Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Diego Stelman de Medeiros Gonçalves

Sistema de Proteção contra Ataques de Botnets usando Redes Definidas por Software

Rio de Janeiro
2020

Diego Stelman de Medeiros Gonçalves

Sistema de Proteção contra Ataques de Botnets usando Redes Definidas por Software

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações.

Orientador: Prof. D.Sc. Marcelo Gonçalves Rubinstein
Orientador: Prof. D.Sc. Rodrigo de Souza Couto

Rio de Janeiro
2020

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

G643 Gonçalves, Diego Stelman de Medeiros.
Sistema de proteção contra ataques de botnets usando redes definidas por software / Diego Stelman de Medeiros Gonçalves. – 2020.
63f.

Orientadores: Marcelo Gonçalves Rubinstein, Rodrigo de Souza Couto.
Dissertação (Mestrado) - Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia eletrônica - Teses. 2. Proteção de dados - Teses. 3. Extranets - Teses. 4. Redes de computadores - Teses. 5. Software - Proteção - Teses. I. Rubinstein, Marcelo Gonçalves. II. Couto, Rodrigo de Souza. III. Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia. IV. Título.

CDU 004.056.54

Bibliotecário: Julia Vieira - CRB7/6022

_____ _____

Assinatura Data

Diego Stelman de Medeiros Gonçalves

Sistema de Proteção contra Ataques de Botnets usando Redes Definidas por Software

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações.

Aprovada em 19 de Fevereiro de 2020.
Banca Examinadora:

Prof. D.Sc. Marcelo Gonçalves Rubinstein (Orientador)
DETEL/PEL/UERJ

Prof. D.Sc. Rodrigo de Souza Couto (Orientador)
PEE/COPPE/UFRJ

Prof. D.Sc. Alexandre Sztajnberg
Faculdade de Engenharia - UERJ

Prof. D.Sc. Miguel Elias Mitre Campista
PEE/COPPE/UFRJ

Rio de Janeiro

2020

DEDICATÓRIA

Dedico este trabalho à minha família, aos meus amigos e aos meus orientadores.

AGRADECIMENTOS

Agradeço aos meus pais por sempre me motivarem quando precisei e pela educação que recebi. À minha irmã por sempre estar do meu lado. Aos meus orientadores pelos conhecimentos passados, pela ajuda fornecida e pela dedicação. Agradeço ao Programa de Pós-Graduação em Engenharia Eletrônica da UERJ pela oportunidade que me deram de fazer parte de um ambiente de pesquisa.

A minha caravela? Ela balança mas não para.

Luís de Camões

RESUMO

GONÇALVES, D. S. M. *Sistema de Proteção contra Ataques de Botnets usando Redes Definidas por Software*. 2020. 63 f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2020.

Os ataques de negação de serviço crescem a cada ano exigindo investimentos financeiros e tecnológicos por parte das corporações para evitar danos aos seus serviços prestados na Internet. Em geral, os sistemas de proteção contra esses ataques são implementados através de equipamentos caros que processam um alto volume de tráfego. Além disso, algumas empresas oferecem serviços de tratamento de tráfego malicioso destinado a outros sistemas autônomos na Internet que também são caros. Esta dissertação propõe um sistema de proteção contra ataques de *botnets* do tipo HTTP flood baseado na tecnologia de redes SDN (*Software Defined Networking*) utilizando a colaboração de outros ASs. Esses ASs utilizam redes SDN controladas através de uma VPN pelo sistema de proteção do servidor web alvo dos ataques. Uma outra VPN implementada é utilizada para permitir que os ASs colaboradores enviem requisições diretamente ao servidor web que encontra-se protegido pelo sistema original. As requisições destinadas ao servidor web com o serviço desejado são atendidas pelo sistema e recebem um redirecionamento para o destino real da aplicação protegida. Através da implementação do sistema com SDN, cada requisição terá um fluxo permissivo escrito em um comutador virtual que da acesso ao servidor web. Como as requisições das *botnets* não acessarão o destino real por não seguirem o redirecionamento recebido, apenas requisições de clientes legítimos alcançarão o servidor protegido. Isso permite ao sistema diferenciar IPs atacantes de IPs de clientes legítimos. Dessa forma, os atacantes são bloqueados através de fluxos de bloqueio inseridos no comutador virtual de entrada do sistema. O sistema proposto foi implementado e avaliações de desempenho foram realizadas. Os resultados obtidos mostram reduções gradativas no consumo de CPU do servidor do controlador local, durante um ataque, na medida que ASs colaboradores são adicionados ao sistema. Com seis ASs colaboradores e com o sistema sendo atacado, foram registradas uma queda de consumo de CPU do servidor do controlador local de 65,32%, uma queda de latência percebida pelos clientes de 6 s para aproximadamente 400 ms e uma queda no consumo de CPU do servidor web de 78%.

Palavras-chave: SDN. Ataques. Proteção.

ABSTRACT

GONÇALVES, D. S. M. *Botnet Attack Protection System using Software Defined Networks*. 2020. 63 f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2020.

Denial of service attacks are growing every year requiring financial and technological investments by corporations to prevent damage to their services provided on the Internet. In general, protection systems against these attacks are implemented using expensive equipment that processes a high volume of traffic. In addition, some companies offer malicious traffic handling services to other autonomous systems on the Internet that are also expensive. This dissertation proposes a protection system against HTTP flood botnet attacks based on SDN (Software Defined Networking) network technology using the collaboration of other ASs. These ASs use SDN networks controlled through a VPN by the protection system of the web server targeted by the attacks. Another implemented VPN is used to allow collaborating ASs to send requests directly to the web server that is protected by the original system. The requests destined to the web server with the final service are answered by the system and receive a redirection to the real destination of the protected application. Through the implementation of the system with SDN, each request will have a permissive flow written on a virtual switch that gives access to the web server. Since requests from botnets will not access the actual destination because they do not follow the received redirect, only requests from legitimate clients will reach the protected server. This allows the system to differentiate attacking IPs from legitimate client IPs. In this way, attackers are blocked through blocking flows inserted into the system's virtual input switch. The proposed system was implemented and performance evaluations were carried out. The results obtained show gradual reductions in CPU consumption of the local controller server, during an attack, as collaborating ASs are added to the system. With six collaborating ASs and the system under attack, a drop in CPU consumption of the local controller server of 65.32%, a drop in latency perceived by customers from 6 s to approximately 400 ms and a drop in in 78% web server CPU consumption.

Keywords: SDN. Attacks. Protection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASC	*AS Colaborador*
API	*Application Programming Interface*
AS	*Autonomous System*
CAPTCHA	*Completely Automated Public Turing test to tell Computers and Humans Apart*
DDoS	*Distributed Denial of Service*
IaaS	*Infrastructure as a Service*
HTTP	*Hypertext Transfer Protocol*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
NAT	*Network Address Translation*
OVS	*Open vSwitch*
SDN	*Software-Defined Networking*
SL	*Sistema Local*
TCAM	*Ternary Content Address Memory*
VM	*Virtual Machine*
VIP	*Virtual Internet Protocol*
VPN	*Virtual Private Network*

SUMÁRIO

INTRODUÇÃO

O ataque distribuído por negação de serviço (DDoS - *Distributed Denial of Service*), consiste em impedir que usuários tenham acesso a um determinado serviço, causando prejuízos consideráveis. Um estudo (INSTITUTE, 2015) mostra que as 641 empresas pesquisadas, que estão envolvidas com prevenção, detecção ou que foram vítimas desses ataques, relataram perdas de 1,5 milhão de dólares de prejuízo em um ano em função desses ataques.

Esse tipo de ataque representa uma ameaça significativa à continuidade dos negócios, afinal, as organizações estão cada vez mais dependentes dos serviços prestados através da Internet. O DDoS visa prejudicar tanto serviços não comerciais quanto aplicativos de negócios criticos nos quais as organizações confiam para gerenciar operações diárias, como servidores de e-mails e automação de vendas por exemplo (NETSCOUT, 2020).

Com o surgimento das Redes Definidas por Software (SDN - *Software Defined Networking*), a combinação do controle da rede separado da funcionalidade do plano de dados em conjunto com a programabilidade da rede oferecem novas possibilidades para pesquisas na área de segurança de redes. A arquitetura SDN pode ser explorada para melhorar a segurança de uma rede através de sistemas de monitoração, análise e resposta mais reativos que os sistemas convencionais (SCOTT-HAYWARD; O'CALLAGHAN; SEZER, 2013). Além disso, grande parte da carga gerencial necessária para administrar uma rede no modelo tradicional foi removida. Em outras palavras, enquanto em uma rede tradicional é necessário gerenciar, expandir e manter uma configuração de rede complexa (com racks de servidores, camadas de comutadores, roteadores, *firewalls* etc.), o modelo de rede SDN remove muito dessa complexidade para que seja possível personalizar e dimensionar serviços de rede rapidamente em relação ao modelo tradicional (GOOGLE, 2018).

Um tipo de ataque DDoS bastante conhecido é o ataque de inundação HTTP (*Hypertext Transfer Protocol*) ou HTTP flood que, por ser de difícil detecção em função de suas características, faz com que servidores web atendam a todas as requisições como sendo requisições normais sem a percepção de que, em alguns casos, trata-se de um ataque com o objetivo de fazer com que os recursos do servidor web se esgotem deixando o serviço indisponível para os usuários finais.

Uma maneira de lidar com esse tipo de ataque é considerar todos os clientes que realizam requisições HTTP para um servidor web como suspeitos em um primeiro momento e classifica-los em função do seu comportamento. Através de um redirecionamento dessas requisições, um desafio HTTP é imposto ao cliente. O desafio não será um problema para usuários legítimos mas os atacantes não conseguirão passar pelo desafio imposto. Um desafio HTTP é um método usado para mitigar ataques DDoS baseados em HTTP (LTD.,

2020).

Existem duas categorias de defesas contra ataques DDoS que envolvem redes SDN e ambas são classificadas em função dos recursos da arquitetura SDN. A primeira categoria é baseada nas soluções de defesa para proteger redes SDN quando as mesmas são o alvo dos ataques DDoS. A segunda categoria corresponde ao conjunto de soluções contra ataques DDoS em que redes SDN são utilizadas para proteger um determinado alvo desses ataques (SWAMI; DAVE; RANGA, 2019). Nessa última categoria, encontram-se os cenários em que a tecnologia SDN é utilizada para proteger redes tradicionais dos ataques.

Objetivos

Esta dissertação apresenta uma solução que protege um servidor web utilizando a tecnologia de redes SDN de ataques DDoS do tipo HTTP flood em uma rede tradicional.

A solução proposta combinou as características flexivas das redes SDN com o redirecionamento HTTP. Dessa maneira, os ataques DDoS do tipo HTTP flood foram tratados e as futuras requisições de clientes maliciosos, a contar a partir da primeira requisição, foram bloqueadas na rede. Além disso, para que o controle da rede SDN não fosse perdido em função do comprometimento dos recursos da CPU do controlador, uma VPN (*Virtual Private Network*) conectou o sistema criado a outros sistemas colaboradores. Esses sistemas ajudaram no processamento das requisições redirecionadas, apesar de estarem localizados em outros sistemas autônomos (AS - *Autonomous Systems*) na Internet.

A solução foi implementada em um ambiente virtual de testes e uma avaliação de desempenho foi realizada a fim de verificar se a solução era válida ou não.

Os resultados da avaliação de desempenho mostram uma redução de 65% no consumo de CPU do controlador SDN do sistema durante um ataque tratado por seis ASs colaboradores. Também foram observados uma redução de 78% no consumo de CPU do servidor web protegido e uma queda de 6 s para aproximadamente 400 ms do tempo de latência percebida pelos clientes na mesma situação.

Organização do Texto

Esta dissertação está estruturada da seguinte forma. O Capítulo 1 apresenta os conceitos sobre ataques DDoS e sistemas de defesa em redes tradicionais para entendimento do problema a ser tratado e a proposta desta dissertação. O Capítulo 2 apresenta os desafios de segurança para redes SDN. O Capítulo 3 trata dos trabalhos relacionados, enquanto o Capítulo 4 apresenta a arquitetura do sistema proposto, descrevendo a função de cada módulo do sistema. O Capítulo 5 aborda a avaliação de desempenho do sistema.

Finalmente, o Capítulo 6 conclui o trabalho e aponta direções de pesquisa futuras.

1 ATAQUES DDOS E MECANISMOS DE DEFESA EM REDES TRADICIONAIS

Este capítulo apresenta os principais conceitos sobre os ataques DDoS, as suas classificações e características e algumas técnicas de defesa em redes tradicionais.

1.1 Classificação e características dos ataques DDoS

Existe uma infinidade de tipos de ataques e vulnerabilidades a serem exploradas por criminosos quando os mesmos desejam realizar alguma atividade ilegal na Internet. Porém, entre todos os tipos de ataques virtuais conhecidos, o ataque distribuído por negação de serviço é capaz de causar prejuízos consideráveis, pois consiste em impedir que usuários tenham acesso a um determinado serviço.

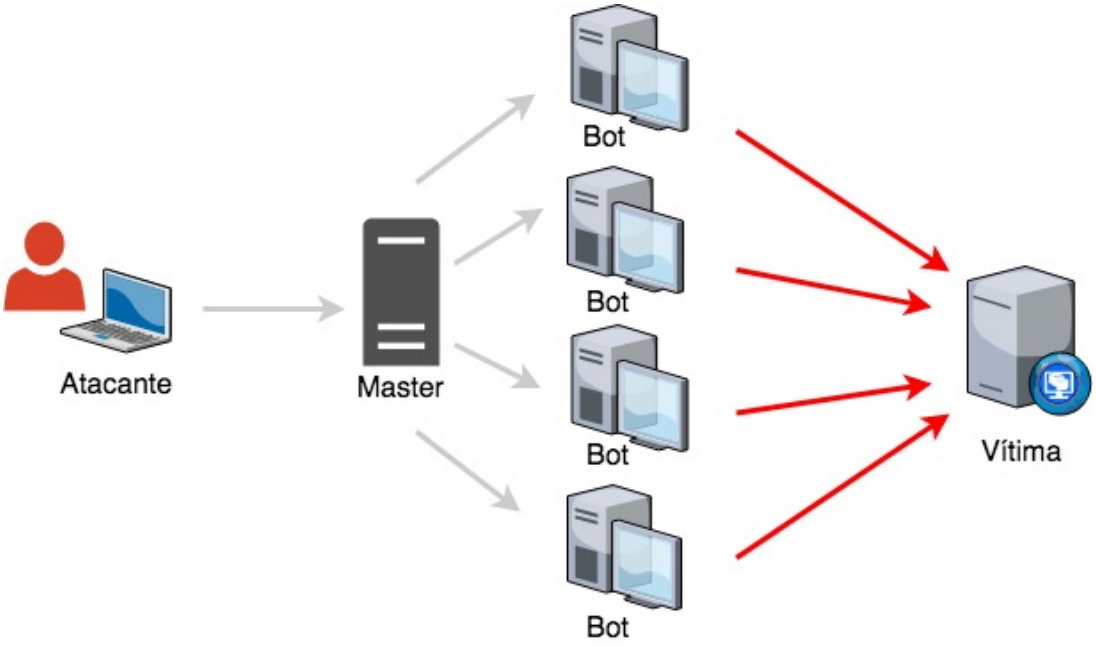
Um ataque DDoS, ilustrado na Figura 1, ocorre quando diversos tipos de pacotes de dados são enviados simultaneamente por várias máquinas para uma única máquina de destino. Essas máquinas atacantes, os *bots*, muitas vezes são controladas remotamente através de uma outra máquina, o *Master*, formando uma *botnet*. Na maioria das vezes, uma máquina passa a fazer parte de uma *botnet* quando um software malicioso (*malware*) é executado na mesma, dando opção de comando e controle ao operador da *botnet*.

Os ataques DDoS são classificados em dois tipos: os ataques de esgotamento de largura de banda e os ataques de esgotamento de conexões (recursos). Os ataques que esgotam a largura de banda de um serviço com tráfego indesejado impedem que um cliente legítimo tenha acesso a esse serviço em função do comprometimento do enlace. Os ataques de esgotamento de conexões são capazes de comprometer recursos como a memória e a capacidade de processamento dos sistemas atacados. Esse tipo de ataque torna os serviços alvos incapazes de atender a novas requisições (HUANG; KOBAYASHI; LIU, 2003).

Algumas características desses ataques de DDoS podem tornar a sua detecção e a sua defesa ineficazes. Como exemplo, apesar de uma vitima ser capaz de detectar um ataque, ela pode não conseguir se defender, dependendo do volume do ataque. Além disso, a vítima pode não ser capaz de diferenciar um pacote legítimo de um pacote de ataque. Como o atacante só precisa de volume para causar danos, um volume abaixo do detectável por sistemas de monitoração seria o suficiente para causar danos (ZHANG; PARASHAR, 2005).

Os ataques DDoS não são iguais e nem sempre possuem o mesmo propósito. Porém o resultado de um ataque bem sucedido é quase sempre o mesmo: instabilizar e indisponibilizar um serviço. A taxonomia dos ataques DDoS é apresentada na Figura 2 que mostra as classificações dos ataques DDoS em função dos seus objetivos específicos que

Figura 1 - Exemplo de botnet.



são:

- Ataques nos recursos dos servidores - procuram esgotar, por exemplo, a capacidade de processamento e de memória de servidores;
- Ataques nos recursos de rede - procuram esgotar os enlaces e os caminhos até o servidor da vítima;
- Ataques nas aplicações - procuram esgotar a aplicação, explorando suas vulnerabilidades, através de requisições para a aplicação final hospedada no servidor da vítima.

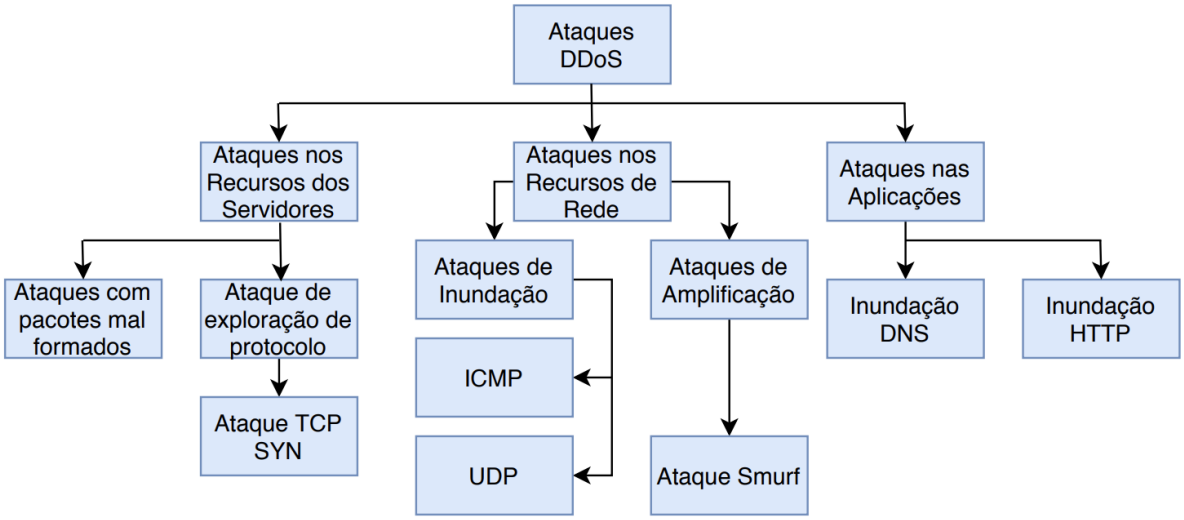
Elaborar um sistema de defesa contra ataques do tipo DDoS direcionados a um determinado serviço na Internet não é uma tarefa trivial. Dependendo do tipo de ataque, e de onde o serviço encontra-se hospedado, diferentes estratégias precisam ser adotadas para que esse serviço, mesmo sob ataque, continue a operar normalmente sem prejuízos para os usuários finais.

