Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

DoSSEC: proposta de detecção e mitigação de ataques SYN Flood em redes SDN

Ranyelson Neres Carvalho

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Jacir Luiz Bordim

Brasília
2020

Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

Esta página não deve ser inclusa na versão final do texto.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

DoSSEC: proposta de detecção e mitigação de ataques SYN Flood em redes SDN

Ranyelson Neres Carvalho

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Jacir Luiz Bordim (Orientador)
CIC/UnB

Prof. Dr. Eduardo Adilio Pelinson Alchieri
Universidade de Brasília

Prof. Dr. João José Costa Gondim
Universidade de Brasília

Profª. Drª. Genaina Nunes Rodrigues
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 11 de Fevereiro de 2020

Dedicatória

Dedico este trabalho à minha família e meus amigos que me ajudaram a chegar mais longe.

Agradecimentos

Agradeço primeiramente a Deus por ter me ajudado nessa caminhada, a minha família por todo apoio. Agradeço em especial a minha namorada Míriam, pela paciência e compreensão nos momentos difíceis, que não foram poucos. Agradeço ao Programa de Pós-Graduação em Informática da Universidade de Brasília, aos professores da banca e ao Departamento de Ciência da Computação por todo suporte e esclarecimentos. Agradeço aos colegas do laboratório, Paulo, Luís B., Marcos, Jeferson, Guilherme D., Guilherme E., Luís A., Leomar, Wittenberg, Walter e em especial ao Lucas por sempre procurar extrair o meu melhor e está sempre disposto a me ajudar sem medir esforços. Agradeço aos meus professores da Universidade de Brasília, em especial ao professor Jacir Luiz Bordim, pelas orientações, conselhos, confiança e paciência, que tiveram grande importância para a obtenção dos resultados.

Resumo

As redes definidas por *software* (SDN) representam uma nova arquitetura de rede que fornece controle central sobre a rede. Ela é caracterizada pela separação entre o plano de dados e o plano de controle, o que define um ambiente programável. No plano de controle, o controlador permite a execução de serviços que definem as políticas de controle e as distribui no plano de dados por meio de um protocolo padrão, como o OpenFlow. Apesar dos inúmeros benefícios fornecidos por essa arquitetura, a segurança em redes SDN ainda é um importante motivo de preocupação. Em particular, os ataques de negação de serviço (DoS) desafiam as arquiteturas SDN de várias maneiras, devido às vulnerabilidades existentes entre os planos. Este trabalho propõe uma solução de segurança denominada DoSSEC, composta por dois módulos (detecção e mitigação) que atuam diretamente no plano de dados. O primeiro módulo fornece como saída as alterações no volume do tráfego que indicam um possível ataque em andamento, por meio das estatísticas geradas pelos *switches* programáveis. O segundo módulo tem como objetivo priorizar os fluxos já validados por meio de um método baseado em reputação, que utiliza filas nos *switches* para denotar o nível de confiança dos fluxos, preservando e priorizando um conjunto benigno do tráfego. Os resultados experimentais obtidos mostram que, o DoSSEC reduz o tempo de resposta em média 30% em comparação com outra solução do estado da arte. Além disso, o DoSSEC melhora a taxa de conexões TCPs realizadas com sucesso e diminui o congestionamento de fluxos, reduzindo os impactos causados pelo ataque DoS (SYN *Flood*).

Palavras-chave: SDN, redes definidas por *software*, segurança, DoS, plano de dados, SYN *Flood*

Abstract

Software-defined networks (SDN) represent a new network architecture that provides central control over the network. It is characterized by the separation between data plane and control plane, which defines a programmable environment. In the control plane, the controller allows the execution of services that define control policies and distributes them in the data plane through a standard protocol, such as OpenFlow. Despite numerous benefits provided by this architecture, security on SDN networks is still an important concern. In particular, denial of service (DoS) attacks challenge SDN architectures in several ways, due to the vulnerabilities that exist between the plans. This work proposes a security solution called DoSSEC, composed of two modules (detection and mitigation) and acts directly on the data plane. The first module provides as output the changes in the volume of traffic that indicate a possible attack in progress, through the statistics generated by the programmable switches. The second module aims to prioritize flows that have already been validated through a reputation-based method, which uses queues at the switches to denote the level of confidence of the flows, preserving and prioritizing a benign set of traffic. The experimental results obtained show that DoSSEC reduces the response time by an average of 30% compared to another state-of-the-art solution. In addition, DoSSEC improves the rate of successful TCP connections and reduces flow congestion, reducing the impacts caused by the DoS (SYN Flood) attack.

Keywords: SDN, software defined networking, security, DoS, data plane, SYN Flood

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AG Algoritmos Genéticos.

AINA Advanced Information Networking and Applications.

APDCM Advances in Parallel and Distributed Computational Models.

API Application Programming Interface.

ASIC Application Specific Integrated Circuits.

ASTESJ Advances in Science, Technology and Engineering Systems Journal.

DDoS Distributed Denial of Service.

DNS Domain Name System.

DoS Denial of Service.

DoSSEC Denial of Service Security.

HTTP HyperText Transfer Protocol.

IDS Intrusion Detection System.

