loc[i,'min_piat']
  desc = dataframe.loc[i,'web_service']
  if duration > 10:
      duration = 10
  else:
      duration = dataframe.loc[i,'min_piat']
       if proto == 6:
        send(IP(dst=dst_ip,src=ip_origem())/TCP(dport=dst_port)
        /Raw(RandString(pkt_total)), count=pkt_total, inter=duration)
      if proto == 17:
        send(IP(dst=dst_ip,src=ip_origem())/UDP(dport=dst_port)
        /Raw(RandString(pkt_total)), count=pkt_total, inter=duration)
      if proto == 1:
        send(IP(dst=dst_ip,src=ip_origem())/ICMP()
        /Raw(RandString(pkt_total)), count=pkt_total, inter=duration)
```