1.2 Mecanismos de defesa contra ataques DDoS

Os mecanismos de defesa contra ataques DDoS propostos na literatura podem ser divididos em três categorias:

- prevenção de ataques;

Figura 2 - Taxonomia dos ataques DDoS.



Fonte: Adaptado de (FAJAR; PURBOYO, 2018).

- detecção de ataques;
- reação a ataques.

Os mecanismos de prevenção de ataques têm como função a eliminação da possibilidade de ataques DDoS contra possíveis vítimas e (ou) que essas vítimas suportem o ataque sem negar os serviços aos clientes (MIRKOVIC; REIHER, 2004).

De acordo com (HARRIS; KONIKOFF; PETERSEN, 2013), para que medidas de prevenção sejam obtidas, é necessário que as boas práticas sejam seguidas assim como a segurança dos protocolos. Exemplos de boas práticas são a aplicação de *patches* de vulnerabilidades nos sistemas e a utilização de firewalls na rede. Como exemplo para segurança dos protocolos, é citada a fase de projeto do protocolo que deve ser realizada com o cuidado de evitar vulnerabilidades no protocolo que possam ser executadas em ações maliciosas (HARRIS; KONIKOFF; PETERSEN, 2013).

Os mecanismos de detecção de ataques procuram por anomalias na rede e no tráfego recebido. Uma anomalia é definida como um comportamento fora de um padrão predefinido pelos administradores da rede. Como exemplo de anomalias, destacam-se um tráfego que repentinamente aumenta em relação a uma média já definida sem motivo aparente, um aumento no consumo de CPU de um determinado equipamento sem motivo aparente ou também uma instabilidade na rede sem motivo aparente (SILVA, 2016).

Os Sistema de Detecção de Intrusão (IDS - *Intrusion Detection System*) são exemplos de sistemas capazes de implementar mecanismos de detecção de ataques. Esses sistemas exibem alertas em função de eventos suspeitos na rede baseados em assinaturas

de ataques já conhecidos dos sistemas, de alguma anomalia detectada na rede ou em ambos os casos combinados. Dependendo do posicionamento na rede, esses sistemas podem rastrear ameaças internas ou externas, dependendo do que deseja-se proteger (SNORT, 2019).

Dependendo da sua implementação, o IDS se divide em duas classes: IDS baseado em *host* e IDS baseado em rede. O IDS baseado em *host* reside em uma máquina individual e monitora as atividades locais dessa máquina em busca de eventos suspeitos (ABD; PHD, 2006). Já um IDS baseado em rede, busca comportamentos suspeitos através da monitoração de tráfegos que transitam em uma rede (G. REIS M., 2019).

Um IDS pode ser classificado também através da maneira como a sua detecção de ataques é realizada. Essas detecções podem ser realizadas através de anomalias ou através de assinaturas de ataques.

Um IDS baseado em anomalia é um sistema que monitora as ações que ocorrem numa rede com o objetivo de aprender (através de técnicas de inteligência computacional) como se comporta o sistema em seu estado padrão. Já um IDS baseado em assinaturas utiliza as informações sobre ataques já conhecidos, contidas em um banco de dados, utilizando algoritmos estatísticos para o reconhecimento de ataques (G. REIS M., 2019).

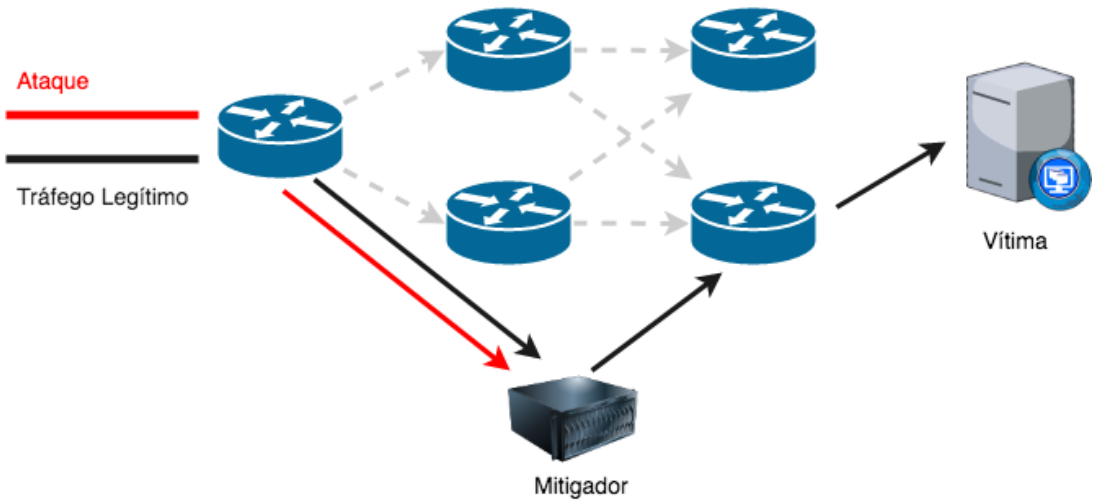
Por último, os mecanismos de reação a ataques conseguem aliviar o impacto de um ataque direcionado a vítima. Quanto mais cedo a detecção dos ataques DDoS for realizada, mais próximo de alcançar esse objetivo os mecanismos chegam (MIRKOVIC; REIHER, 2004).

Como exemplo de mecanismo de reação a ataques e, de acordo com (PATEL; QASSIM; WILLS, 2010), um sistema de prevenção contra intrusões (IPS - *Intrusion Prevention System*) é um dispositivo que possui todos os recursos de um IDS e também é capaz de reagir a ameaças detectadas de algumas maneiras que são:

- reconfigurar outros dispositivos de segurança como *firewalls* ou roteadores para bloquear futuros ataques;
- remover o conteúdo malicioso de um ataque no tráfego de rede para filtrar os pacotes ameaçadores;
- reconfigurar outros controles de segurança e privacidade nas configurações de navegadores para evitar futuros ataques.

Um outro exemplo de atuação de um mecanismo de reação é apresentado em (MAHAJAN et al., 2002), que propõe um esquema em que os roteadores intermediários (entre o atacante e o alvo do ataque) aprendem uma assinatura de congestionamento com base no endereço IP da vítima e no volume de tráfego direcionado para esse endereço IP. O termo assinatura de congestionamento foi utilizado pelos autores analogamente a uma

Figura 3 - Funcionamento de um mitigador de ataques.



Fonte: Adaptado de (ARBOR, 2018).

assinatura de ataque. Essa assinatura denota quem é a entidade causadora de um determinado congestionamento na rede. A assinatura de congestionamento não precisa analisar o tráfego para saber se é maligno ou benigno, proporcionando um ganho na resposta a uma situação considerada fora do padrão. Além disso, um mecanismo que impede que a largura de banda seja desperdiçada com pacotes inválidos (*pushback*) é utilizado para solicitar aos roteadores adjacentes que limitem o tráfego correspondente a uma assinatura específica.

Os mecanismos de detecção e reação a ataques também são implementados através de equipamentos conhecidos como mitigadores de ataques.

Os mitigadores de ataque normalmente são equipamentos com *hardwares* de custo elevado e com grande capacidade de processamento que suportam ataques em larga escala e que conseguem separar os tráfegos legítimos dos tráfegos de ataque sem deixar que o serviço hospedado no servidor da vítima seja afetado. Um exemplo de mitigador é o Arbor TMS (ARBOR, 2018) que consegue trabalhar com fluxos de até 160 Gbps e com até 110 Mpps na sua última versão existente e é capaz de tratar vários tipos de ataque DDoS como, por exemplo, ataques de reflexão de inundação, ataques de fragmentação, ataques a pilha TCP, ataques a aplicações, ataques SSL/TLS, envenenamento de DNS, entre outros. O sistema proposto nessa dissertação é capaz de tratar ataques a aplicações de inundação HTTP a um baixo custo quando comparado com um mitigador de ataques. O baixo custo possível através da utilização de *softwares* livres em toda a implementação do sistema.

Desconsiderando diversos detalhes técnicos para facilitar o entendimento do funcionamento de um mitigador, a Figura 3 mostra o funcionamento do mitigador de ataques

em sua forma mais comum de uso. O servidor alvo dos ataques é protegido quando o mitigador desvia todo o tráfego direcionado a esse servidor para análise e, após essa analise, só permite a chegada de tráfego legítimo até o servidor alvo dos ataques.

2 VULNERABILIDADES, ATAQUES DDOS E MECANISMOS DE DEFESA EM REDES SDN

O sistema de defesa desenvolvido nesse trabalho utiliza os benefícios fornecidos pela tecnologia de redes SDN para proteger um servidor web na Internet. Esse servidor web é um potencial alvo de ataques do tipo DDoS originários de *botnets* e precisa garantir estabilidade dos seus serviços durante esses ataques. Para isso, estudos sobre as redes definidas por software e sua arquitetura, os diferentes mecanismos de defesa contra ataques DDoS, o protocolo HTTP e os ataques DDoS foram necessários e possibilitaram sua implementação.

2.1 Redes definidas por software

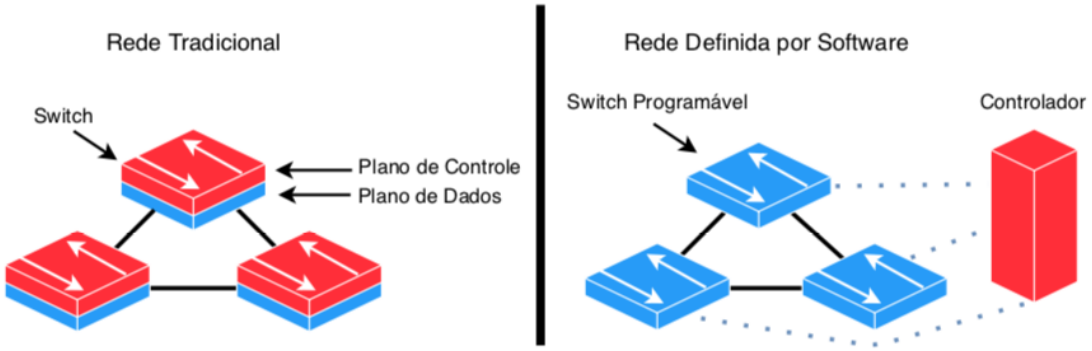
As redes de computadores tradicionais, de um modo geral e em função do seu tamanho, são redes de difícil configuração e manutenção (KREUTZ et al., 2014). Essas dificuldades são percebidas, por exemplo, na limitação do que pode ser feito com os equipamentos de rede, já que cada fabricante possui seu código proprietário fechado. Para gerenciar esses equipamentos é necessário o treinamento de uma equipe técnica especializada dificultando a capacidade de implementação de políticas de alto nível. Outra dificuldade é a integração entre planos de controle e de dados da rede em todos os dispositivos que as compõem. Essa integração faz com que cada dispositivo seja responsável por definir e executar suas tarefas de encaminhamento e controle de dados, limitando a flexibilidade da rede (BARBOSA, 2018).

Em função da necessidade de automatização de serviços de rede em nuvens privadas virtuais (VPC - *Virtual Private Cloud*) e *data centers*, as redes definidas por softwares foram desenvolvidas gerando benefícios importantes para operadoras de rede e de TI por meio da automação pretendida anteriormente (CISCO, 2019).

A Figura 4 apresenta a principal característica do SDN que é o desacoplamento do plano de controle e do plano de dados existentes em redes tradicionais, retirando o plano de controle dos dispositivos tradicionais de rede (comutadores e roteadores). O plano de dados encaminha os pacotes usando tabelas de encaminhamento geradas a partir de informações fornecidas pelo plano de controle e toda a lógica de controle é implementada por um controlador.

Como a lógica de controle em uma rede SDN é centralizada e a rede programável, tabelas de fluxos com a lógica de encaminhamento de pacotes são utilizadas pelos comutadores possibilitando assim que determinados dispositivos de rede atuem em mais de uma função. Como exemplo, comutadores atuando como firewalls, balanceadores de carga ou

Figura 4 - Descrição simples de uma rede tradicional e de uma rede definida por software.



Fonte: Adaptado de (WOLFF, 2019).

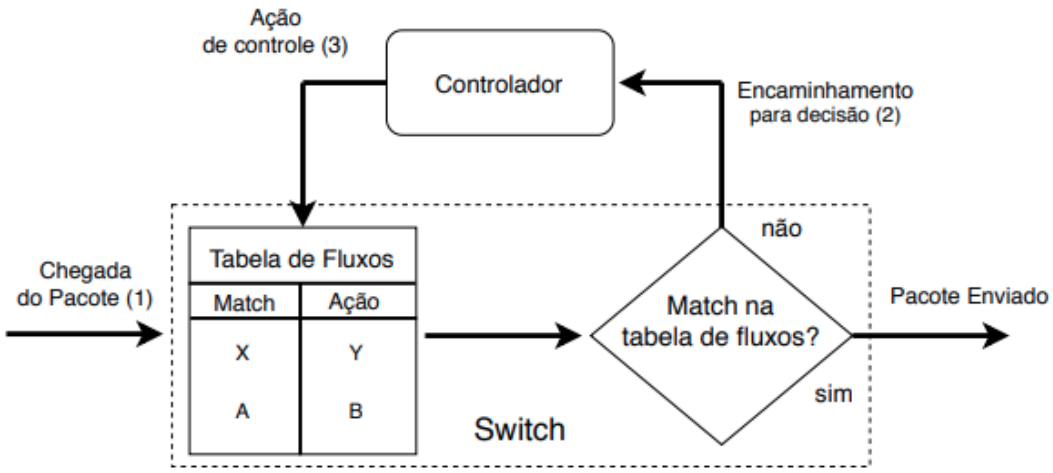
até mesmo como roteadores podem ser implementados em diferentes casos de uso dessa tecnologia de rede.

Uma definição de arquitetura para a SDN é apresentada em (KREUTZ et al., 2014), na qual essa arquitetura contém quatro pilares:

- os planos de controle e dados são desacoplados ao se remover dos dispositivos de rede a funcionalidade de controle. Os dispositivos se tornam elementos de encaminhamento;
- as decisões de encaminhamento não são baseadas em destinos mas sim em fluxos. Em SDN, um fluxo é uma sequência de pacotes entre uma origem e um destino. Todos os pacotes de um fluxo recebem políticas de serviço idênticas nos dispositivos permitindo a unificação dos comportamentos de diferentes dispositivos de rede. A programação de fluxo permite uma grande flexibilidade, limitada apenas às capacidades das tabelas de fluxo utilizadas;
- a lógica de controle é centralizada em um dispositivo externo conhecido como controlador SDN ou sistema operacional de rede (NOS - *Network Operating System*). O NOS é uma plataforma de software que fornece os recursos necessários para facilitar a programação dos dispositivos de encaminhamento;
- a rede é programável através de aplicações dispostas no NOS que interagem com os dispositivos do plano de dados, sendo essa, a principal proposta do SDN.

O funcionamento de uma rede SDN pode ser facilmente entendido a partir da Figura 5 que apresenta, de uma maneira simplificada, um exemplo genérico de funcionamento de uma rede SDN. Esse exemplo genérico inicia-se com a chegada de um novo pacote (1) que, caso não exista um fluxo para o mesmo na tabela de fluxos, o pacote será encaminhado ao controlador (2) que decidirá o destino do pacote em questão (3) ao

Figura 5 - Exemplo de funcionamento de um SDN.



Fonte: Adaptado de (SINGH et al., 2017).

inserir um fluxo para esse novo pacote na tabela de fluxos. Dependendo da ação decidida pelo controlador, o pacote poderá ser bloqueado ou encaminhado pelo comutador.

Com a implementação do SDN, diversas inovações foram obtidas em um cenário, até então, limitado pelas redes tradicionais. Ajustes em equipamentos foram facilitados dependendo do tráfego e da carga de processamento de cada um introduzindo um escopo de controle mais dinâmico. Controle de acesso em redes e gerenciamento de energia também foram facilitados através das redes SDN. Além disso, a programabilidade fornecida pelas redes SDN permite a comunicação em todos os níveis, do hardware ao software, até os usuários finais (operadores de rede) (SEZER et al., 2013).

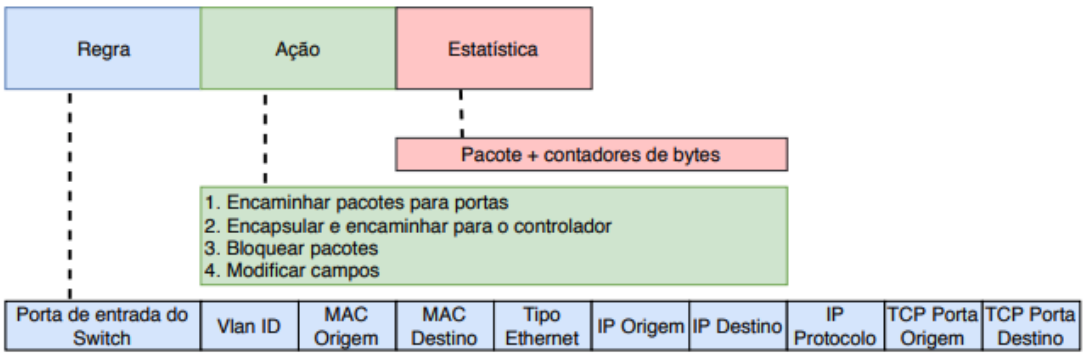
2.2 **Protocolo OpenFlow**

O conceito de redes SDN está diretamente relacionado com o entendimento do protocolo OpenFlow sendo importante observar a diferença entre ambos.

Analisando em termos de desenvolvimento de projeto, o SDN é um protótipo de rede que propõe a separação do plano de controle do plano de dados. As redes tradicionais exibem elementos de plano de controle e plano de dados centralizados em cada componente da rede. O SDN, através do desacoplamento dos planos, centraliza o controle da rede com elementos de plano de dados distribuídos na rede.

O OpenFlow é um protocolo definido para fornecer todas as funcionalidades propostas pelas redes SDN. De acordo com (MCKEOWN et al., 2008), o OpenFlow é um protocolo aberto que permite ao administrador de uma rede programar a tabela de fluxo em diferentes equipamentos. As tabelas de fluxo são utilizadas para executar ações de

Figura 6 - Campos do cabeçalho, ações e estatísticas do OpenFlow.