IP Internet Protocol.

MAC Media Access Control.

NOS Networking Operating System.

NTP Network Time Protocol.

OFX OpenFlow Extension Framework.

P4 Programming Protocol Independent Packet Processors.

PISA Protocol-Idenpendent Switch Architecture.

QoS Quality of Service.

REST Representational State Transfer.

RNA Rede Neural Artificial.

RPC Remote Procedure Call.

ToS Type of Service.

Capítulo 1

Introdução

O paradigma de Redes Definidas por *Software* (SDN) foi proposto como um meio para remediar as limitações impostas pela arquitetura de uma rede tradicional, tendo em vista sua arquitetura complexa, políticas inconsistentes e dependência de fornecedor [6]. A rede SDN promove uma nova maneira de gerenciar a rede por meio de uma separação lógica entre o plano de controle e o plano de dados, de modo a oferecer um ambiente programável.

Em uma rede SDN o plano de controle é responsável por realizar todo o gerenciamento da rede, no qual é encarregado de criar funcionalidades para realizar as decisões operacionais de como a rede deverá se comportar [7]. Para isso, faz uso das informações fornecidas pelo plano de dados. O plano de dados compreende a infraestrutura de rede [1]. Ele é responsável pelo encaminhamento e descarte de pacotes na rede. A comunicação entre os dois planos é realizada através de um protocolo, a implementação mais difundida é o protocolo OpenFlow, que define um método genérico para permitir que o controlador interaja com os dispositivos de encaminhamento da rede [8]. Porém, tal separação torna mais difícil garantir a segurança da rede, uma vez que o comportamento geral da rede é definido por uma combinação de configurações entre os dois planos [9].

Um dos principais desafios de segurança de uma rede SDN são os ataques de negação de serviço, devido as fragilidades existentes entre o plano de controle e o plano de dados. Este ataque pode ocorrer nos dois planos, com o objetivo de exaurir a capacidade de processamento do controlador ou reduzir a largura de banda da aplicação, por meio de um grande volume de tráfego gerado em um espaço curto de tempo [10] [11].

1.1 Motivação

Este trabalho aborda a detecção e mitigação de ataques de negação de serviço (DoS) volumétrico do tipo SYN *Flood* em uma rede SDN, que a princípio podem servir como

base para a detecção e mitigação de ataques de negação de serviços distribuída (DDoS). Os ataques DoS/DDoS volumétricos do tipo SYN *Flood*, têm como objetivo tornar indisponível uma máquina ou recursos de rede a seus usuários, através de um extenso volume de tráfego enviado a vítima [11]. Este tipo de ataque em uma rede SDN, pode causar danos tanto no plano de dados quanto no plano de controle.

Soluções como SDN-Guard [12], Slicots [13] e Operetta [14] procuram reduzir os efeitos causados por um ataque SYN *Flood*, adicionando métodos de proteção junto ao controlador. Embora, estas soluções sejam capazes de mitigar os efeitos causados por esse ataque, ainda são necessárias medidas para reduzir o gargalo de comunicação entre o plano de dados e controle, o que pode ocasionar uma sobrecarga ao canal de comunicação e ao controlador e causar ataques de saturação na rede.

Para atenuar este problema, Shin *et al.* [15] propuseram o Avant-Guard. A solução adiciona um método de proteção contra o ataque SYN *Flood* no plano de dados, no qual encaminha apenas conexões validadas ao seu destino, protegendo assim o controlador de ataques de saturação e reduzindo o gargalo de comunicação entre os planos. No entanto, a solução ainda é vulnerável a uma variação do mesmo tipo de ataque, no qual busca-se exaurir a capacidade de gerenciamento dos dispositivos de encaminhamento que compõem o plano de dados.

1.2 Justificativa

A ausência de soluções que possam mitigar e realizar ações para distinguir o tráfego legítimo do tráfego espúrio em uma rede SDN, podem deixar a rede suscetível a ataques de saturação aos dispositivos de encaminhamento e ao controlador, como ataques de negação de serviço [16]. Este tipo de ataque aproveita o longo atraso na comunicação entre os planos (dados e controle), causando inundação no canal de comunicação entre o *switch* e o controlador, impedindo que fluxos legítimos possam ser atendidos pelo controlador e assim acarretando em uma saturação na rede [15].

O desenvolvimento de soluções que atuem no plano de dados e que sejam capazes de detectar e mitigar ataques de negação de serviço em uma rede SDN, promovem a redução no custo de comunicação e atrasos que a interação entre o plano de dados e controle impõe sobre a rede SDN [17]. Estas soluções podem garantir a escalabilidade e resiliência da rede, de modo a não impor uma carga prejudicial ao plano de controle ou no plano de dados e assim tornar a rede mais segura e estável, promovendo um pré-processamento no plano de dados para que o plano de controle tome as decisões operacionais [18].

1.3 Objetivo geral

O objetivo geral deste trabalho é fornecer uma solução de detecção e mitigação de ataques DoS do tipo volumétrico em redes SDN, que protege o alvo e a infraestrutura SDN, por meio da priorização de fluxos legítimos por meio de um método baseado em reputação, que utiliza filas nos dispositivos que compõem o plano de dados para denotar o nível de confiança dos fluxos. Desta forma, eles terão prioridades mais alta e serão movidos para uma fila apropriada, enquanto que os demais serão deslocados para filas com prioridades menores. Aqueles que tiverem comportamento fora dos padrões estabelecidos serão bloqueados, de modo a reduzir o congestionamento no acesso à rede e a comunicação entre os planos e assim, minimizar os impactos causados pelo ataque SYN *Flood* na rede.

1.3.1 Objetivos específicos

- Projetar uma arquitetura no plano de dados que permite o uso de diferentes técnicas de análise estatísticas para a detecção de ataques DoS;
- Identificar/validar os métodos de mitigação mais adequados para uso em redes SDN;
- Projetar uma solução de mitigação de ataques DoS em redes SDN que priorize fluxos legítimos diretamente no plano de dados;
- Analisar o desempenho da solução desenvolvida em diversos cenários, comparando-a com outra abordagem encontrada na literatura.

1.4 Metodologia

A metodologia aplicada neste trabalho consistirá inicialmente de uma revisão do estado da arte acerca de ataques de negação de serviço, redes SDN e soluções relacionadas à segurança em redes SDN, com ênfase no ataque SYN *Flood*. Em seguida, serão avaliadas as técnicas existentes de detecção e mitigação, seus benefícios e limitações no que tange aos objetivos traçados. Com base nestas avaliações, será definido mecanismos alternativos que possibilitem reduzir os impactos causados pelo ataque SYN *Flood* em uma rede SDN. A solução proposta será avaliada por meio de simulação. Para tal, utilizaremos um simulador conhecido e amplamente aceito pela comunidade científica de forma a permitir a comparação e validação com propostas semelhantes na literatura. Os resultados, após validados, servirão de indicativo para eventuais melhorias e ajustes.

1.5 Contribuições

Em particular as principais contribuições desta dissertação são as seguintes:

- Proposta de uma solução que atua no plano de dados para reduzir os impactos causados pelo ataque SYN *Flood* em uma rede SDN, baseado em um método de reputação para atribuir o nível de confiança aos fluxos;
- Arquitetura no plano de dados que permite o uso de diferentes técnicas de análise estatísticas para a detecção de ataques DoS, por meio de uma integração entre o *switch* programável e um módulo para o cálculo de diferentes funções estatísticas;
- Redução do tempo de resposta para obter uma página *web* em média 30% para um ambiente sob ataque em comparação com outra solução da literatura;

Os resultados parciais de tais contribuições foram:

- Artigo publicado e apresentado no *21st Workshop on Advances in Parallel and Distributed Computational Models (APDCM)-2019*, intitulado "*Entropy-based DoS attack identification in SDN*" [19] - Qualis CC B4.
- Artigo aceito para ser publicado na revista *Advances in Science, Technology and Engineering Systems Journal (ASTESJ)-2020*, intitulado "*Enhancing an SDN architecture with DoS attack detection mechanisms*" [20].
- Artigo aceito para ser apresentado no *34th Advanced Information Networking and Applications (AINA)-2020*, intitulado "*New programmable data plane architecture based on P4 OpenFlow Agent*" [21] - Qualis CC A2.

1.6 Estrutura do Documento

Este documento está organizado da seguinte forma:

- **Capítulo 2:** apresenta uma fundamentação teórica acerca dos conceitos relacionados para o desenvolvimento do presente trabalho, como ataques de negação de serviço, métodos de detecção de ataques, redes SDN e suas definições básicas;
- **Capítulo 3:** apresenta uma revisão do estado da arte acerca dos trabalhos relacionados a segurança em redes SDN, com ênfase no ataque SYN *Flood*;
- **Capítulo 4:** apresenta a solução de detecção e mitigação desenvolvida;
- **Capítulo 5:** apresenta os resultados obtidos da pesquisa;

- **Capítulo 6:** apresenta a conclusão do trabalho e indica os possíveis trabalhos a serem desenvolvidos no futuro.

Capítulo 2

Fundamentação Teórica

Neste capítulo são abordados temas relacionados à segurança em redes, como os principais ataques de negação de serviço presentes em redes de arquitetura tradicional e os mecanismos de detecção de ataques. Também se aborda redes SDN, bem como o principal protocolo para a comunicação entre o plano de dados e controle, os principais controladores para o desenvolvimento de uma rede SDN, o recente conceito de plano de dados programável, segurança em SDN e as plataformas para a realização de simulações de redes SDN. A Seção 2.1 apresenta os principais conceitos relacionados à segurança em redes de arquitetura tradicional e a Seção 2.2 os conceitos e definições acerca de SDN.

2.1 Segurança em Redes

Ao longo dos anos, com a popularização da Internet devido ao número de serviços oferecidos que as utilizam, o termo segurança se tornou um requisito indispensável para sua utilização. Cada vez mais, os serviços demandam privacidade, e isso implica que os dados precisam ser transferidos e armazenados de forma segura. Associado com esse rápido crescimento, o número de ataques também apresentou um alto crescimento, causando prejuízos significativos aos usuários desse serviço [22].