Fonte: Adaptado de (BRITO, 2019).

controle na rede de acordo com as necessidades existentes. Apesar das tabelas de fluxo serem diferentes entre os fabricantes, um conjunto comum de funções executadas por todos os fabricantes, permitiu a exploração das ações de controle pelo OpenFlow.

Um comutador OpenFlow realiza encaminhamento de pacotes entre suas portas sendo esse encaminhamento controlado por um controlador. Cada fluxo escrito no comutador Openflow possui uma ação associada. Segundo (MCKEOWN et al., 2008), as três ações básicas que todos os comutadores OpenFlow devem suportar são:

- encaminhamento de pacotes entre as portas do comutador OpenFlow;
- encaminhamento de pacotes para o controlador através de um canal seguro (para a decisão do controlador em inserir um novo fluxo ou não);
- fluxos de bloqueio para pacotes. Pode ser usado para contenção de ataques ou redução de tráfego.

A tabela de fluxos dentro do comutador OpenFlow identifica os fluxos para que o plano de dados execute ações sobre os pacotes que são pertencentes àquele fluxo. O comutador se guia por padrões apresentados na Figura 6 como Regra, Ação e Estatística. A Regra possui os campos do cabeçalho usados na especificação de fluxos do OpenFlow. A Ação permite ao comutador saber o que será feito com determinado pacote que pertence a um determinado fluxo. A Estatística contém os contadores que mostram, por exemplo, quantas vezes pacotes foram encaminhados através de um determinado fluxo.

2.3 Vulnerabilidades e segurança em redes SDN

Acompanhando todas as vantagens oferecidas pela tecnologia de redes SDN, diversos desafios relacionados a segurança surgiram em conjunto com esses benefícios.

Algumas vulnerabilidades conhecidas do SDN a ataques DDoS, em função da sua própria estrutura, são (SWAMI; DAVE; RANGA, 2019):

- limitação de TCAM - os comutadores OpenFlow utilizam a memória TCAM para reservar regras de fluxo. No entanto, como todo hardware, essa memória tem limite e pode ser esgotada. Essa limitação pode tornar o SDN sensível a ataques DDoS;
- ponto único de falha do controlador - o controlador, por ser o elemento central de uma rede SDN, não pode falhar. Caso isso ocorra, a rede pode entrar em colapso;
- desacoplamento do controle e do plano de dados - com o desacoplamento, um atacante pode atrapalhar a comunicação entre esses planos (realizadas através do OpenFlow) utilizando um ataque DDoS;
- comutadores sem controle próprio - os comutadores OpenFlow são dispositivos de encaminhamento simples e sem controle local. O controlador toma as decisões pelos comutadores e decide as ações para encaminhamento de pacotes. Dessa forma, os comutadores OpenFlow podem reduzir o desempenho do controlador em função de uma grande quantidade de tráfego a ser encaminhado. Não existe um controle próprio por parte do comutador nessas situações.

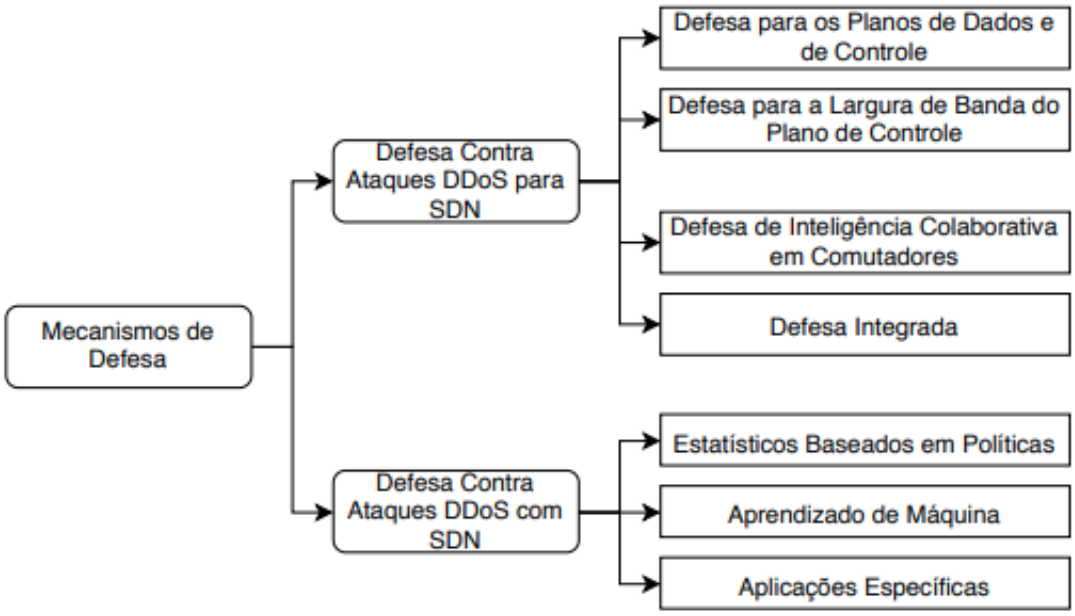
Como exemplo de vulnerabilidade, em função da centralização do controlador, a rede pode ser comprometida facilmente, afinal, caso o comutador não encontre nenhuma entrada de fluxo correspondente para novos pacotes, esses pacotes serão encaminhados para o controlador. Assim, surge uma boa chance de um atacante esgotar os recursos do controlador e ameaçar a disponibilidade de toda a rede (DAO et al., 2015).

Desta forma, para saturar a tabela de fluxos, um atacante gera uma grande quantidade de pacotes com endereços de destino desconhecidos, o que em pouco tempo fará com que a capacidade de armazenamento limitada de um comutador seja completamente consumida. Com isso, novas regras geradas por tráfego legítimo não poderão ser inseridas na tabela e seus pacotes não serão encaminhados corretamente.

Um outro exemplo é mostrado em (A. BARBOSA A., 2019) quando um ataque do tipo DDoS é realizado para saturar a tabela de fluxos do comutador, com o objetivo de atacar o plano de dados da rede. A vulnerabilidade explorada por este tipo de ataque está relacionada à maneira como o OpenFlow lida com pacotes com endereços de destino desconhecidos. Nesses casos, o protocolo insere novas regras de fluxos para cada novo endereço na tabela de fluxos. Com o objetivo de saturar essa tabela, um atacante pode gerar várias entradas aleatoriamente. Todas essas entradas serão escritas na tabela de fluxos e a mesma poderá chegar ao seu limite. Como resultado, a tabela de fluxos não aceita novas entradas e fluxos legítimos não são alocados.

Os recursos SDN também devem ser examinados na perspectiva de segurança o que acaba por trazer dois pontos de vista sobre a proteção de redes contra ataques DDoS

Figura 7 - Mecanismos de defesa em SDN.



Fonte: Adaptado de (SWAMI; DAVE; RANGA, 2019).

nesse caso. No primeiro ponto, o SDN pode se tornar a própria vítima de ataques por causa do mecanismo de controle centralizado, enquanto que, no segundo ponto, o SDN utiliza seus recursos para proteger redes convencionais (SWAMI; DAVE; RANGA, 2019).

A Figura 7 contém os mecanismos de defesa existentes em redes SDN contra ataques DDoS. As redes SDN podem ser empregadas para atenuar ataques DDoS em redes tradicionais, sendo que, um atacante também pode executar ataques DDoS na própria rede SDN. Assim, as soluções de defesa são classificadas com base nos recursos da estrutura do SDN. A linha de Defesa contra ataques DDoS com SDN contém mecanismos de proteção a redes tradicionais empregando redes SDN para essa defesa. A linha de Defesa contra ataques DDoS para SDN contém mecanismos de defesa para proteger redes SDN contra ataques DDoS.

2.3.1 Mecanismos de Defesa contra ataques DDoS para redes SDN

As redes SDN estão expostas a ataques DDoS, principalmente devido à separação da lógica de controle dos dispositivos de encaminhamento. Essas vulnerabilidades precisam ser estudadas para evitar problemas em casos de ataques a esse ambiente. Alguns mecanismos de defesa para essas vulnerabilidades foram classificados em quatro categorias de soluções, com base em possíveis causas de ameaças em ambientes SDN (SWAMI;

DAVE; RANGA, 2019), que são:

- defesa para o plano de dados e de controle - para o plano de dados, aborda soluções para evitar que comutadores inundem os controladores durante um ataque por serem dispositivos sem controle local e de simples encaminhamento. Para o plano de controle, aborda soluções contra saturação do plano de controle em ataques direcionados aos mesmos;

- defesa para a largura de banda do plano de controle - aborda soluções que evitam que o plano de controle fique sem comunicação através do congestionamento do seus enlaces;

- defesa de inteligência colaborativa em comutadores - aborda soluções de inclusão de controles implementados nos comutadores SDN para que não mais sejam simples dispositivos de encaminhamento. Isso evitaria que o canal de controle fosse sobrecarregado assim como o próprio controlador;

- defesa integrada - são soluções que utilizam outros dispositivos integrados com a rede SDN para amenizar impactos tanto no plano de dados quanto no plano de controle de redes SDN.

Como exemplo de mecanismo de defesa para o plano de dados, em (DURNER et al., 2017) uma proposta focada em ataques de estouro de tabelas em comutadores foi sugerida com abordagem de coleta de dados estatísticos para detecção de ataques em conjunto com um método de mitigação de fluxos maliciosos. Uma tabela classifica clientes suspeitos usando um *hash* para cada entrada e, em função dessa tabela, novas regras são definidas para manipular os ataques.

Para os mecanismos de defesa integrada o exemplo de (DRIDI; ZHANI, 2018) é mostrado através de um sistema chamado SDN-Guard que, com a colaboração de um IDS, tenta reduzir os efeitos de ataques de DoS no controlador SDN, na largura de banda de comunicação com o controlador e nos recursos do comutador. O SDN-Guard é utilizado com o controlador como um aplicativo de segurança. Os dados enviados pelo IDS contém alertas de tráfego malicioso detectado pelo mesmo. Com base nesse alerta, o SDN-Guard toma decisões adequadas para mitigar os ataques.

2.3.2 Mecanismos de Defesa contra ataques DDoS com SDN em redes tradicionais

Com o advento da tecnologia das redes SDN, novos mecanismos de detecção e reação contra ataques DDoS surgiram e foram benéficos para a perspectiva de segurança nas redes tradicionais. Os mecanismos de defesa para esse cenário foram classificados em três categorias (SWAMI; DAVE; RANGA, 2019):

- estatísticos e baseados em políticas - para mecanismos estatísticos, são abordos soluções já conhecidas de monitoração de tráfego de uma rede (p.ex. entropia) mas com uma capacidade de detecção e resposta mais rápida e dinâmicas em função das características das redes SDN. Para mecanismos baseados em políticas, a defesa normalmente define regras (políticas) para os fluxos de tráfego na rede. Caso determinado fluxo se encaixe nesse padrão pré definido, o sistema tratará como tráfego malicioso;

- aprendizado de máquina - são mecanismos que abordam a utilização de algoritmos de aprendizado de máquinas em conjunto com os recursos das redes SDN;

- aplicações específicas - são mecanismos que utilizam aplicações desenvolvidas para lidar com ataques específicos de maneira colaborativa com as redes SDN.

Um exemplo de mecanismo de defesa baseado em políticas é mostrado em (SAHAY et al., 2017), em que o mecanismo mitiga ataques direcionados às redes dos clientes. Um ISP, nessa abordagem, funciona como um provedor de serviços de segurança que conta com a colaboração dos clientes para lidar ataques DDoS. Além disso, as decisões tomadas para um cliente não afetam os outros clientes ligados ao mesmo ISP. O sistema conta com um módulo de monitoração que recebe alertas dos clientes para armazenamento dessas informações. Além desse módulo, existe o módulo de politicas e informações sobre ataques que é utilizado para decidir qual politica utilizar com determinado tráfego suspeito baseados nos alertas e dados armazenados.

Em (ALSHAMRANI et al., 2017), um sistema de defesa contra uma grande variedade de ataques DDoS em SDN foi proposto. Através de um algoritmo de aprendizado de máquina, as informações de tráfego são monitoradas frequentemente para serem classificadas. Existe um módulo de detecção de ataques DDoS que se baseia em características definidas como padrão de tráfego de rede. Quando o tráfego se comporta fora dos padrões normais predefinidos, o sistema reduz a capacidade de banda do atacante e mantém a capacidade de banda normal para clientes legítimos.

Um exemplo de mecanismo de defesa com aplicações específicas é apresentado em (KIM; SHIN, 2017). A proposta dos autores nesse trabalho foi a criação de um mecanismo de mitigação de inundação de enlaces denominado SDHoneyNet. Esse mecanismo utiliza recursos do SDN em conjunto com um *Honeypot*. Os *Honeypots*, em segurança da informação, são recursos computacionais de segurança dedicados a serem sondados, atacados ou comprometidos (CERT.BR, 2020). No método proposto, o recurso criado força um atacante a visitar a topologia de uma rede falsa a fim de reunir informações sobre suas atividades.

Nesse trabalho, o sistema proposto protege um serviço na web contra ataques de esgotamento de recursos de aplicações mais especificamente do tipo inundação HTTP. O

ataque HTTP flood é um ataque do tipo DDoS onde o atacante utiliza requisições válidas de HTTP GET (ou POST) para atacar uma aplicação em um servidor na Internet. Esses ataques são volumétricos e, geralmente, realizados através de *botnets*. Como as requisições normalmente são válidas, o ataque torna-se de difícil detecção por sistemas comuns de defesa, como IDSs por exemplo. Além disso, detecções baseadas em volume de tráfego também podem falhar com esse tipo de ataque; afinal, o volume de tráfego pode ser abaixo dos limites de detecção definidos previamente dependendo do comportamento e do tipo de aplicação atacada (VERMA; XAXA, 2016).

A dificuldade em torno da detecção de ataques HTTP flood é observada quando utilizam-se algumas técnicas convencionais contra ataques de negação de serviço, como por exemplo a utilização de IDSs (YATAGAI; ISOHARA; SASASE, 2007). Através de assinaturas de ataques contidas nos IDS, diversos tipos de ataques podem ser detectados através da monitoração dos pacotes que ingressam em uma determinada rede. No entanto, como ataques do tipo HTTP flood utilizam o protocolo HTTP sem anomalias (com requisições válidas e aceitas pela aplicação), esse tipo de ataque é de difícil detecção nesse caso.

O sistema proposto por essa dissertação é capaz de lidar com os ataques do tipo HTTP flood mantendo o sistema operacional e estável durante um ataque, sem prejuízos para os usuários finais.

No capítulo seguinte, os trabalhos na linha de pesquisa dessa dissertação são apresentados. Esses trabalhos serviram de inspiração para o desenvolvimento do sistema proposto nessa dissertação.

3 TRABALHOS RELACIONADOS

Este capítulo descreve o estado da arte, relacionando esta dissertação com trabalhos já publicados nessa linha de pesquisa. A seguir descrevem-se os trabalhos listados conforme suas contribuições.

Uma maneira de mitigar ataques utilizando uma rede SDN é apresentada em (DAO et al., 2015). Na solução proposta, em que todas as requisições que chegam até o controlador têm seus IPs armazenados em uma tabela e um contador é associado a cada um desses IPs para registrar a quantidade de pacotes enviados pelos mesmos. Após isso, o controlador insere fluxos para as novas entradas com os seus *hard timeout* e *idle timeout* com intervalos de tempo menores do que os valores encontrados em fluxos inseridos para pacotes considerados legítimos. O *hard timeout* elimina o fluxo inserido em um determinado tempo preconfigurado, enquanto o *idle timeout* elimina o fluxo inserido caso em um determinado tempo o fluxo não seja utilizado. Esses valores menores foram escolhidos para que os fluxos não durem muito tempo; afinal, esses fluxos novos ainda não são considerados confiáveis. Se a quantidade de pacotes e conexões dentro do tempo estabelecido pelo controlador não estiverem de acordo com o que o sistema considera como normal, o IP é classificado como atacante e um fluxo de bloqueio é escrito no OVS (*Open vSwitch*) pelo controlador. Caso contrário, o controlador atualiza os valores de *idle* e *hard timeout* nos fluxos com valores de fluxos confiáveis já estabelecidos pelo sistema.

O trabalho de (SAHAY et al., 2015) apresenta uma maneira de mitigar ataques utilizando uma rede SDN em conjunto com um *middlebox* que aplica políticas de segurança contra os ataques direcionados aos clientes. Os tráfegos de entrada recebem identificações (*tags*) e são direcionados para os melhores enlaces na rede através de fluxos escritos no comutador de entrada. Como o controlador não tem conhecimento da origem desses tráfegos, o mesmo considera todos os tráfegos de entrada como legítimos em um primeiro momento. Quando o tráfego chega até a rede do cliente e é identificado como suspeito, o controlador recebe um alerta da rede do cliente e esse tráfego é direcionado para um caminho na rede que leva até o *middlebox*. Através da alteração do *tag* do tráfego, ou seja, o fluxo referente a um determinado tráfego que agora é considerado malicioso, o tráfego é direcionado para o *middlebox* e não mais diretamente para o cliente. No *middlebox* esse tráfego é reprocessado para posteriormente ser enviado ao cliente.

Em (RASHIDI; FUNG, 2016), é apresentado um sistema denominado CoFence que, através de virtualização de um Sistema de Prevenção de Intrusão (IPS - *Intrusion Prevention System*), é capaz de redirecionar o tráfego de um determinado ataque para outras redes pertencentes ao domínio CoFence. O domínio CoFence trata-se de um conjunto de redes que confiam umas nas outras através de contrato de serviços. Assim que o ataque é detectado, o tráfego excedente começa a ser redirecionado para outras redes do

domínio, para ser filtrado pelos IPSs das redes colaboradoras, e somente tráfegos válidos retornam à rede de origem. A ideia é que as redes pertencentes ao domínio CoFence sejam capazes de ajudar umas às outras em situações de ataques DDoS, criando assim um sistema de defesa colaborativa. O trabalho concentra-se, além das virtualizações das funções das redes (NFV - *Network Functions Virtualization*), em um mecanismo de alocação de recursos. Esse mecanismo determina o quanto de recurso um domínio deve oferecer aos solicitantes para que o recurso seja distribuído de forma otimizada.

Uma técnica conhecida como defesa de alvo móvel (MTD - *Moving Target Defence*) pode ser bastante explorada em ambientes SDN em função da flexibilidade fornecida por essas redes. Essa técnica modifica algumas configurações das redes para dificultar ações de atacantes e, com as redes SDN, essas mudanças podem ser feitas rapidamente em função de eventos ou comportamentos predefinidos. A troca do endereço IP de um servidor e a troca de fluxos de encaminhamento são exemplos de modificações de configurações de uma rede. O trabalho de (JAFARIAN; AL-SHAER; DUAN, 2012) utiliza a flexibilidade fornecida pelas redes SDN para ocultar endereços IPs de varreduras dos atacantes. Utilizando uma abordagem conhecida como Mutação de Host Aleatório OpenFlow (OF-RHM - *OpenFlow Random Host Mutation*), os autores propõem uma forma de esconder dos atacantes os endereços IPs reais através de IPs Virtuais (VIPs - *Virtual IPs*). Esses VIPs são trocados aleatoriamente e rapidamente, preservando o serviço em seu IP real de origem, que encontram-se atrás desses VIPs. O OF-RHM pode invalidar a coleta de informações de rastreadores externos em até 99%.

Baseando-se na flexibilidade fornecida pelas redes SDN, (LIM et al., 2014) apresentam uma solução voltada para ataques mais específicos em servidores web, sendo que a estratégia é aguardar uma resposta do cliente utilizando um CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*). Quando novas requisições chegam até os servidores protegidos pelo sistema, o controlador insere fluxos permissivos para que a comunicação entre cliente e servidor seja possível. Quando o sistema percebe que encontra-se sob ataque, através de um número máximo de conexões definido pelos autores, o sistema muda o IP desse servidor, insere fluxos permissivos para o novo endereço, finaliza as conexões para o IP antigo e envia um redirecionamento para o novo endereço. Caso o cliente seja capaz de seguir esse redirecionamento e, se o CAPTCHA for corretamente respondido, a conexão é estabelecida. Caso contrário, um fluxo de bloqueio é inserido no sistema para a requisição já classificada como maliciosa.

Contra os ataques do tipo SYN flood e HTTP flood, o trabalho de (LUKASEDER et al., 2017) propõe um sistema de defesa utilizando redes SDN que utiliza uma etapa de detecção e outra de observação de possíveis ataques. A etapa de detecção avalia se um determinado servidor encontra-se sob ataque através do tempo de atraso de requisições HTTP realizadas para o mesmo. Se o tempo de atraso for superior a um valor definido pelos autores,o servidor pode estar sob ataque e a etapa de observação é ativada. A etapa de

observação passa a verificar os pacotes destinados ao serviço alvo do ataque. Informações como quantidade de conexões abertas são registradas e os clientes recebem uma pontuação em função do assédio a esse serviço. Essa pontuação classifica cada cliente em função da sua periculosidade. Depois que o controlador é informado sobre clientes suspeitos no final da etapa de observação, os OVSs do sistema são programados para redirecionar os pacotes suspeitos para um servidor *CAPTCHA*. Se um cliente suspeito resolver um *CAPTCHA*, ele recuperará o acesso por um determinado tempo. Caso contrário, um fluxo de bloqueio será criado para o cliente suspeito.

Em (Hameed; Khan, 2017) os autores propõem um esquema colaborativo de mitigação de ataques DDoS usando SDN, no qual diferentes controladores encontram-se em diferentes ASs. Esses controladores se comunicam através de um protocolo seguro denominado C-to-C. Esse protocolo permite que informações sobre ataques detectados e ações de bloqueio em múltiplos ASs simultaneamente, sejam compartilhadas entre os controladores para serem executadas rapidamente. Quando o sistema detecta IPs suspeitos, os controladores recebem via C-to-C uma lista contendo esses IPs e os fluxos de bloqueio são atualizados em todo o sistema.

Outra preocupação quando se trabalha com redes SDN corresponde ao controlador. O controlador, por ser o ponto central da rede, torna-se um elemento muito importante para a segurança como um todo. Caso o controlador seja sobrecarregado, todo o desempenho da rede pode ser comprometido. Um ataque DDoS direcionado a um determinado ponto da rede SDN pode gerar problemas no controlador apesar do atacante não saber onde o mesmo encontra-se dentro da rede SDN. Esses problemas podem ser ocasionados tanto por gargalos nos enlaces entre o controlador e os comutadores quanto pelo alto processamento exigido do controlador durante ataques.

Em (WANG; XU; GU, 2015), a preocupação com o comprometimento do controlador durante um ataque DDoS é tratada através da criação do FloodGuard, que é um sistema que utiliza dois módulos de atuação para tratamento desse tipo de ataque. Esses módulos são aplicações de controle em que um módulo é o analisador proativo de regras de fluxo e o outro é conhecido como módulo de migração de pacotes. O primeiro módulo insere no comutador regras de fluxo específicas através da análise das mensagens *packet-in* destinadas ao controlador. Através de um algoritmo desenvolvido pelos autores, os fluxos proativos são inseridos em função do que o controlador poderá lidar em um instante futuro. Essa ação evita um alto consumo de recursos do controlador pois muitas mensagens *packet-in* seriam processadas durante um ataque. O segundo módulo, através de algoritmos desenvolvidos pelos autores, detecta ataques através da taxa em tempo real de mensagens *packet-in* e a utilização da infraestrutura (p.ex. memória do controlador e CPU). Com esses dados, o módulo calcula a porcentagem de uso dos recursos da infraestrutura e classifica como ataque a partir de um certo valor definido. Detectado o ataque, o módulo armazena temporariamente os fluxos de ataque (não encontrados na tabela do

comutador), realiza uma classificação dos mesmos e envia ao controlador somente fluxos benignos através do algoritmo *round-robin* e com uma taxa de pacotes/s controlada para não sobrecarregar o controlador.

O trabalho de (DRIDI; ZHANI, 2016) mostra um sistema de proteção que evita a sobrecarga do controlador, o congestionamento do enlace de comunicação entre o controlador e os OVSs e também a saturação das tabelas de encaminhamento de fluxo dos comutadores. O sistema é composto por três módulos que são: módulo de agregação, módulo de monitoração e módulo de gerenciamento de fluxos. O módulo de agregação evita que fluxos maliciosos permaneçam por um longo tempo na tabela dos comutadores, agregando fluxos maliciosos com propriedades parecidas (p.ex., mesma origem e destino) e os encaminhando para o mesmo enlace de saída. O módulo de monitoração colhe dados estatísticos sobre os fluxos para serem usados pelos outros módulos e o módulo de gerenciamento de fluxos determina o roteamento de cada fluxo e o *hard timeout* de cada um deles baseado na probabilidade de ameaça do fluxo, fornecida por um IDS.

A proposta dessa dissertação foi inspirada em trabalhos que utilizam técnicas de redirecionamento de tráfego para estruturas colaboradoras localizadas em outras redes na Internet contra ataques DDoS, como (RASHIDI; FUNG, 2016) e (Hameed; Khan, 2017). Além desses trabalhos, também serviram de inspiração trabalhos que se preocupam com o desempenho dos controladores das redes SDN, como (WANG; XU; GU, 2015) e (DRIDI; ZHANI, 2016). Além desses trabalhos, o trabalho de (LIM et al., 2014) também serviu de inspiração. Esse trabalho, através da flexibilidade fornecida pelas redes SDN, cria uma defesa associando essa flexibilidade a métodos de redirecionamentos de URL. O sistema proposto permite que todas as requisições de acessos oriundas da Internet cheguem até um IP exposto na Internet pelo sistema. Esse IP exposto contém um serviço que redireciona as requisições para um outro IP que hospeda o serviço desejado pelo usuário final. Ao utilizar estruturas colaboradoras, o sistema previne problemas no controlador durante um ataque distribuindo os fluxos em excesso. A estrutura colaboradora realiza o mesmo procedimento de redirecionamento, porém, quando o cliente responder a esse redirecionamento, será encaminhado ao serviço desejado na rede original através de uma VPN estabelecida entre as redes. O OVS da rede original (isto é, a rede que contém o serviço hospedado), percebe que qualquer tráfego oriundo da VPN é confiável e o cliente acessará o serviço normalmente. A VPN permite tanto a comunicação segura entre os controladores quanto a confiabilidade do tráfego entre os mesmos. Além disso, a VPN permite que o sistema seja implementado como se fosse uma rede local apesar de sua partes estarem distribuídas pela Internet.

4 ARQUITETURA E FUNCIONAMENTO DO SISTEMA PROPOSTO

Para um melhor entendimento do funcionamento do sistema de defesa proposto, esta seção apresenta a arquitetura do sistema e como o sistema funciona integrado a outros ASs parceiros na Internet.

O sistema é composto de um Sistema Local (SL) que pode trabalhar com um ou mais ASs Colaboradores (ASCs), como apresentado na Figura 8. O Sistema Local é o sistema que abriga o servidor com a aplicação web e contém um Armazenador, um Redirecionador e uma rede SDN composta por um Controlador e um comutador virtual (OVS). É de se destacar que o Sistema Local, sem um ASC funcionando em conjunto, é capaz de se defender normalmente. A colaboração de outro AS é uma ampliação do sistema que visa proteger os recursos do Controlador em situações de ataque e reduzir a latência percebida pelos clientes na mesma situação. Um ASC é uma estrutura que contém uma rede SDN composta por um Controlador, um OVS e um Tradutor. Os controladores dos ASCs, o Redirecionador e o Armazenador se comunicam através de uma VPN Controladores assim como o Tradutor e a aplicação web se comunicam através de uma VPN Tradutor.

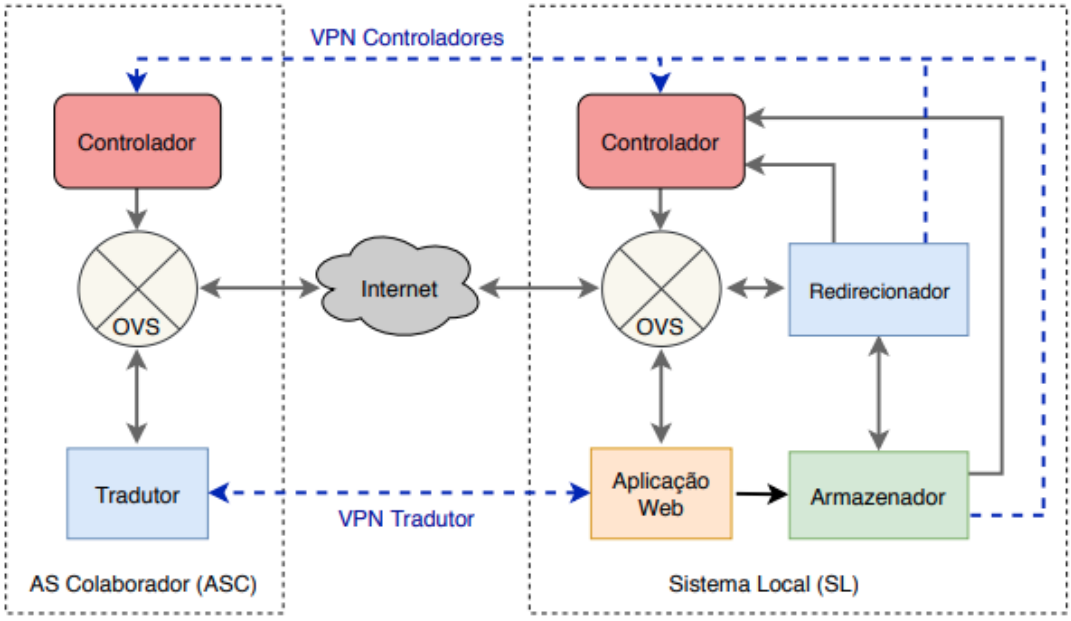
A seguir são descritos os principais componentes do sistema proposto apresentado na Figura 8 e suas funcionalidades.

4.1 Sistema Local

O Sistema Local possui essa denominação pois abriga o serviço a ser protegido dos ataques vindos de *botnets*. Independentemente do AS que o sistema é alocado e do número de ASs Colaboradores que compõem o sistema como um todo, o SL é parte central de todo o sistema. Os blocos que compõem o SL são: o Redirecionador, a aplicação web e o Armazenador.

O Redirecionador é um servidor Linux que hospeda uma aplicação web que redireciona todas as requisições para determinados destinos específicos, além de ser capaz de se comunicar com o controlador e o Armazenador. É a parte central de todo o sistema. Com um IP exposto na Internet cadastrado no DNS como sendo o IP da aplicação web e através de um fluxo permissivo instalado no OVS para esse IP, as requisições web são feitas diretamente a esse bloco do sistema. Para cada requisição recebida pelo Redirecionador, algumas entradas de dados são escritas no Armazenador. Após a escrita no Armazenador, um fluxo permissivo para o IP de origem da requisição é escrito no OVS através do Controlador. Esse fluxo permite que o IP de origem da requisição seja capaz de alcançar o IP da aplicação web através de uma requisição futura e assim, o cliente consegue acesso

Figura 8 - Arquitetura do sistema proposto.



ao serviço desejado. Após a escrita do fluxo no OVS, o cliente é redirecionado para o IP da aplicação web por meio da resposta 302 *status code* do protocolo HTTP. Quando o cliente recebe a resposta da sua requisição ao sistema com o código 302, o cabeçalho HTTP contém um campo denominado *Location* que contém o IP a ser acessado em uma nova requisição automática. No caso do sistema proposto, o IP do campo *Location* é o IP da aplicação web que já contém com fluxo permissivo escrito no OVS previamente. Esse fluxo foi escrito em função da primeira requisição realizada para o Redirecionador.

A utilização do método de redirecionamento foi escolhida em função desse método tornar-se um desafio HTTP para os usuários que pretendem acessar a aplicação web. É esperado que os usuários legítimos superem o desafio enquanto que os usuários maliciosos não sejam capazes de superar o mesmo desafio. Esse desafio envia aos usuários mensagens de redirecionamento 302 como resposta para as suas requisições. Um usuário legítimo através de um navegador web passará nesse desafio, enquanto que um atacante, ao não implementar uma pilha HTTP completa, ignorará esse redirecionamento enviando a solicitação original novamente. Esse tipo de desafio é considerado eficaz mas possui a desvantagem de, em algumas situações, realizar bloqueios indevidos conhecidos como falsos positivos. Algumas ferramentas e *botnets* mais sofisticadas podem investir na superação desse desafio até mesmo utilizando navegadores web, porém, o atacante deverá investir em recursos do seu lado que, na maioria dos casos, diminuirá a capacidade de ataque (LTD., 2020). Além disso, em função da dificuldade em se modificar as configurações de uma *botnet* rapidamente, o método de redirecionamento foi considerado adequado e utilizado nesse trabalho.

A aplicação web encontra-se hospedada em um servidor Linux e se comunica com o Armazenador sempre que um determinado cliente realiza requisições válidas para essa aplicação. Quando um cliente consegue realizar requisições para a aplicação web, sabe-se que o mesmo já realizou requisições para o Redirecionador, já foi cadastrado no Armazenador e possui um fluxo permissivo instalado no OVS para o acesso ao seu destino final (aplicação web). Na primeira requisição realizada com sucesso, a aplicação web atualiza o estado do IP de origem do cliente no Armazenador. Essa atualização é importante para garantir o fluxo permissivo por tempo indeterminado no OVS. Essa permissão é concedida pois o sistema classificou o cliente como legítimo; afinal, todas as etapas de validação anteriores foram cumpridas pelo cliente. Caso o IP de origem de uma requisição não seja válido, essa atualização no Armazenador não ocorre, levando o sistema a classificar o cliente como atacante.

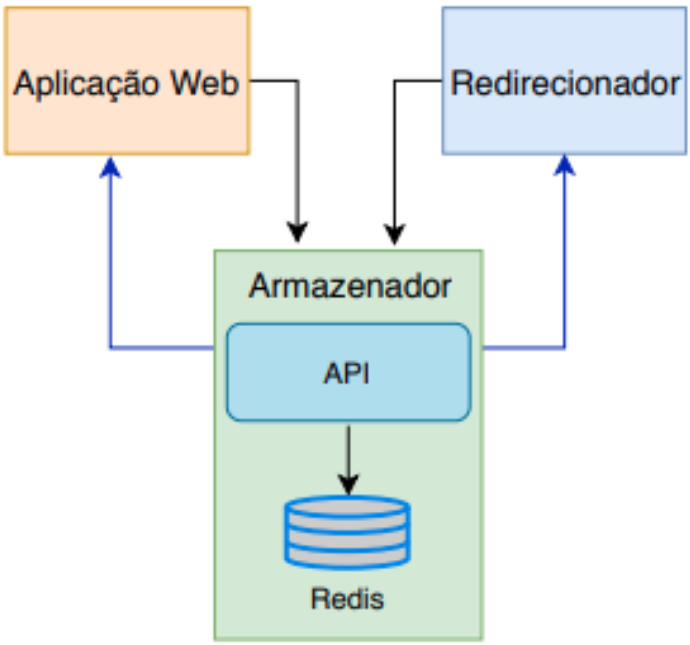
O Armazenador é um servidor Linux que contém a aplicação de armazenamento das informações do sistema, além de ser capaz de se comunicar com o controlador. O Armazenador registra informações referentes às requisições de entrada no sistema. Sempre que uma requisição entra no sistema, o Redirecionador realiza uma consulta ao Armazenador para verificar se o IP referente a requisição de entrada é conhecido pelo sistema. Dependendo do estado que o IP é classificado no Armazenador, o Redirecionador tomará diferentes decisões. Poderá ignorar a requisição, caso tenha sido realizada por um IP classificado como malicioso, ou enviará uma resposta com o redirecionamento a ser seguido pelo cliente. O Armazenador também é atualizado frequentemente pela aplicação web a cada requisição recebida pela mesma. Essa ação é importante pois permite ao sistema saber quando um cliente é legítimo ou não.

Uma outra função do Armazenador é monitorar os seus registros para bloquear IPs suspeitos de serem maliciosos. Caso o Armazenador contenha em seus registros IPs que não foram validados pela aplicação web após um intervalo de tempo, esses IPs serão classificados como maliciosos. Com essa classificação, os IPs suspeitos são atualizados no Armazenador como IPs de ataque e fluxos de bloqueios são inseridos no OVS de entrada do sistema pelo Armazenador para cada IP encontrado. Além desses fluxos de bloqueio, os fluxos permissivos inseridos previamente para o IP de destino do redirecionamento serão removidos. Caso esses IPs realizem novas requisições após serem classificados como maliciosos, os mesmos não conseguirão acesso ao Redirecionador.

A aplicação de armazenamento do Armazenador foi implementada através de uma API (*Application Programming Interface*) com o *framework* Flask (FLASK, 2019). Suas informações são armazenadas utilizando um armazenador de dados em memória conhecido como Redis (REDIS, 2019). O Redis pode atuar como um banco de dados ou também como um *cache* e, por conta da sua velocidade e facilidade de uso, foi escolhido para lidar com o registro dos IPs que realizam requisições para o sistema.

A comunicação do Armazenador com a aplicação web e o Redirecionador é exibida

Figura 9 - API de comunicação com o Armazenador.



na Figura 9. O Redirecionador atualiza os dados sempre que necessário e realiza consultas para decidir as ações a serem tomadas em função do estado de determinados IPs. A aplicação web atualiza o Armazenador indicando que determinados IPs são válidos pois seguiram as instruções de redirecionamento previamente enviadas aos mesmos. O Armazenador atualiza o estado dos IPs que ainda não foram validados pela aplicação web para o estado de atacante caso esses IPs estejam a mais de 10 s sem validação.