Diversos ataques foram desenvolvidos para as mais diferentes aplicações na Internet, eles procuram de alguma forma, explorar as vulnerabilidades existentes dos protocolos de comunicação, com o propósito de roubar informações ou simplesmente tornar um serviço indisponível [23]. Como resultado, diversos mecanismos de detecção foram desenvolvidos para tornar a Internet um ambiente mais seguro e confiável [24].

Um dos ataques mais conhecidos no âmbito da comunidade de segurança em redes, são os ataques de negação de serviço que visam causar indisponibilidade dos serviços oferecidos por aplicativo ou serviço [25]. Logo, se faz necessário o investimento no desenvolvimento

de mecanismos de segurança que sejam capazes de detectar e assim propor meios para reduzir os impactos causados por esse tipo de ataque.

Voltado para este cenário, as subseções a seguir apresentam a arquitetura dos principais ataques de negação de serviço e os mecanismos de detecção que possibilitam identificar esse tipo de ataque em uma rede.

2.1.1 Ataques de Negação de Serviço

Os ataques de negação de serviço (do inglês, *Denial of Service* - DoS) visam esgotar os recursos do dispositivo ou da infraestrutura para causar indisponibilidade de serviço [11]. Uma evolução dos ataques DoS, são os ataques de negação de serviço distribuído (do inglês, *Distributed Denial of Service* - DDoS), que utilizam os mesmos princípios dos ataques DoS, porém o ataque é realizado de maneira coordenada por vários dispositivos comprometidos que atacam um alvo simultaneamente.

Os ataques DoS/DDoS têm evoluído desde mecanismos que realizavam inundações simples para mecanismos mais complexos que aproveitam as características de diversas aplicações e protocolos para exaurir os recursos das vítimas. Porém, muitos ataques possuem características similares que permitem realizar uma classificação. Eles podem ser classificados em duas grandes categorias: ataques volumétricos e não volumétricos (de baixo volume e baixa taxa) [26].

Ataques Volumétricos

Os ataques volumétricos são caracterizados pelo envio de grandes volumes de tráfego com o objetivo de sobrecarregar algum recurso da vítima, normalmente, a banda disponível ou sua capacidade de processamento, para que as solicitações legítimas não sejam atendidas [27]. Eles podem ser subdivididos em ataques diretos e ataques de reflexão.

Nos ataques diretos, os *hosts* comprometidos enviam um grande volume de tráfego diretamente para a vítima. Os ataques mais comuns para esta categoria são SYN *Flood*, UDP *Flood* e ICMP *Flood*.

O ataque SYN *Flood* consiste em enviar um grande número de solicitações de conexão TCP [28], com endereços de origem falsificados, na forma de segmentos SYN para um servidor [29]. A Figura 2.1 demonstra o funcionamento do ataque. O atacante envia um grande volume de solicitações de conexão (SYN) a um servidor, com endereços de origem falsificados. Em seguida, o servidor responde a cada mensagem recebida com um segmento SYN-ACK e aguarda o segmento ACK para estabelecer a conexão. Como o endereço de origem do pacote SYN é falso, a resposta ACK nunca chegará porque o SYN-ACK foi enviado para o endereço falsificado, o que faz com que a conexão fique em

um estado semi-aberto (SYN-RECV) e os recursos de memória não são liberados. Assim, com um grande número de segmentos SYN sendo enviados, é possível preencher a fila de conexão TCP e, eventualmente, causar falhas no servidor de maneira que ele não consiga responder às solicitações normais do cliente TCP.

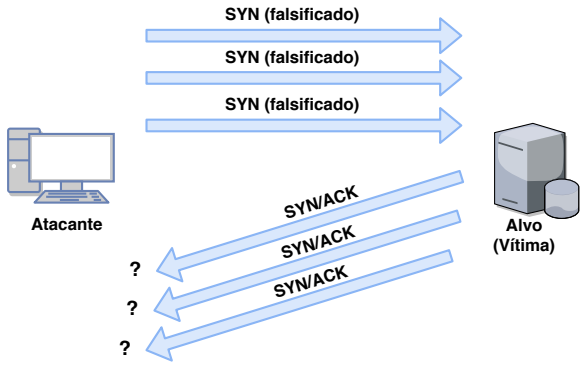


Figura 2.1: Ataque SYN Flood

O ataque UDP *Flood* emprega o uso do protocolo UDP, em que não é necessário estabelecer uma conexão para transferir dados [23]. A Figura 2.2 demonstra como o ataque é realizado. Neste ataque, o atacante envia um grande número de pacotes UDP para uma vítima especificando uma porta aleatória na máquina de destino. Quando o *host* de destino recebe um pacote UDP, ele determinará o que está escutando na porta de destino, se nada estiver escutando, ele retornará um pacote ICMP para o endereço IP de origem falsificado, notificando que a porta de destino é inacessível.