Quando o SL opera com colaboração de ASCs, o sistema passa a distribuir as requisições de entrada entre ASs colaboradores na Internet. Essa colaboração é proposta devido aos limites de processamento do Controlador do SL, afinal, ataques do tipo DDoS normalmente necessitam de muita capacidade de processamento para serem mitigados. Como a capacidade de processamento de qualquer equipamento possui um limite, dividir as requisições com ASs colaboradores pode ser uma solução para o limite dessa capacidade.

A Tabela 1 contém as informações das requisições registradas no Armazenador. Cada requisição é registrada através dos seguintes campos:

- IP - contém o IP de origem da requisição. Esse IP é o identificador único para um determinado cliente a partir do momento da sua primeira requisição;
- CONTROLE - contém as informações do estado, ID do Controlador e tempo para cada IP registrado.

Para um melhor entendimento sobre as informações registradas e o funcionamento do sistema, através da Tabela 1 são mostradas três requisições com estados diferentes

Tabela 1 - Exemplo de Informações de Armazenamento.

IP	CONTROLE
195.3.1.4	2
12.44.32.199	1_1_156969833
178.43.55.12	0_1_156969839

entre si. Cada IP é associado a um conjunto de caracteres que guarda as informações de controle separadas pelo caractere "_". O primeiro caractere é referente ao estado de um determinado IP. O estado contém os valores definidos pelo sistema para classificar um determinado cliente. Os valores 0, 1 e 2 classificam, respectivamente, um cliente como não validado, validado e atacante. Quando o IP é definido como atacante, somente o valor do estado fica registrado pois essa informação é salva apenas para registro; afinal, o IP já encontra-se bloqueado na entrada do sistema. O segundo caractere contém o ID do ASC (e do SL) que o IP deverá ser redirecionado. O sistema é capaz de saber, através desse ID, qual controlador utilizar e para qual IP as requisições devem ser redirecionadas. O terceiro caractere contém o valor do momento no qual o IP foi registrado no Armazenador. Esse valor é salvo em *Unix Time* e, através desse valor, o Armazenador é capaz de calcular há quanto tempo a requisição encontra-se cadastrada.

Ainda no exemplo, os IPs das requisições são 195.3.1.4, 12.44.32.199 e 178.43.55.1. Esses IPs contêm, respectivamente, os estados atacante, validado e não validado. Além disso, suas respectivas requisições entraram no sistema em momentos diferentes. A primeira requisição foi classificada como maliciosa pois o IP de origem não realizou outra solicitação à aplicação web. A aplicação web não atualizou o registro dessa requisição no Armazenador que verificou que passados 10 s nada aconteceu. Assim, a requisição seria atualizada para maliciosa no Armazenador, o fluxo permissivo para a aplicação web seria removido e fluxos de bloqueio em todos os OVSs seriam adicionados para o IP solicitante. A segunda requisição foi classificada como válida pois o IP de origem realizou outra solicitação a aplicação web. A aplicação web atualizou o registro dessa requisição no Armazenador e o Redirecionador verificou que passados 10 s o IP de origem realizou uma nova requisição como era esperado. O tempo de 10 s foi adotado pelo sistema por ser maior que a média da latência em situações de ataque, ou seja, é um tempo razoável para essa verificação. A terceira requisição ainda não foi classificada pelo sistema e contém apenas o fluxo permissivo inicialmente adicionado para todas as requisições.

4.2 **AS Colaborador**

O AS Colaborador é uma estrutura criada para receber requisições redirecionadas pelo SL. O sistema proposto pode operar com quantos ASCs forem possíveis. Cada ASC

possui apenas um bloco denominado Tradutor. Essa estrutura pode ser disponibilizada por uma empresa que, através de um contrato pré-estabelecido, seria remunerada pelo volume de tráfego redirecionado ou pelo tempo de utilização da estrutura, por exemplo. A colaboração também poderia ser gratuita caso o SL e diferentes ASCs pertençam a mesma empresa, não necessariamente em diferentes ASs. Essa estrutura é dedicada apenas ao acesso a aplicação web no SL não existindo a possibilidade do SL trabalhar como um ASC. Além disso, somente requisições que acessaram previamente o Redirecionador terão fluxos permissivos escritos para acesso a aplicação web via VPN. Qualquer outra requisição não acessará a aplicação web por não possuir um fluxo permissivo para esse acesso.

O Tradutor foi implementado para receber as requisições redirecionadas pelo SL e encaminhá-las para a aplicação web através de uma VPN. O sistema assume que todo o tráfego que percorre essa VPN é confiável, ou seja, acessa a aplicação web diretamente.

Quando uma requisição chega no SL e é redirecionada para um ASC, o Redirecionador registra os dados dessa requisição no Armazenador. Através da VPN Controladores, o Redirecionador se comunica com o Controlador do ASC para inserir um fluxo permissivo no OVS do ASC. Nesse caso, o IP de destino desse fluxo não é o IP da aplicação web mas o IP do Tradutor.

A Figura 10 apresenta um exemplo de funcionamento do Tradutor. Quando o fluxo permissivo é inserido no OVS do ASC para o IP de destino 107.13.1.2 (endereço do Tradutor), a requisição redirecionada pelo SL acessa esse IP e o Tradutor realiza um NAT (*Network Address Translation*) no IP de origem para o IP inválido 10.10.1.2. O IP de destino também é traduzido para um IP dentro da rede 10.10.1.0/24 que pertence a aplicação web. Esse IP traduzido acessa a aplicação web através da VPN que, nesse caso, possui o IP 10.10.1.9.

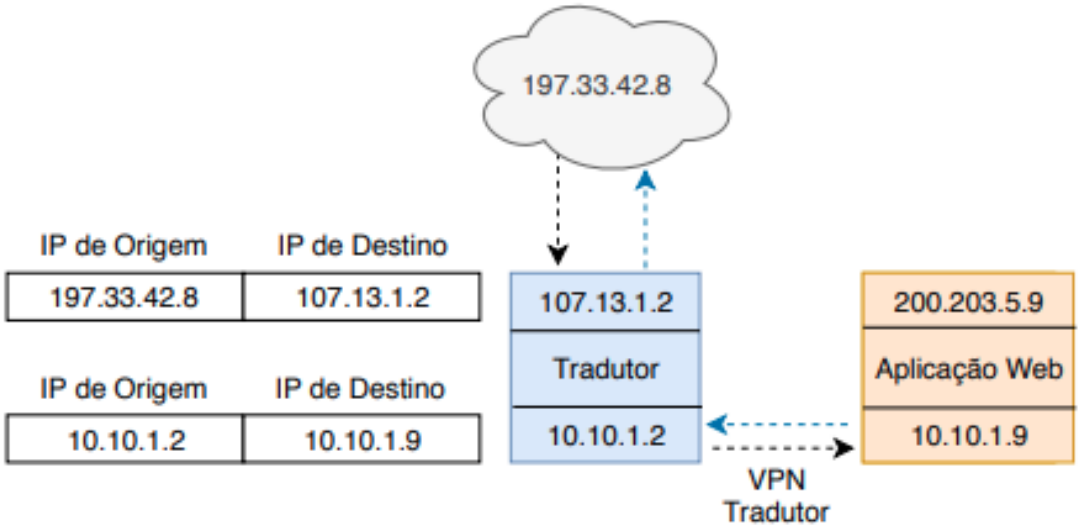
Quando a requisição chega até a aplicação web pela VPN, o Armazenador é atualizado informando que o cliente chegou ao seu destino corretamente. Essa atualização só é possível pois o IP de origem é enviado junto com a requisição. Sem esse cuidado, a aplicação web só possui a informação do IP traduzido pelo Tradutor e não consegue atualizar a informação no Armazenador.

Obedecendo também o tempo de 10 s, caso nenhuma requisição atinja a aplicação web, o Armazenador remove o fluxo permissivo no OVS do ASC, atualiza o estado do IP de origem para atacante e insere um fluxo de bloqueio no OVS do SL.

4.3 **Funcionamento do Sistema de Defesa**

O sistema proposto pode funcionar de dois modos distintos: um modo sem colaboração de outros ASs e outro modo com a colaboração de outros ASs. Cada um desses modos é descrito a seguir.

Figura 10 - Exemplo de funcionamento da estrutura do AS Colaborador.



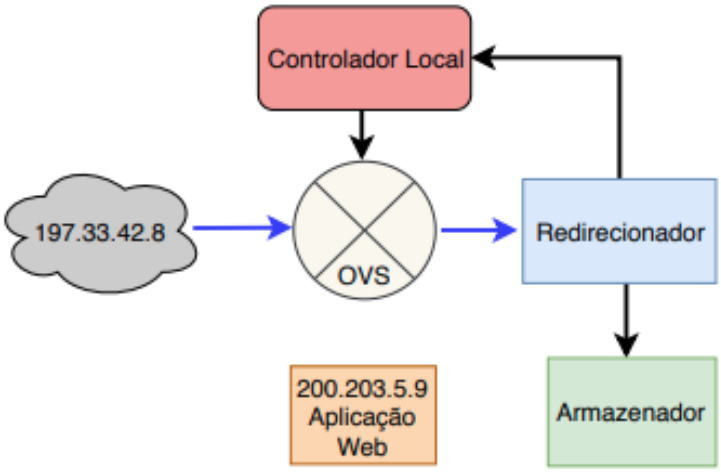
4.3.1 Funcionamento do sistema de defesa sem colaboração

O funcionamento do sistema de defesa proposto, sem colaboração de outros ASs, é ilustrado na Figura 11. Inicialmente, o OVS possui apenas um fluxo permissivo para que todo e qualquer endereço IP consiga acesso ao Redirecionador. Quando o Redirecionador recebe uma requisição qualquer da Internet, imediatamente essa requisição é cadastrada no Armazenador. O cadastro dessa nova entrada terá o campo estado com o valor não validado, ou seja, é uma entrada que ainda precisa ser verificada pelo sistema. Após esse cadastro, um fluxo permissivo no OVS é inserido para que o IP de origem dessa requisição seja capaz de acessar a aplicação web protegida pelo sistema, de acordo com a Figura 11a.

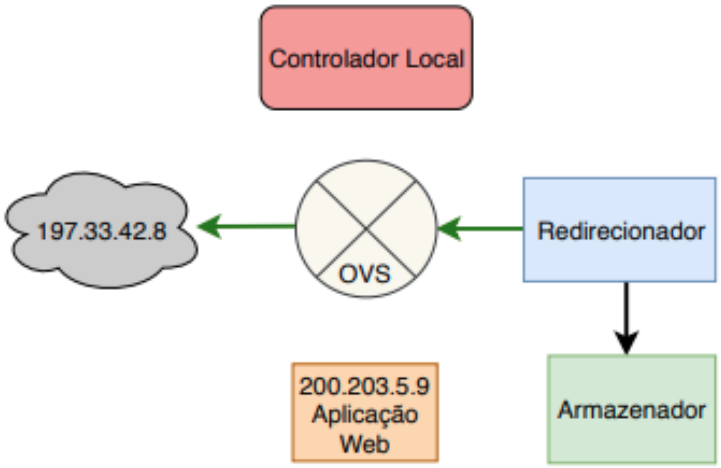
Um ponto importante a se observar é a atualização do campo estado no Armazenador. Caso uma requisição já esteja cadastrada no Armazenador com o valor do campo estado igual a 1 (validada), o Redirecionador apenas envia o redirecionamento para o cliente com destino ao IP cadastrado no campo aplicação. Assim, novas escritas de fluxos no OVS não são realizadas pois o fluxo permissivo para este cliente já foi escrito através de uma requisição prévia. O campo estado com valor 2 (ataque) é utilizado meramente para registro; afinal assim que o Redirecionador escreve um fluxo de bloqueio para um determinado IP e atualiza o campo estado para 2, esse IP não consegue mais realizar requisições para o Redirecionador. Dessa forma o seu cadastro no Armazenador não é mais atualizado.

Após as informações da requisição serem armazenadas e o fluxo permissivo inserido no OVS, com destino à aplicação web, um *redirect* 302 é enviado do Redirecionador para o solicitante, conforme mostrado na Figura 11b. Esse fluxo possui um *hard timeout* de 60 s. Esse tempo foi escolhido em função da latência percebida pelos clientes sem o sistema

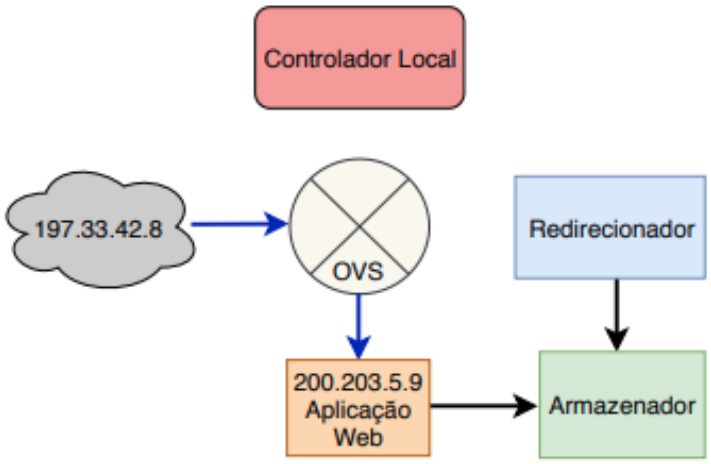
Figura 11 - Exemplo de funcionamento do sistema de defesa sem colaboração.



(a) Requisição do Cliente



(b) Acesso à Aplicação



(c) Acesso à Aplicação

atuante. É um intervalo de tempo 10 vezes maior que a média dessa latência e garante que, caso um cliente não confirme a sua legitimidade, o fluxo seja apagado da tabela do OVS.

Além de armazenar tráfegos, o Armazenador também monitora os dados armazenados. O Armazenador procura por requisições que tenham seu campo estado ainda não validado. Constantemente, todo campo estado com valor 0 tem seu tempo de escrita no Armazenador verificado através dos valores no campo tempo. Se em 10s a aplicação web não atualiza o campo estado para 1, o Armazenador assume que a requisição não foi capaz de seguir o redirecionamento enviado previamente, tratando-se muito provavelmente de uma requisição maligna. Após isso, o campo estado é atualizado com o valor 2 (atacante), um fluxo de bloqueio é inserido no OVS e o fluxo permissivo para a aplicação web é removido do OVS. Essa remoção de fluxo acontece em intervalos de tempo de 30 s, ou seja, menor que o tempo de *hard timeout* do fluxo inserido previamente. O *hard timeout* setado em 60 s garante que o fluxo nunca ficará na tabela em caso de falta de confirmação ou problema no sistema.

Apesar de estar fora do escopo do trabalho, é importante mencionar que a implementação de esquemas de revogação de proibição é possível. Essa revogação pode ser implementada de algumas maneiras. Como exemplo, é possível que os fluxos de bloqueio possuam um *idle timeout* setado com um valor de tempo predeterminado. Assim, caso o fluxo não seja utilizado durante esse tempo, o mesmo será removido. Outra opção de implementação seria classificar os IPs através de pontuação. Caso um IP seja classificado como suspeito, o mesmo recebe uma pontuação e um fluxo de bloqueio é inserido para esse IP. O *hard timeout* do fluxo de bloqueio inserido é configurado com determinado valor de tempo. O valor aumenta conforme a pontuação do IP observado aumente. Sendo assim, quanto mais suspeito mais tempo o IP fica bloqueado para o sistema.

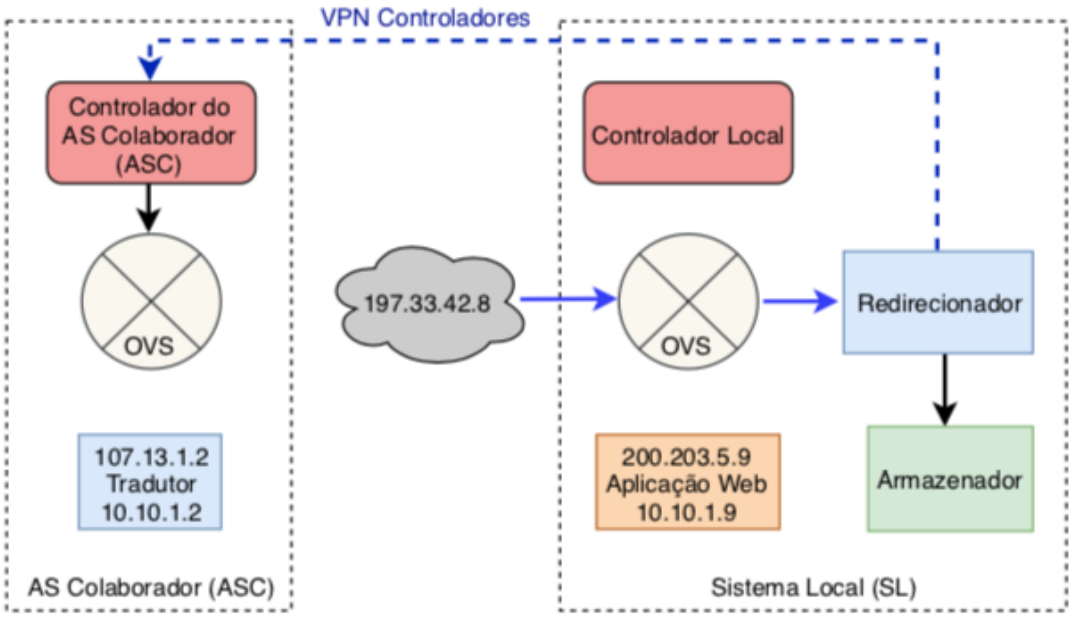
A Figura 11c mostra o solicitante seguindo as instruções do redirecionamento recebido do Redirecionador. Quando o solicitante acessa o conteúdo da aplicação web, o Armazenador tem o estado dessa requisição atualizado para validado. Em futuras requisições do mesmo IP, por se tratar de um IP já conhecido pelo sistema, o Redirecionador apenas redirecionará diretamente esse IP para o endereço da aplicação web. Esse endereço é definido no Armazenador, pelo campo aplicação, sem a necessidade de escrita de novos fluxos no OVS para este acesso.

4.3.2 Funcionamento do sistema de defesa com colaboração

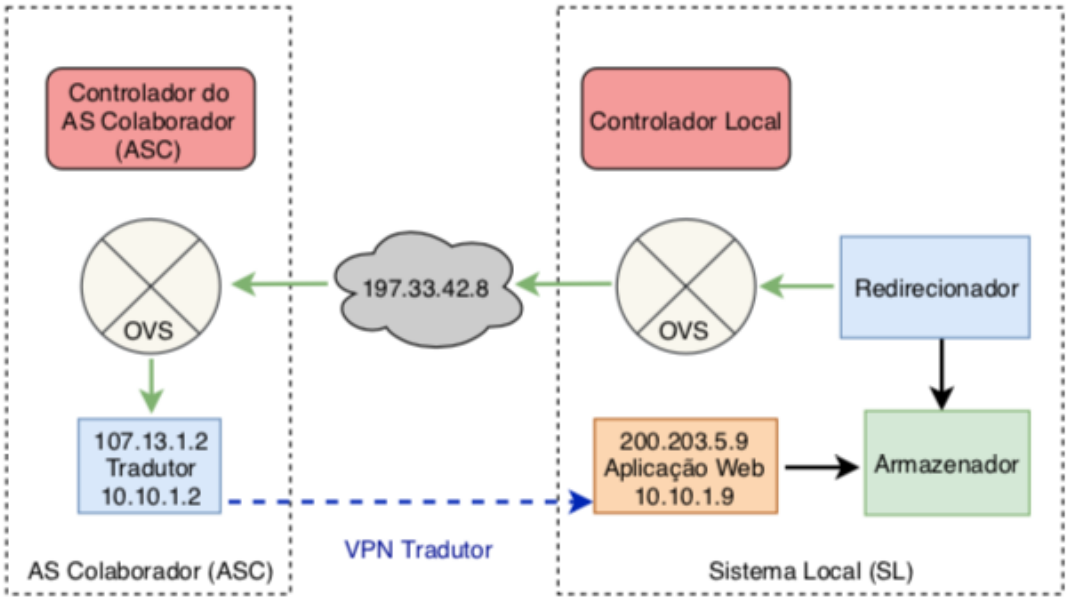
O funcionamento do sistema de defesa proposto, com colaboração de outros ASs, é ilustrado na Figura 12.

Como no modo sem colaboração, o OVS inicialmente possui apenas fluxos per-

Figura 12 - Exemplo de funcionamento do sistema de defesa com colaboração.



(a) Requisição do Cliente



(b) Acesso à Aplicação

missivos para que todo e qualquer endereço IP consiga acesso ao Redirecionador. O Redirecionador possui um conhecimento prévio dos IPs de destino de todas os IPS dos Tradutores em cada ASCs e de todos os controladores que atuam em cada ASC (além do IP da aplicação web e do Controlador Local). Dessa maneira, quando o Redirecionador recebe requisições, imediatamente para cada requisição, um cadastro é realizado no Armazenador com as informações referentes a cada requisição. Após o cadastro, cada requisição terá um fluxo permissivo escrito em algum OVS do sistema. A distribuição de endereços IPs por OVSs é feita através do algoritmo de balanceamento *round-robin* utilizado pelo Redirecionador.

O algoritmo de balanceamento *round-robin* é aplicado em diversas áreas como, por exemplo, na aplicação de preempção de processos em sistemas operacionais e também na transmissão e balanceamento de pacotes em redes de computadores. Para esta dissertação, o algoritmo foi utilizado para balanceamento de carga encaminhando uma solicitação para cada OVS por vez (HAHNE, 1986).

Após o IP ser cadastrado e o fluxo inserido, um *HTTP redirect* 302 é enviado para o solicitante, conforme ilustrado na Figura 12b. A partir do momento em que novos registros de IPs surgem no Armazenador, o Armazenador verifica quais entradas estão em estado ainda não validado por mais de 10 s. Caso alguma requisição fique no estado não validado por um tempo maior do 10 s, um fluxo de bloqueio é inserido no OVS e o fluxo permissivo para acesso à aplicação (seja à aplicação web local ou à aplicação de algum Tradutor no sistema) é removido.

O solicitante, seguindo as instruções do redirecionamento recebidas do Redirecionador nesse exemplo, procura o IP do Tradutor que encontra-se em outro AS na Internet. O fluxo permissivo para o acesso ao Tradutor já encontra-se escrito no OVS e assim, o Tradutor realizará uma tradução do endereço IP de origem e de destino (*Source* e *Destination NAT*) para IPs dentro da rede VPN que interliga os tradutores com o IP da aplicação web. Nesse caso a comunicação se dá entre os IPs 10.10.1.2 de origem e 10.10.1.9 de destino.

Quando o solicitante acessa o conteúdo da aplicação web, a aplicação web atualiza o estado desse IP no Armazenador para validado. Com essa atualização, o fluxo permissivo ficará escrito no OVS escolhido pelo Redirecionador. Em caso de futuras requisições pelo mesmo IP, o Redirecionador apenas redirecionará diretamente esse IP para o endereço de destino registrado no Armazenador sem a necessidade de escrita de fluxos no OVS através do controlador.

5 AVALIAÇÃO DE DESEMPENHO DO SISTEMA PROPOSTO

O desempenho do sistema proposto é avaliado por meio de três experimentos. No primeiro experimento, o consumo de CPU da aplicação web e a latência das requisições para a aplicação web são medidas sem a atuação do sistema. No segundo experimento, o funcionamento do sistema é avaliado sem a colaboração de outro AS para lidar com o tráfego que chega no sistema. O consumo de CPU do controlador local, o consumo de CPU da aplicação web e a latência das requisições para a aplicação web são medidos tanto em casos de ataque quanto em casos de fluxo normal de funcionamento (sem ataque). O terceiro experimento realiza as mesmas medidas do segundo experimento, mas com o sistema funcionando com a ajuda dos ASCs.

5.1 Cenários

Todos os experimentos foram realizados utilizando um MacBook Pro com 16 GB de RAM e um processador Intel(R) Core(TM) i5-3210 CPU @ 2,50 GHz. O sistema proposto foi testado no Mininet (MININET, 2019), um sistema de emulação de redes de código aberto, escrito em Python, projetado para trabalhar com ambientes SDN. O ambiente configurado para os testes foi montado no Virtual-Box 6.0 (VIRTUALBOX, 2019) com um SL e até seis ASCs. Devido às limitações do Mininet, foi necessário separar em diferentes máquinas virtuais (*Virtual Machines* - VMs) cada um dos controladores, o Redirecionador, a aplicação web a ser protegida e o Armazenador. Assim, foram utilizadas 12 VMs para a realização dos experimentos, listadas a seguir:

- sete VMs para os controladores;
- uma VM para o Armazenador;
- uma VM para o Redirecionador;
- uma VM para a aplicação web;
- uma VM para o ambiente Mininet;
- uma VM para o simulador de *botnet* Bonesi.

Todas as VMs foram instaladas com o sistema operacional Ubuntu 16.04 LTS. As máquinas que hospedaram os controladores possuem, cada uma, 512 MB de RAM com uma interface virtual de rede de 1 Gbit/s. As VMSs referentes ao Armazenador, à aplicação web a ser protegida e ao simulador de *botnet* Bonesi possuem, cada uma,

1 GB de RAM com uma interface virtual de rede de 1 Gbit/s. A VM alocada para o Redirecionador possui 2 GB de RAM com uma interface de 1 Gbit/s, enquanto que a VM que hospeda o ambiente Mininet possui 2 GB de RAM com uma interface de 1 Gbit/s.

O controlador utilizado neste trabalho foi o Ryu (RYU, 2019) por uma questão de simplicidade e praticidade; afinal, o seu código fonte é escrito em Python (PYTHON, 2019). O Python, além de ser usado pela API do Mininet, também foi a linguagem utilizada em todos os códigos neste trabalho[1]. Além disso, o controlador Ryu possui um bom desempenho quando comparado com outros controladores mais populares, como por exemplo o POX, o FloodLight e o OpenDaylight (KHONDOKER et al., 2014).

Os experimentos foram realizados em três cenários. No primeiro cenário, as requisições eram realizadas diretamente ao servidor web. No segundo cenário, as requisições foram realizadas para o sistema atuando sem colaboração. No terceiro cenário, as requisições foram realizadas para o sistema atuando com colaboração. Em todos os cenários, as requisições foram realizadas em situações de ataque e em situações sem ataque.

Para avaliar o comportamento do sistema proposto, 50 clientes enviam requisições HTTP benignas para o Redirecionador. A quantidade de requisições enviadas por cada cliente foi escolhida aleatoriamente, através da função *random.randint(x,y)* do *Python*. Essa função trabalha com distribuição uniforme entre os valores de x e y que, para esse experimento, os valores utilizados foram 5 e 50 respectivamente. A sequência de requisições realizadas entre esses clientes também utilizou a mesma função. Com a escolha definida das requisições e da sequência dos clientes, essa informação foi salva para que todo o experimento fosse realizado com a mesma configuração. Isso é necessário para que a comparação entre os cenários propostos possam ser avaliados em função de um referencial comum.

Em situações sem ataques, apenas as requisições benignas são enviadas em todos os modos de operação pelos clientes e os resultados dessas simulações coletados. Em situações com ataques, em conjunto com as requisições benignas, uma *botnet* simulada através do simulador Bonesi (BONESI, 2019), ataca os alvos definidos em cada modo de operação. Os ataques utilizados nos experimentos são do tipo HTTP flood, taxa de 100 requisições/s. A escolha do número de requisições do ataque foi baseada no comportamento da latência sem o sistema atuando na proteção da aplicação web. Com valores maiores que 100 requisições/s não havia retorno das conexões, tornando impossível registrar os valores da latência. As requisições HTTP benignas foram enviadas com o objetivo de acessar a aplicação web protegida seguindo o redirecionamento enviado pelo Redirecionador (*status code* 302) pois, ao contrário de uma *botnet*, essas são capazes de seguir as instruções de redirecionamento do protocolo HTTP.

[1] Os códigos estão disponíveis em https://github.com/dstelman/sistema.git

De forma prática a latência percebida pelos clientes foi medida através do cálculo do tempo entre o envio da requisição e o retorno da mesma para cada cliente.

O consumo de CPU foi medido através da função psutil do *Python* que fornece o percentual de consumo da CPU a cada segundo em uma VM. Os valores foram registrados em intervalos de 1 s em cada experimento.

5.2 **Resultados**

A seguir o desempenho da aplicação web sem a proteção do sistema proposto é analisado. Seus resultados são usados como base para comparação com os resultados obtidos pelo sistema proposto. Posteriormente os dois cenários com o sistema em funcionamento são avaliados com todos os seus modos de operação. Os valores apresentados em todas as figuras de resultados são médias com intervalos de confiança de 95%.

5.2.1 Comportamento da aplicação web sem a atuação do sistema

O cenário com a aplicação web sem o sistema proposto atuante mostra o quanto o desempenho da aplicação web, percebida pelos clientes, pode piorar em situações de ataque em comparação com o seu desempenho quando apenas lida com tráfego benigno. Para este caso, a Figura 13 mostra a variação da latência percebida pelos clientes da aplicação web quando a mesma lida com tráfego benigno e com tráfego de ataque. Analisando o gráfico, nota-se que a latência aumenta de alguns milissegundos para 6 s quando há ataque; ou seja, há um aumento significativo na latência percebida pelos clientes. Ainda nesse cenário, de acordo com a Figura 14, o consumo de CPU da VM da aplicação web quando atacada dobra em relação ao consumo da mesma CPU em uma situação sem ataque. Esse comportamento é esperado, já que o objetivo de um ataque DDoS é deixar uma aplicação inacessível para os usuários finais.

5.2.2 Comportamento do sistema atuando sem colaboração

O objetivo desse cenário é verificar o quanto o sistema é capaz de melhorar a latência das requisições e verificar a queda no consumo de CPU da aplicação a ser protegida. Além disso, o consumo de CPU do Controlador Local deve ser verificado e levado em consideração pois, caso a CPU do Controlador Local sature, o sistema entra em colapso não atendendo novas requisições dos usuários legítimos.

A Figura 15 apresenta a latência percebida pelos clientes quando o sistema traba-

Figura 13 - Latência percebida pelos clientes com a aplicação web desprotegida.

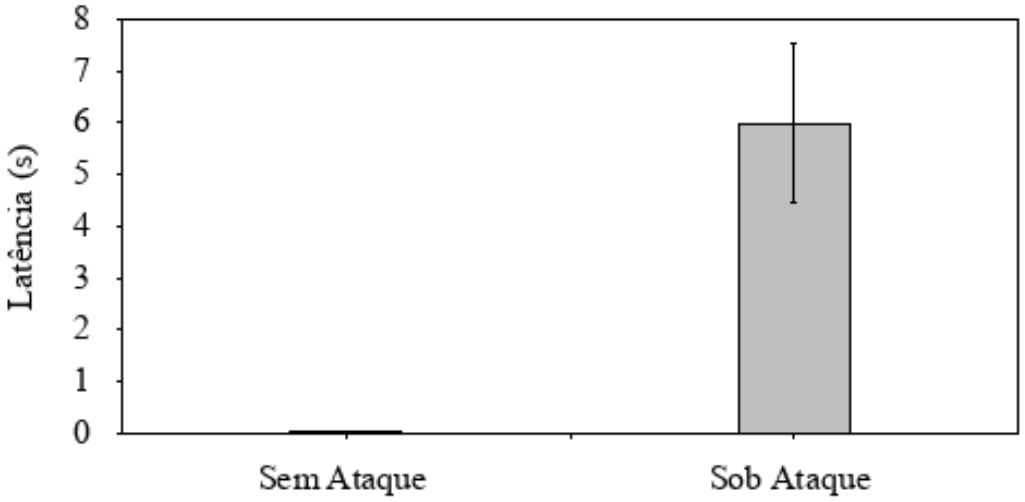


Figura 14 - Consumo de CPU da VM da aplicação web desprotegida.

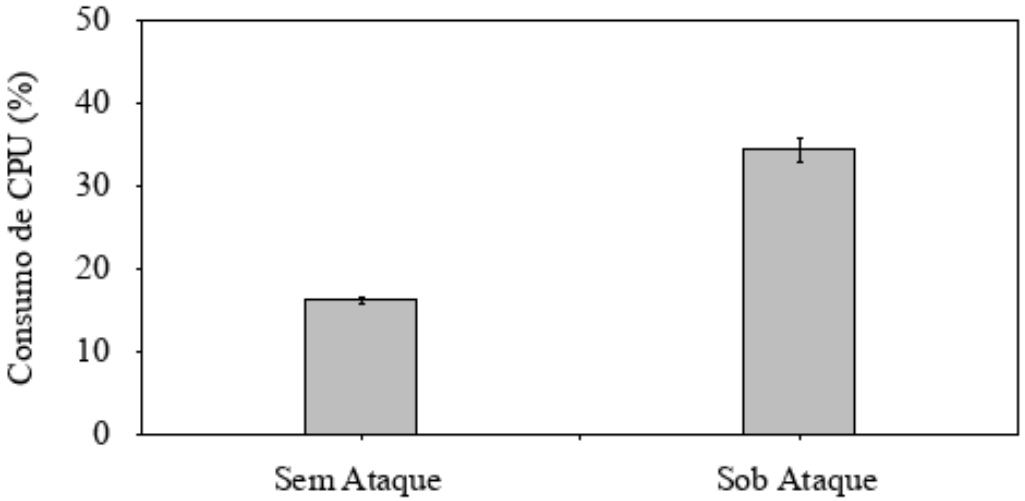
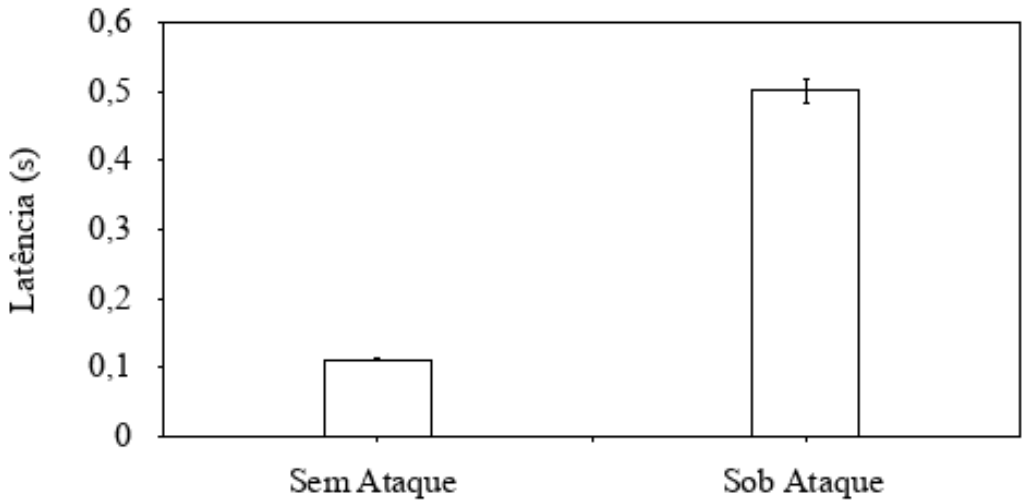


Figura 15 - Latência percebida pelos clientes com o sistema sem colaboração.

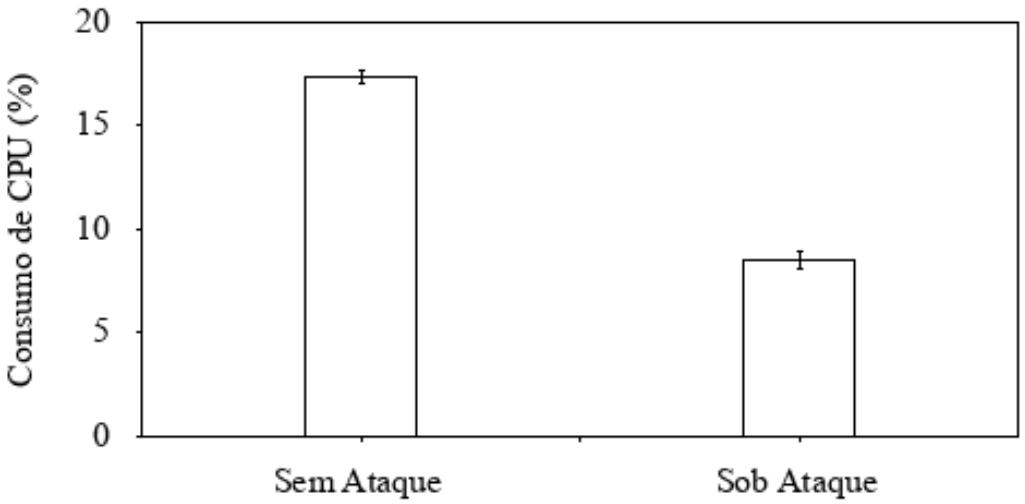


lha sem colaboração. Nota-se, em comparação com a Figura 13, um aumento da latência em situações sem ataque quando comparada com a latência do sistema desprotegido. Isso se deve ao fato de um maior processamento ser imposto pelo sistema quando requisições internas são realizadas tanto ao Armazenador quanto ao Controlador Local. Essas requisições exigem um tempo maior de conclusão dessas ações, principalmente na consulta ao Redis do Armazenador, antes do envio do redirecionamento para o solicitante. Esse tempo é todo refletido na latência do sistema para o usuário final. A latência nesse caso subiu de 40 para 111 ms. Esse aumento, em termos práticos, é imperceptível para o usuário final, de acordo com o estudo de (NNGROUP, 2019).

A Figura 15 também apresenta uma queda de latência das requisições válidas durante um ataque, em comparação com os valores mostrados na Figura 13. A queda de 6 para 0,5 s mostra que o sistema é capaz de diminuir significativamente o tempo de resposta das requisições feitas por usuários válidos durante uma situação de ataque.