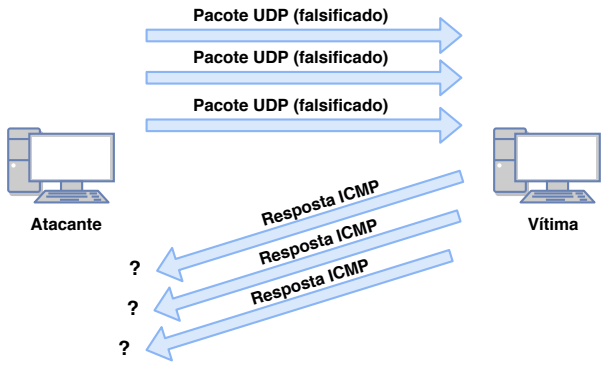


Figura 2.2: Ataque UDP Flood

Se um grande número de pacotes UDP forem enviados para portas inativas no *host* de destino, não apenas o *host* de destino ficará inacessível, mas os computadores no mesmo segmento também serão desativados devido à grande quantidade de tráfego. Por se tratar

de um protocolo em que nenhuma configuração de conexão é necessária antes que os dados sejam transferidos, podem ser difíceis de prevenir e detectar o ataque, a única característica do fluxo malicioso é seu grande volume.

O ataque ICMP *Flood* é similar ao UDP, mas utiliza pacotes ICMP para criar as inundações [23]. A Figura 2.3 demonstra como o ataque é realizado. Ele consiste no envio de uma grande quantidade de pacotes do tipo *ICMP-ECHO-REQUEST* ("*ping*") para o *host* de destino, de maneira que ele não conseguirá responder com rapidez suficiente para aliviar a quantidade de tráfego na rede.

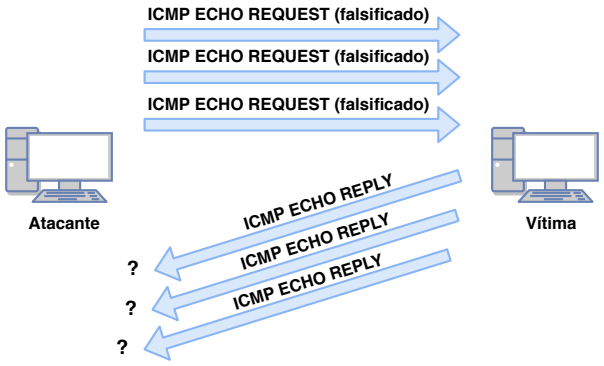


Figura 2.3: Ataque ICMP flood

Nos ataques de reflexão, os *hosts* intermediários denominados refletores são usados para inundar a vítima. Alguns ataques nesta categoria empregam o uso de protocolos bem conhecidos, como DNS [30] (*Domain Name System*) e NTP [31] (*Network Time Protocol*) para exaurir os recursos da vítima.

O DNS é um sistema hierárquico e distribuído que traduz um endereço da *web* fornecido pelo usuário em um endereço IP correspondente e vice-versa [32]. O sistema é composto por diversos servidores DNS que prestam serviços a clientes. Um modo de operação de um servidor utilizado no DNS é o recursivo. Esse tipo de servidor é responsável por receber consultas de clientes e consultar servidores externos para obter a resposta [33]. Quando um servidor recursivo está configurado para responder qualquer cliente e não somente os locais se diz que esse servidor é aberto.

O ataque volumétrico por reflexão utilizando o DNS pode funcionar da seguinte forma, como mostra a Figura 2.4 [34]. O atacante envia uma mensagem para vários servidores DNS recursivos abertos (comprometido) com o IP de origem alterado, colocando o endereço IP da vítima com endereço de origem da consulta e pede que o servidor responda com todos os seus registros sobre uma determinada zona sejam enviadas para a vítima e não para a máquina que fez as consultas. A vítima recebe as respostas DNS, com ampli-

ficações aproximadamente 10 a 80 vezes do que a consulta, ocasionado à amplificação do ataque.

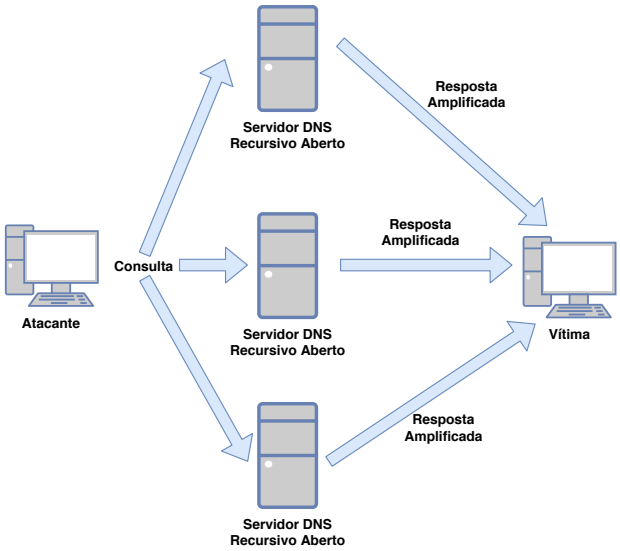


Figura 2.4: Ataque DNS

O NTP é um protocolo utilizado para sincronizar o relógio de sistemas computacionais [35] [36]. Ele funciona sobre um arquitetura cliente-servidor onde os clientes sincronizam os seus relógios com os servidores. Existem muitos servidores NTP públicos em toda a Internet que são usados por clientes legítimos para sincronizar o relógio do sistema.

O ataque volumétrico por reflexão utilizando o NTP, explora servidores NTP abertos que permitem o uso do comando *monlist*. Este comando retorna até os últimos 600 endereços IP do cliente que se conectaram a um servidor NTP. O tamanho do pacote da solicitação *monlist* é cerca de 64 *bytes* e a resposta pode ser ampliada para 100 respostas de 482 *bytes* cada [35]. A arquitetura do ataque pode ser observada na Figura 2.5. O ataque é realizado enviando uma solicitação *monlist* com um endereço de origem falsificado do alvo pretendido do ataque para diversos servidores NTP comprometidos. Em seguida, os servidores enviam as respostas para o IP falsificado (vítima) e, em seguida, é inundado com pacotes grandes, o que caracteriza a amplificação do ataque.

Ataques não volumétricos

Os ataques não volumétricos exploram recursos específicos do protocolo de comunicação utilizado pela aplicação, causando esgotamento de alguns recursos da vítima por meio de um baixo volume de tráfego. Alguns ataques como, *Slow* HTTP (*HyperText Transfer Protocol*) *headers* conhecido como *Slowloris*, *Slow Body* (RUDY) e *Slow Read*, exploram

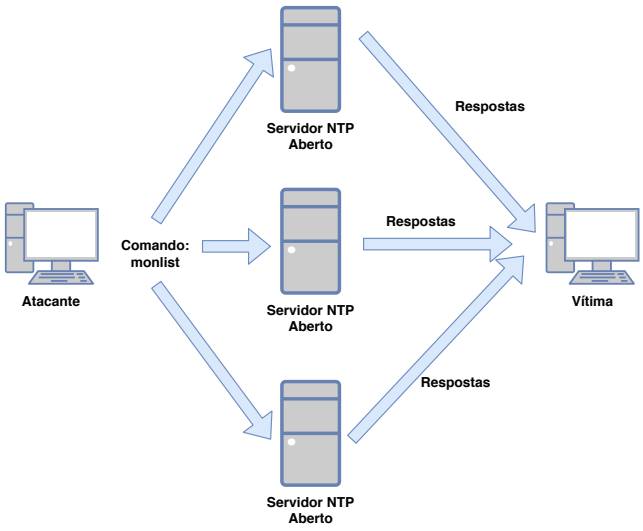


Figura 2.5: Ataque NTP

características que fragilizam o protocolo HTTP [37]. Este protocolo é projetado para manter a conexão aberta até que o envio ou recebimento de dados seja concluído [38].

Em uma solicitação e resposta HTTP normal, o cliente envia um HTTP GET para solicitar um recurso do servidor, e o servidor responde a essa solicitação e envia o recurso solicitado ao cliente. O protocolo HTTP, por sua natureza, precisa que os cabeçalhos (*headers*) das mensagens GET sejam completamente recebidos antes de serem processados. Se o cabeçalho HTTP GET não estiver completo, o servidor presume que o cliente está tendo uma conexão de internet lenta, portanto, o servidor mantém os recursos ocupados e aguardando o restante do cabeçalho HTTP GET. O ataque *Slowloris*, aproveita esta característica do servidor para enviar lentamente cabeçalhos HTTP GET falsos com a intenção de evitar com que temporizador de conexão expire e assim, manter o servidor ocupado de forma a não atender solicitações de clientes legítimos.

Ao contrário do *Slowloris*, no ataque RUDY, o atacante envia o cabeçalho completo para a vítima, mas esse cabeçalho tem um comprimento de conteúdo falso, que é muito maior do que o corpo real da mensagem. O atacante então, envia o corpo da mensagem em pequenos pedaços para manter o servidor ocupado que fica aguardando o restante do corpo da mensagem e assim, ele é impedido de processar outras requisições.

No ataque *Slow Read*, o atacante utiliza a funcionalidade de janelas TCP. O tamanho da janela TCP é um número de *bytes* que um receptor de dados está disposto a receber. O receptor de dados pode usar janelas TCP para controle de fluxos. O ataque consiste em definir o tamanho da janela TCP em quantidade relativamente pequena, em que na próxima etapa o atacante solicita uma quantidade relativamente grande de dados e define o tamanho da janela para zero ou um número muito pequeno, portanto, o servidor não

consegue enviar dados enquanto mantém a conexão aberta, de modo a esgotar os recursos do servidor web.

Os ataques acima mencionados representam uma grande ameaça para os serviços de rede e à Internet. Com pouco ou nenhum aviso, esses ataques tornam um serviço ou recurso indisponível e em decorrência disto a mitigação desses ataques é uma tarefa crucial e parte de uma estratégia geral do gerenciamento de ameaças para qualquer serviço. Para que haja a mitigação é necessário detectar esses ataques e assim, garantir a continuidade dos serviços ofertados.

A seção a seguir, apresenta os métodos de detecção quem podem auxiliar no reconhecimento dos ataques DoS/DDoS apresentados.

2.1.2 Métodos de detecção de ataques

Os métodos de detecção de intrusão desempenham um papel fundamental na descoberta de diferentes tipos de ataques em uma rede de computadores. Eles estão alinhados a três principais atividades de segurança, como monitoramento, detecção e resposta.