A Figura 16 apresenta o consumo de CPU da VM da aplicação web com o sistema atuando sem colaboração. Em comparação com os valores obtidos na Figura 14, nota-se uma queda de 80% no consumo da CPU quando o sistema é atacado. Esse valor ficou abaixo dos 17,38% registrados na situação sem ataque. Como nenhuma requisição maliciosa chega até a aplicação web, com o sistema atuando sem colaboração, as mesmas não são processadas e a CPU na aplicação web continua a ser utilizada somente para tratar as requisições válidas, que agora não chegam mais continuamente. Também é de se notar que o consumo de CPU da VM da aplicação web aumentou quando o sistema de proteção atuando sem colaboração é utilizado no caso em que não existem ataques. Esse aumento é explicado através das requisições realizadas pela aplicação web para o

Figura 16 - Consumo de CPU da VM da aplicação web sem colaboração.



Armazenador que não existiam quando a aplicação web não era protegida pelo sistema.

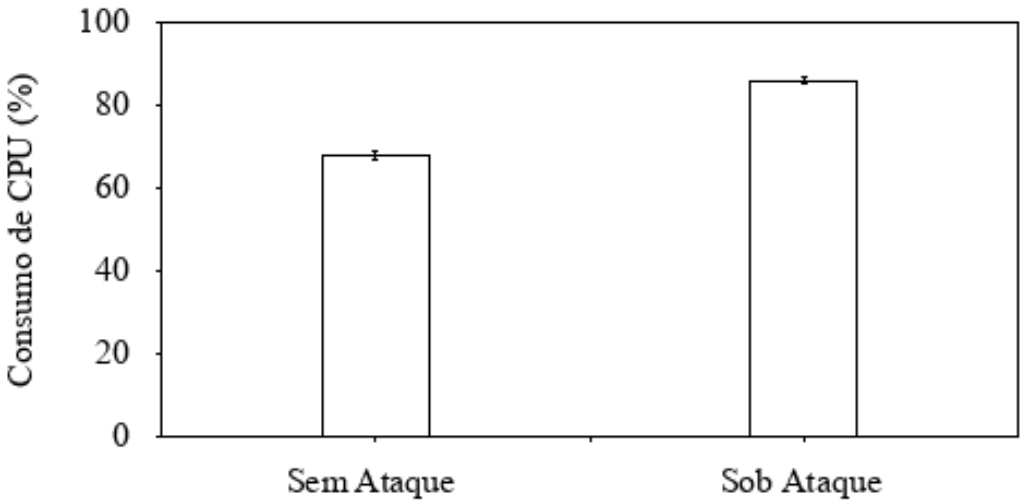
A Figura 17 apresenta o consumo de CPU da VM do Controlador Local em situações de ataque e de tráfego benigno. Nota-se que o consumo da CPU da VM do Controlador Local no caso de ataque chegou a 85,68%. Um ataque maior do que o utilizado nos experimentos pode fazer com que a CPU alcance toda a sua capacidade de processamento levando o sistema ao colapso. Em caso de colapso, o Redirecionador não consegue inserir novos fluxos no OVS através do Controlador Local. Isso impede que novos clientes tenham acesso à aplicação desejada e que ataques futuros possam ser tratados. Também verifica-se, através da Figura 17 que o consumo de CPU da VM do Controlador Local quando o sistema não é atacado é de 67% da sua capacidade.

Esse cenário mostrou que o sistema é eficaz na redução da consumo de CPU da VM da aplicação web. Além disso, o sistema é capaz de reduzir a latência percebida pelos usuários podendo ser utilizado como uma opção contra ataques do tipo HTTP flood em uma aplicação web na Internet.

5.2.3 Comportamento do sistema atuando com colaboração

O objetivo desse cenário é verificar o quanto o sistema é capaz de melhorar a latência das requisições e verificar a queda no consumo de CPU da VM da aplicação web a ser protegida quando ASs colaboradores são utilizados. Além disso, o consumo de CPU da VM do Controlador Local deve ser verificado e levado em consideração pois, caso essa CPU sature, o sistema entra em colapso não atendendo novas requisições dos usuários

Figura 17 - Consumo de CPU da VM do Controlador Local sem colaboração.

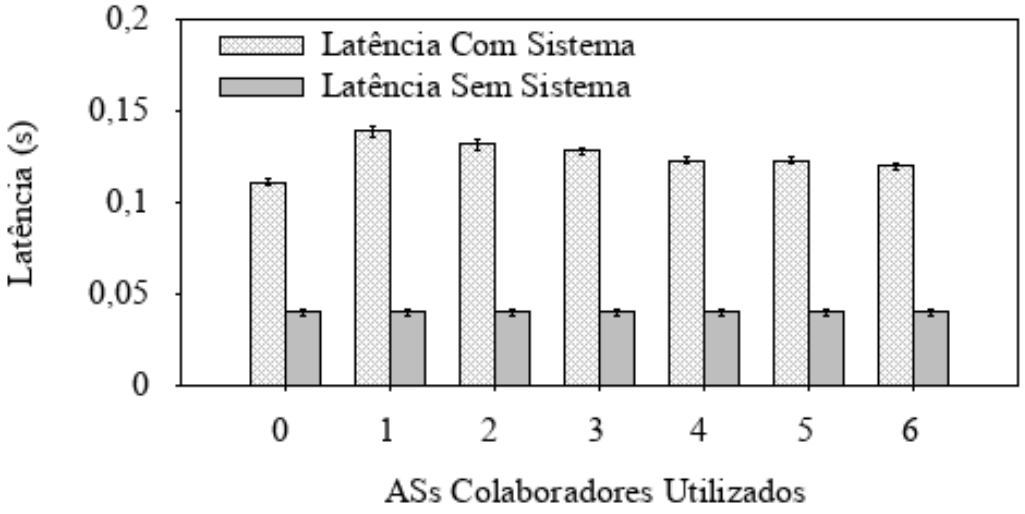


legítimos. No cenário considerado, os ASs colaboradores atuam para a ajudar o sistema, com o objetivo de obter uma redução no consumo de CPU da VM do Controlador Local.

A Figura 18 apresenta a latência percebida pelos clientes legítimos em situações sem ataque para um número de ASs colaboradores de 1 a 6. Para efeito de comparação a figura também contém os valores de latência quando não há colaboração (valor 0) apresentados anteriormente na Figura 15. É de se observar que a latência aumenta quando o sistema passa a proteger a aplicação web. Isso é justificado pelas novas interações realizadas pelo sistema (antes inexistentes) que são os acessos ao Redis e às APIs dos controladores pelo Redirecionador. Além das novas interações, a latência também aumenta em função dos redirecionamentos para outros ASs. Nota-se que a latência aumenta quando comparada com o sistema sem colaboração. Esse aumento se deve ao NAT realizado nos pacotes em cada ASC que ajuda a aumentar o tempo de latência percebido pelos clientes que, em termos práticos, é um aumento imperceptível para o usuário final, de acordo com o estudo de (NNGROUP, 2019). Ao comparar os valores de latência com os diferentes números de ASCs, é percebida uma queda de latência a medida que o número de ASCs aumenta. A partir do quarto ASC, os valores encontrados são estatisticamente iguais.

A Figura 19 contém os resultados referentes à latência percebida pelos clientes legítimos em situações de ataque. Na figura encontra-se também o valor da latência para o sistema operando sem colaboração para fins comparativos (ASC com valor 0). Como já observado na Figura 13 o valor da latência é de 6 s durante um ataque sem o sistema e o sistema sem colaboração faz com que o tempo de latência seja de 0,5 s. Os resultados mostram a queda da latência das requisições válidas durante um ataque conforme o número de ASCs aumenta. A partir da utilização do quarto AS colaborador

Figura 18 - Latência percebida pelos clientes para diferentes números de ASs colaboradores no caso sem ataque.



os valores das latências são estatisticamente iguais. Como o Redirecionador possui um limite de processamento, não se percebe novas quedas de latência caso mais ASCs sejam inseridos no sistema.

A Figura 20 apresenta o consumo de CPU da VM da aplicação web com o sistema atuando com colaboração e sem ataque. Também são apresentados os valores obtidos em situações em que a aplicação web atua sem o sistema proposto de proteção. Além disso, os valores do consumo de CPU da VM da aplicação web com o sistema atuando sem colaboração também são representados nesse gráfico para fins comparativos. Em relação à CPU da VM da aplicação web, os resultados obtidos são estatisticamente iguais aos resultados apresentados na Seção 5.2.2 pelos mesmos motivos. Também nota-se o aumento do consumo de CPU da aplicação web quando o sistema de proteção atua com colaboração. Esse aumento também é obtido pelos mesmos motivos apresentados na Seção 5.2.2.

A Figura 21 apresenta o consumo de CPU da VM com a aplicação web tendo o sistema atuando com colaboração e em situação de ataque. Também são mostrados os valores obtidos em situações em que a aplicação web atua sem o sistema proposto de proteção e sem colaboração, para fins comparativos. Em relação à CPU da VM com a aplicação web, os resultados obtidos mostram uma queda significativa de valor similar com quaisquer quantidades de AS colaboradores em relação à aplicação sem o sistema de proteção. Ao se comparar com o cenário sem colaboração, era de se esperar um consumo parecido afinal somente os tráfegos legítimos alcançam a aplicação web protegida pelo sistema.

Figura 19 - Latência percebida pelos clientes para diferentes números de ASs colaboradores no caso de ataque.

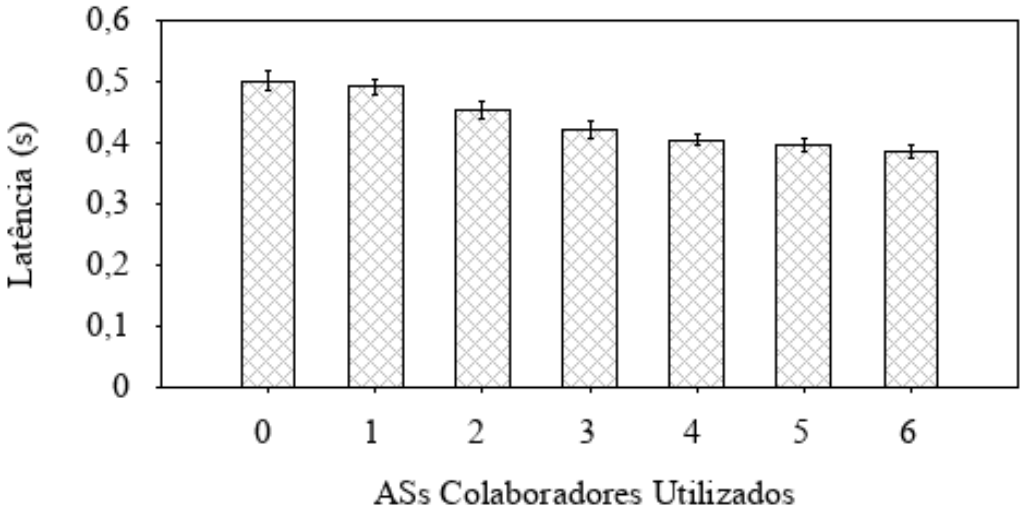


Figura 20 - Consumo de CPU da VM aplicação web no caso sem ataque.

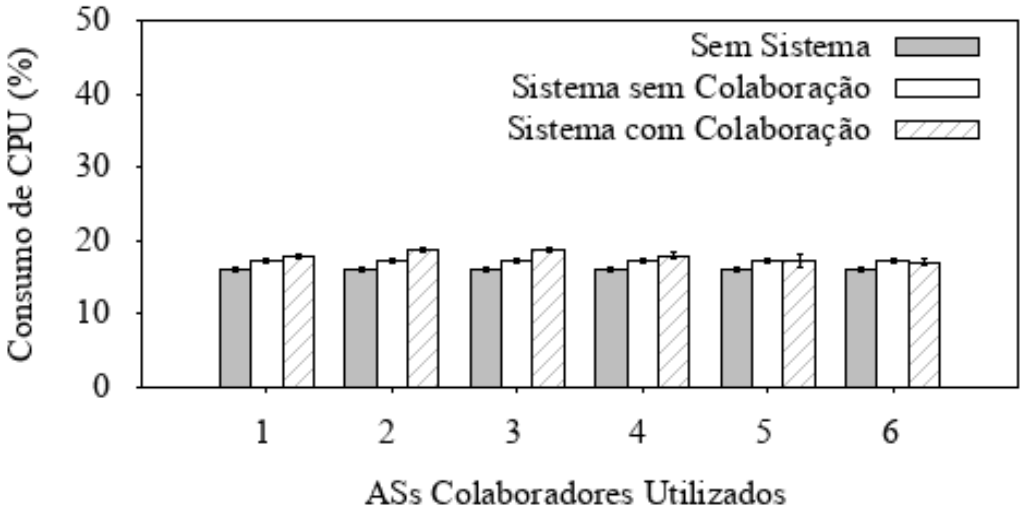
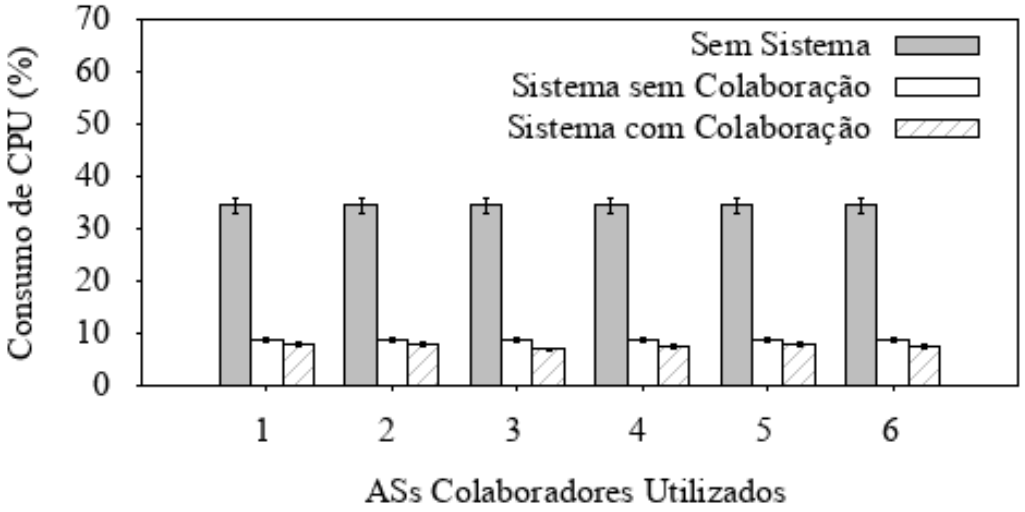


Figura 21 - Consumo de CPU da VM aplicação web no caso de ataque.

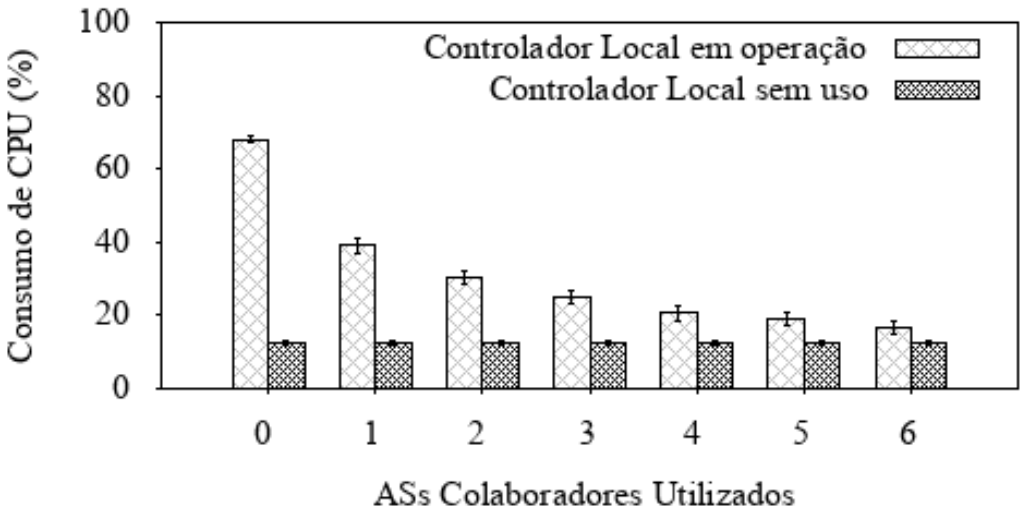


A Figura 22 apresenta o consumo de CPU da VM do Controlador Local do sistema em situações sem ataque. O consumo de CPU do Controlador Local diminui conforme o número de ASs colaboradores inseridos no sistema aumenta. Nota-se que com seis ASCs o consumo de CPU fica próximo ao consumo mínimo registrado quando o Controlador Local não encontra-se em uso. Esse consumo mínimo é o quanto de CPU da VM do Controlador Local é consumida apenas para o Controlador se manter em funcionamento. A queda no consumo de CPU do Controlador Local em relação ao cenário sem colaboração faz com que o sistema não se aproxime da saturação e só foi possível através da redistribuição de tráfego entre cada ASC e o SL. Outro detalhe, mostrado na Tabela 2, é que o somatório dos consumos das CPUs de todos controladores dos ASCs é elevado. Porém, como um dos objetivos do sistema é evitar o colapso do Controlador Local, esse consumo é aceitável e esperado. Um ataque DDoS demanda muito processamento para ser tratado e, como todos os recursos de hardwares são limitados, os ASCs ajudam o SL também em termos de processamento. Por limitações de *hardware*, não foi possível aumentar o números de ASCs para que fosse possível verificar uma provável queda, ainda mais acentuada, do consumo da CPU do Controlador Local.

Tabela 2 - Consumo de CPU dos Controladores do Sistema sem Ataque.

SL	ASC 1	ASC 2	ASC 3	ASC 4	ASC 5	ASC 6
67,71 ± 1,17 %	-	-	-	-	-	-
39,11 ± 2,11 %	42,03 ± 2,10 %	-	-	-	-	-
30,37 ± 1,96 %	29,97 ± 1,98 %	29,50 ± 1,98 %	-	-	-	-
24,90 ± 1,85 %	25,02 ± 1,84 %	24,91 ± 1,91 %	23,81 ± 1,78 %	-	-	-
20,55 ± 1,90 %	19,09 ± 1,81 %	17,95 ± 1,72 %	22,50 ± 1,99 %	18,44 ± 1,74 %	-	-
19,17 ± 1,96 %	18,75 ± 2,00 %	18,03 ± 1,92 %	16,97 ± 1,86 %	18,48 ± 1,97 %	18,91 ± 1,94 %	-
16,51 ± 1,77 %	16,33 ± 1,77 %	17,79 ± 1,90 %	17,48 ± 1,85 %	16,84 ± 1,81 %	18,81 ± 2,03 %	15,93 ± 1,75 %

Figura 22 - Consumo de CPU da VM do Controlador Local sem ataque.