O objetivo de um sistema (mecanismo) de detecção de intrusão (do inglês, *Intrusion Detection System* - IDS), é detectar os ataques (intrusos) que ocorrem em uma rede. As principais funções desses métodos incluem a identificação de atividades maliciosas, o registro de informações sobre elas, bloqueio e a notificação desses incidentes a administradores de segurança. Os IDS podem ser baseados em assinatura (uso indevido) ou em anomalia [39].

O IDS baseado em assinatura, detecta os ataques que são previamente adicionados em sua base de dados. O método compara o tráfego com os modelos conhecidos, chamados de assinaturas e, quando os resultados são encontrados, informa o ataque. Os IDS baseados em assinaturas são eficazes na detecção de ataques conhecidos, produzindo vantagens, como alertas para uma assinatura correspondida e gera uma baixa taxa de alarme falso. Porém, eles falham ao tentar identificar ataques desconhecidos.

O IDS baseado em anomalia, detecta os ataques desconhecidos identificando um comportamento anormal na rede. Este tipo de detecção é configurado para entender o comportamento normal de uma rede, de modo que ele possa detectar desvios a partir de uma linha de base e assim, quaisquer variações dessa linha são relatadas como possíveis ataques. Este método apresenta vantagens, como identificar ataques cuja a informação completa do ataque não está presente e não é necessário ter um banco de dados com informações pré-estabelecidas sobre os ataques. Porém, ele gera uma alta taxa de alarmes falsos.

Os métodos de detecção de ataques podem ser classificados de diversas formas. Segundo Lee *et al.* [40], a classificação pode ser baseada no volume de tráfego, no qual

analisa a estrutura geral do tráfego e tenta encontrar anomalias de acordo com os níveis de desvio do volume de tráfego normal. Já You *et al.* [41] propuseram que a classificação dos mecanismos pode ser baseada no princípio de utilizar atributos do pacote TCP/IP, para detectar desvios no tráfego de ataque. Em contrapartida as classificações propostas, Beitollahi *et al.* [42], apresentam uma classificação mais abrangente em que os mecanismos de detecção são classificados de acordo com os algoritmos usados para a detecção. De acordo essa taxonomia, existem quatro grupos: técnicas estatísticas, ponto de mudança sequencial, análise de *Wavelets* e redes neurais. Em redes neurais podemos estender para o campo da inteligência artificial como um todo.

Seguindo a classificação proposta por Beitollahi *et al.* [42], nas técnicas estatísticas são analisadas várias propriedades de campos específicos nos cabeçalhos de pacotes durante condições normais do tráfego. Em seguida, usando essas propriedades estatísticas, o sistema cria um modelo de referência para o tráfego normal. Quando ocorre um ataque, as propriedades estatísticas calculadas para o tráfego atual mostram diferenças do modelo de referência e, usando essas diferenças, os pacotes de ataques são detectados. Alguns métodos estatísticos, como Qui-Quadrado e Entropia são capazes de relatar esse tipo de informação.

- **Qui-quadrado**: o qui-quadrado [27] ou χ^2, constitui uma medida que retrata a diferença entre duas distribuições consecutivas. O princípio básico deste método é comparar proporções, isto é, as possíveis divergências entre as frequências observadas e esperadas para um certo evento. A equação é definida como:

$$\chi^2 = \sum_{i=1}^{n} [\frac{(o_i - e_i)^2}{e_i}], \tag{2.1}$$

  onde n é o número total de observações, o_i é a frequência observada para cada classe e e_i é a frequência esperada para aquela classe. Quando as frequências observadas são muito próximas às esperadas, o valor do qui-quadrado é pequeno, ou seja, existe uma alta correlação entre os conjuntos observados. Mas, quando as divergências são grandes, o valor do qui-quadrado assume valores altos, isso significa que não há um relacionamento entre o conjunto de dados observado.

- **Entropia**: a entropia de Shannon [43] mede a probabilidade de um evento acontecer com relação ao número total de eventos, por meio dela é possível descrever o grau de dispersão ou concentração de uma distribuição. A equação é definida como:

$$H = -\sum_{i=1}^{n} p_i \log p_i, \tag{2.2}$$

onde n é o número de diferentes ocorrências no espaço amostral sob análise e p_i a probabilidade associada a cada ocorrência i, ou seja, a frequência de ocorrência de cada item único dividido pelo número total de itens. O resultado deste cálculo varia entre 0 e $\log N$. O valor mínimo indica concentração máxima na distribuição medida, ou seja, um único valor i ocorreu durante todo o intervalo de observação. O valor máximo indica dispersão total na distribuição medida, ou seja, uma distribuição uniforme para todas as ocorrências dentro do intervalo de observação [43]. Em suma, quanto maior a aleatoriedade, maior a entropia e vice-versa. Apesar de retornar um valor negativo, esse valor é comumente transformado em valor positivo para retratar o grau de dispersão ou concentração da informação.