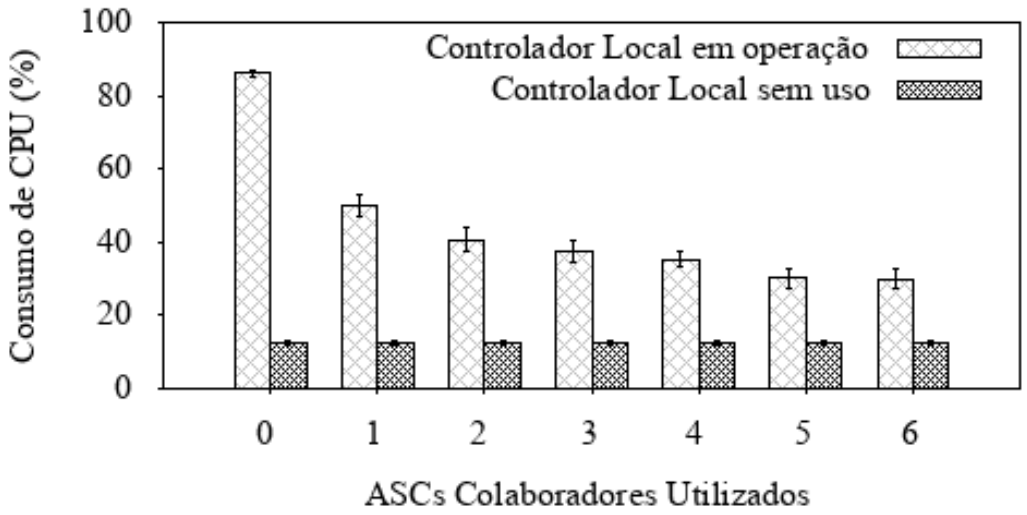
A Figura 23 mostra o consumo de CPU da VM do Controlador Local em situações de ataque. De acordo com a Figura 23, o consumo de CPU do Controlador Local baixa em função do aumento de ASCs colaboradores inseridos no sistema. Nota-se que, em situações de ataque, o consumo de CPU não fica próximo ao consumo mínimo registrado quando o Controlador Local não encontra-se em uso. Apesar da queda não se aproximar tanto desse valor, a colaboração é válida pois as quedas são acentuadas. A queda no consumo de CPU do Controlador Local em relação ao cenário sem colaboração faz com que o sistema não se aproxime da saturação e só foi possível através da redistribuição de tráfego entre cada ASC e o SL. Outro detalhe, mostrado na Tabela 3, é que o somatório dos consumos das CPUs de todos controladores dos ASCs é elevado. Porém, pelos mesmos motivos mostrados na Tabela 2, os valores se justificam.

Tabela 3 - Consumo de CPU dos Controladores do Sistema com Ataque.

SL	ASC 1	ASC 2	ASC 3	ASC 4	ASC 5	ASC 6
86,08 ± 0,96 %	-	-	-	-	-	-
50,08 ± 2,97 %	51,86 ± 2,93 %	-	-	-	-	-
40,67 ± 3,25 %	47,22 ± 3,67 %	35,24 ± 2,88 %	-	-	-	-
37,30 ± 2,95 %	40,90 ± 3,60 %	28,06 ± 2,28 %	29,48 ± 2,33 %	-	-	-
35,41 ± 2,96 %	31,38 ± 3,44 %	24,27 ± 2,10 %	25,38 ± 2,21 %	25,05 ± 2,23 %	-	-
30,16 ± 2,66 %	28,36 ± 3,25 %	24,65 ± 2,18 %	22,37 ± 1,95 %	23,33 ± 2,04 %	20,81 ± 1,83 %	-
29,85 ± 2,73 %	30,25 ± 3,33 %	21,25 ± 1,93 %	18,58 ± 1,76 %	19,77 ± 1,83 %	18,06 ± 1,63 %	18,32 ± 1,62 %

Este cenário avaliado mostra que o sistema com cooperação é capaz de atuar contra ataques do tipo HTTP Flood e melhora seu desempenho se comparado ao cenário sem colaboração. Isso acontece pois o sistema com colaboração em termos de latência percebida pelos clientes e consumo de CPU da VM aplicação web, possui ganhos. A principal vantagem de se utilizar a colaboração é obter uma redução do consumo da CPU

Figura 23 - Consumo de CPU da VM do Controlador Local com ataque.



do Controlador Local, como mostrado nas Figuras 22 e 23. Isso se torna um artifício importante ao evitar o colapso do sistema como um todo. Essa redução é um dos desafios no tratamento de ataques DDoS pois o custo de processamento de requisições exigido do sistema alvo é alto. Além disso, dependendo da magnitude de um ataque sobre um sistema alvo, utilizar a capacidade de processamento de terceiros (ASCs) mostrou-se uma solução viável para esse tipo de problema.

6 CONCLUSÃO

Esta dissertação propôs um sistema de proteção baseado em SDN para uma aplicação web contra ataques de *botnets* do tipo HTTP flood. O sistema tem como objetivo preservar o Controlador da rede que contém a aplicação web a ser protegida, denominado Controlador Local, por meio da colaboração com outros sistemas localizados em outros sistemas autônomos, denominados ASs Colaboradores. Além disso, a latência percebida pelos clientes durante situações de ataque também é reduzida ao se utilizar o sistema de proteção. O sistema também visa não permitir o aumento do consumo de CPU da aplicação web, pois ataques do tipo HTTP flood podem conseguir saturar a CPU do servidor web tornando o ataque eficaz ao deixar o sistema indisponível para os usuários.

O sistema atua em dois modos, sendo que, no primeiro modo não existe colaboração de outras redes SDN (ASCs - ASs Colaboradores) e no segundo modo, ASs Colaboradores ajudam a absorver o tráfego de entrada. O sistema foi implementado e foi realizada uma avaliação de desempenho do sistema. Além disso, a utilização de VPNs para a comunicação entre o Sistema Local e os ASs Colaboradores foi escolhida por questão de confiabilidade e simplicidade. O sistema assume que todo o tráfego que passa pelos tuneis de VPN são confiáveis não necessitando de tratamento para os mesmos. Dessa forma, o administrador do sistema "não precisa"se preocupar com roteamento e segurança nesses casos. Outro ponto importante é que em uma visão macro do sistema, apesar de um AS Colaborador estar em outro ponto da internet, em termos lógicos, a VPN acaba por aproximar esses ASs do Sistema Local como se fossem redes locais.

Os resultados mostraram que para evitar a perda do Controlador Local, a distribuição do tráfego ajuda a baixar o consumo de CPU do mesmo, evitando assim o colapso do sistema. Além disso, percebe-se que o sistema é capaz de diminuir a latência para requisições legítimas durante um ataque em ambos os modos de funcionamento. Com a utilização de mais ASCs a tendência da latência é diminuir.

O consumo de CPU da aplicação web, em situações sem colaboração, mostrou-se baixo durante um ataque porém isso se deve ao fato da latência ser maior do que em situações sem ataque no sistema. Como somente requisições válidas chegam até a aplicação web, o intervalo entre as mesmas é maior exigindo menos da CPU. Além disso, algumas requisições previamente bloqueadas pelo sistema não chegam até a aplicação web. Em situações com colaboração, o consumo permaneceu o mesmo em relação ao sistema sem colaboração, conforme esperado; afinal, os tráfegos maliciosos nas duas situações (com e sem colaboração) não chegam até a aplicação web.

Nota-se que o desempenho em função de latência e do consumo de CPU da VM da aplicação protegida foi diferente nos modos colaborativo e sem colaboração, porém com ganhos em ambos os casos. Em função da queda do consumo de CPU da VM do

Controlador Local, a medida que ASs colaboradores foram incluidos no sistema, os valores registrados como resultado dessas inclusões foram significativos para cada AS colaborador. Em situações de ataque, a partir do quinto AS Colaborador, os resultados foram estatisticamente iguais. Para situações sem ataque, isso ocorreu a partir do quarto AS Colaborador. Nota-se nos resultados uma tendência de queda nesses valores a medida que novos ASs Colaboradores são incluídos no sistema. Como o sistema não possui limitação no número de ASs Colaboradores, é possível simular o sistema com mais ASs Colaboradores para observar essa queda mais detalhadamente.

Para trabalhos futuros algumas modificações podem ser implementadas para um melhor desempenho do sistema. Como exemplo, é possível tratar as requisições de entradas com vários *workers* em paralelo para que mais requisições possam ser atendidas em menos tempo, evitando assim filas e congestionamentos no processamento dessas requisições. Na mesma linha de raciocínio, além de *workers* em paralelo, também seria possível virtualizar o Redirecionador através de servidores em paralelo, evitando assim que apenas uma única máquina processe todas as requisições que chegam até o Redirecionador. Com essa implementação, seria esperado um ganho de latência ainda maior do que o obtido para o usuário final, pois as requisições não chegariam enfileiradas em um único destino.

Também é possível implementar o Redirecionador com a capacidade de mudar algumas de suas características quando necessário. Esse módulo poderá utilizar, por exemplo, a tecnologia de Funções Virtuais de Rede para a mudar o seu IP rapidamente na Internet. Isso evitaria que o sistema se tornasse um alvo fácil de varredura por atacantes por não mais se tratar de um IP estático exposto na Internet. Neste trabalho utilizou-se um IP estático para o Redirecionador, pois a intenção era mostrar a possibilidade de distribuição do tráfego entre ASs Colaboradores e o acesso ao serviço protegido para clientes legítimos diretamente através de uma VPN. Além disso, com um túnel criptografado ligando os pontos do sistema, o número de saltos dos pacotes são menores do que se esses pacotes fossem enviados pela Internet normalmente. Caso se utilize IPs dinâmicos, os ataques contra o sistema seriam menos eficazes, pois toda a *botnet* teria que ser modificada para utilizar novos IPs definidos pelo sistema.

Um outro ponto que pode ser abordado como melhoria desse projeto no futuro é a aplicação do conceito de *threat intelligence*. Esse conceito permite bloquear os IPs suspeitos de uma forma mais inteligente na qual os IPs seriam classificados não apenas como maliciosos ou não maliciosos, mas através de níveis de periculosidade. Esses níveis são determinados pelo próprio sistema através de cálculos estatísticos para cada IP e os bloqueios seriam realizados em função do peso que cada IP possui determinado por requisições prévias. Isso evitaria que falsos positivos e requisições perdidas fossem bloqueados permanentemente. A ideia básica desse conceito consiste em quanto menor a sua periculosidade, menos tempo um IP fica bloqueado no sistema. Caso contrário,

o IP fica bloqueado mais tempo ou, dependendo dos limites estabelecidos, bloqueado permanentemente.

REFERÊNCIAS

A. BARBOSA A., F. E. F. M. M. T. C. Y. P. *Software Defined Networks*. 2019. https://www.gta.ufrj.br/grad/16_2/2016sdn/seguranca.html.

ABD, H. A. L. P.; PHD, X. T. Information security labs in ids/ips for distance education. In: ACM. *Proceedings of the 7th conference on Information technology education*. [S.l.], 2006. p. 47–52.

ALSHAMRANI, A. et al. A defense system for defeating ddos attacks in sdn based networks. In: ACM. *Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access*. [S.l.], 2017. p. 83–92.

ARBOR. *ARBOR TMS*. 2018. https://www.arbornetworks.com/ddos-protection-products/arbor-tms.

ARBOR. *Proven, comprehensive threat protection and service enablement*. 2018. https://www.arbornetworks.com/images/documents/data%20sheets/ds_tms_en.pdf.

BARBOSA, R. R. *Migração de redes tradicionais para SDN*. Dissertação (Mestre) — Universidade de São Paulo, São Paulo, 2018.

BONESI. *Bonesi*. 2019. https://github.com/markus-go/bonesi.

BRITO, A. V. R. I. V. S. *Internet do Futuro e Programabilidade da Rede - uma visão prática de SDN/OpenFlow e P4*. 2019. https://www.lacnic.net/innovaportal/file/3207/1/programabilidaderede-sdn-p4-lacnog2018.pdf.

CERT.BR. *Honeypots e Honeynets: Definições e Aplicações*. 2020. https://www.cert.br/docs/whitepapers/honeypots-honeynets/.

CISCO. *Why is intent based networking good news for software defined networking*. 2019. https://blogs.cisco.com/analytics-automation/why-is-intent-based-networking-good-news-for-software-defined-networking.

DAO, N.-N. et al. A feasible method to combat against ddos attack in sdn network. In: IEEE. *2015 International Conference on Information Networking (ICOIN)*. [S.l.], 2015. p. 309–311.

DRIDI, L.; ZHANI, M. F. Sdn-guard: Dos attacks mitigation in sdn networks. In: IEEE. *Cloud Networking (Cloudnet), 2016 5th IEEE International Conference on*. [S.l.], 2016. p. 212–217.

DRIDI, L.; ZHANI, M. F. A holistic approach to mitigating dos attacks in sdn networks. *International Journal of Network Management*, Wiley Online Library, v. 28, n. 1, p. e1996, 2018.

DURNER, R. et al. Detecting and mitigating denial of service attacks against the data plane in software defined networks. In: IEEE. *2017 IEEE Conference on Network Softwarization (NetSoft)*. [S.l.], 2017. p. 1–6.

FAJAR, A. P.; PURBOYO, T. W. A survey paper of distributed denial-of-service attack in software defined networking (sdn). *International Journal of Applied Engineering Research*, v. 13, n. 1, p. 476–482, 2018.

FLASK. *Flask*. 2019. http://flask.palletsprojects.com/en/1.1.x/.

G. REIS M., D. R. S. R. J. R. R. *Sistema de Detecção de Intrusão*. [S.l.], 2019. https://www.gta.ufrj.br/grad/16_2/2016ids/index.html.

GOOGLE. *Google Cloud Platform for Data Center Professionals: Networking*. 2018. https://cloud.google.com/docs/compare/data-centers/networking.

HAHNE, E. L. *Round robin scheduling for fair flow control in data communication networks*. [S.l.], 1986.

Hameed, S.; Khan, H. A. Leveraging sdn for collaborative ddos mitigation. In: *2017 International Conference on Networked Systems (NetSys)*. [S.l.: s.n.], 2017. p. 1–6.

HARRIS, B.; KONIKOFF, E.; PETERSEN, P. Breaking the ddos attack chain. *Institute for Software Research*, 2013.

HUANG, Q.; KOBAYASHI, H.; LIU, B. Analysis of a new form of distributed denial of service attack. In: *Proc. 37th Ann. Conf. Information Science and Systems (CISS'03)*. [S.l.: s.n.], 2003.

INSTITUTE, P. *The Cost of Denial-of-Services Attacks*. Michigan, 2015.

JAFARIAN, J. H.; AL-SHAER, E.; DUAN, Q. Openflow random host mutation: transparent moving target defense using software defined networking. In: ACM. *Proceedings of the first workshop on Hot topics in software defined networks*. [S.l.], 2012. p. 127–132.

KHONDOKER, R. et al. Feature-based comparison and selection of software defined networking (sdn) controllers. In: IEEE. *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. [S.l.], 2014. p. 1–7.

KIM, J.; SHIN, S. Software-defined honeynet: Towards mitigating link flooding attacks. In: IEEE. *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. [S.l.], 2017. p. 99–100.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *arXiv preprint arXiv:1406.0440*, 2014.

LIM, S. et al. A sdn-oriented ddos blocking scheme for botnet-based attacks. In: IEEE. *Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on*. [S.l.], 2014. p. 63–68.

LTD., R. *DDoS Attack Definitions - DDoSPedia*. 2020. https://security.radware.com/ddos-knowledge-center/ddospedia/http-challenge/.

LUKASEDER, T. et al. An extensible host-agnostic framework for sdn-assisted ddos-mitigation. In: IEEE. *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. [S.l.], 2017. p. 619–622.

MAHAJAN, R. et al. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM Computer Communication Review*, ACM, v. 32, n. 3, p. 62–73, 2002.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008.

MININET. *Mininet*. 2019. https://github.com/mininet/mininet/wiki/documentation.

MIRKOVIC, J.; REIHER, P. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, ACM, v. 34, n. 2, p. 39–53, 2004.

NETSCOUT. *What is DDoS*. 2020. https://www.netscout.com/what-is-ddos.

NNGROUP. *Powers of 10: Time Scales in User Experience*. 2019. https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux.

PATEL, A.; QASSIM, Q.; WILLS, C. A survey of intrusion detection and prevention systems. *Information Management & Computer Security*, Emerald Group Publishing Limited, v. 18, n. 4, p. 277–290, 2010.

PYTHON. *Python*. 2019. https://www.python.org/.

RASHIDI, B.; FUNG, C. Cofence: a collaborative ddos defence using network function virtualization. In: IEEE. *Network and Service Management (CNSM), 2016 12th International Conference on*. [S.l.], 2016. p. 160–166.

REDIS. *Redis 3.0.6*. 2019. https://redis.io/.

RYU. *Ryu*. 2019. https://osrg.github.io/ryu/.

SAHAY, R. et al. Towards autonomic ddos mitigation using software defined networking. In: . [S.l.: s.n.], 2015.

SAHAY, R. et al. Adaptive policy-driven attack mitigation in sdn. In: *Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures*. New York, NY, USA: ACM, 2017. (XDOMO'17), p. 4:1–4:6. Disponível em: ⟨http://doi.acm.org/10.1145/3071064.3071068⟩.

SCOTT-HAYWARD, S.; O'CALLAGHAN, G.; SEZER, S. Sdn security: A survey. In: IEEE. *2013 IEEE SDN For Future Networks and Services (SDN4FNS)*. [S.l.], 2013. p. 1–7.

SEZER, S. et al. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine*, IEEE, v. 51, n. 7, p. 36–43, 2013.

SILVA, E. d. C. d. *Detecção de ataques de negação de serviço utilizando aprendizado de máquina*. Dissertação (Mestre) — Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2016.

SINGH, D. et al. Modelling software-defined networking: switch design with finite buffer and priority queueing. In: IEEE. *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. [S.l.], 2017. p. 567–570.

SNORT. *Snort*. 2019. https://snort.org.

SWAMI, R.; DAVE, M.; RANGA, V. Software-defined networking-based ddos defense mechanisms. *ACM Computing Surveys (CSUR)*, ACM, v. 52, n. 2, p. 28, 2019.

VERMA, A.; XAXA, D. K. A survey on http flooding attack detection and mitigating methodologies. *International Journal of Innovations & Advancement in Computer Science*, v. 5, n. 5, p. 18–21, 2016.

VIRTUALBOX. *VirtualBox*. 2019. https://www.virtualbox.org.

WANG, H.; XU, L.; GU, G. Floodguard: A dos attack prevention extension in software-defined networks. In: IEEE. *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. [S.l.], 2015. p. 239–250.

WOLFF, M. *O que é SDN e como essa arquitetura de rede funciona*. 2019. https://penseemti.com.br/artigos/o-que-e-sdn-e-como-essa-arquitetura-de-rede-funciona/.

YATAGAI, T.; ISOHARA, T.; SASASE, I. Detection of http-get flood attack based on analysis of page access behavior. In: IEEE. *2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. [S.l.], 2007. p. 232–235.

ZHANG, G.; PARASHAR, M. Cooperative mechanism against ddos attacks. In: *Security and Management*. [S.l.: s.n.], 2005. p. 86–96.