Alguns trabalhos já foram publicados, mostrando a eficácia desses métodos. Leu *et al.* [44], propuseram uma arquitetura de detecção de intrusão distribuída baseada em agentes. O objetivo era detectar ataques DoS/DDoS usando o qui-quadrado com base na frequência de conexão dos endereços IP de origem. Os resultados experimentais mostram que a abordagem usada pode efetivamente detectar ataques DoS/DDoS. Já Oshima *et al.* [45] propuseram o uso do qui-quadrado para detectar o tráfego considerado espúrio de origem do atacante, com base no número da porta de destino e no IP de origem. Tritilanunt *et al.* [46], propuseram um método de detecção utilizando a entropia, para calcular a aleatoriedade do tráfego com base no tamanho do pacote. De maneira semelhante Kurihara *et al.* [47], utilizaram a entropia e analisaram dois campos do cabeçalho do pacote, o horário de chegada e o endereço IP de origem, para distinguir o tráfego legítimo do tráfego DoS/DDoS.

As técnicas baseadas em algoritmos de ponto de mudança sequencial, isolam a mudança de uma estatística de tráfego causada por ataques [42]. Essas abordagens normalmente filtram o tráfego na vítima de acordo com diferentes critérios, tais como endereço, porta ou protocolo e armazenam os dados resultantes. O tráfego filtrado é tratado como uma série temporal, ou seja, uma coleção de observações feitas sequencialmente ao longo do tempo. A detecção de uma mudança no tempo é fornecida através da avaliação de todos os dados passados, a cada tempo deve-se distinguir entre dois estados do processo: fora de controle ou sob controle [48]. Uma classe de algoritmos de detecção de ponto de mudança sequencial é o algoritmo de soma cumulativa (CUSUM) [49].

- **CUSUM:** a soma cumulativa (CUSUM) baseia-se na coleta sucessiva de amostras no qual é obtida a estatística da soma acumulada. O procedimento começa propriamente com o cálculo dos desvios do valor-alvo, isto é, a diferença entre o valor observado (média amostral) e o valor-alvo. De posse deste desvio, a soma acumulada

é iniciada. A equação é definida como:

$$C_i = \sum_{j=1}^{i}(X_i - \mu_0) = C_{i-1} + (X_i - \mu_0) \tag{2.3}$$

onde C_i é a soma acumulada incluindo a i-ésima amostra, pois combinam informações de diversas amostras, μ_0 valor-alvo (valor pretendido) e X_i valor observado. Quando processo está sob controle, o resultado da equação flutua em torno de zero. Se a média desloca para um valor superior ou inferior, o processo tende a estar fora de controle.

O algoritmo CUSUM tem a propriedade de armazenar os valores das somas unilaterais do processo analisado, que são posteriormente comparadas com o limite de controle H para indicar se o processo está ou não sob controle. Seja X_i cada observação do processo, quando o processo está sob controle, as observações seguem uma distribuição normal com média e desvio padrão. Se o processo tende a se afastar do valor pretendido, então são utilizadas as somas unilaterais para detectar mudanças: C_i^+ (CUSUM Superior) para detectar mudanças positivas e C_i^- (CUSUM Inferior) para detectar mudanças negativas. Ambas são calculadas através do algoritmo de soma cumulativa conforme as equações:

$$C_i^+ = max[0, X_i - (\mu_0 + K) + C_{i-1}^+] \tag{2.4}$$

$$C_i^- = max[0, X_i - (\mu_0 - K) + C_{i-1}^-] \tag{2.5}$$

onde C_i^+ e C_i^- iniciais são iguais a zero, K é o fator de sensibilidade e deve ser escolhido de forma que o valor da soma $\mu_0 + K\sigma$ (ou $\mu_0 - K\sigma$), esteja situado entre a média do processo e a média deslocada (fora de controle) que se deseja avaliar. Para cada amostra são obtidos valores dos desvios C_i^+ e C_i^-, eles são inseridos em uma tabela e acumulados sucessivamente. A soma acumulada destes desvios é comparada com um intervalo H. Se $C_i^+ > H$ ou $C_i^- < H$, então o processo é considerado fora de controle.

Algumas pesquisas utilizando o algoritmo de CUSUM, mostraram ser eficazes para a detecção de ataques DoS/DDoS. No trabalho desenvolvido por Haining *et al.* [50], os autores propuseram um esquema de soma cumulativa (CUSUM) para encontrar os pontos de mudança no tráfego. A solução monitora o desvio do comportamento de curto prazo do comportamento de longo prazo estabelecido, se a diferença cumulativa for maior que um limite estabelecido, um ataque é evidenciado. Seguindo a mesma linha, Alezeni *et al.* [51], propuseram o monitoramento das janelas de congestionamento do pacote TCP em um

intervalo de tempo durante o ataque e utilizaram o algoritmo CUSUM, para detectar mudanças nos valores das janelas de congestionamento que possam evidenciar um ataque DoS/DDoS. Os autores utilizaram apenas o CUSUM superior para indicar o aumento no volume de tráfego.

As técnicas de análise de *Wavelets* descrevem um sinal de entrada em termos de componentes espectrais, que são representações das amplitudes ou intensidades [42]. As *Wavelets* fornecem uma descrição simultânea de tempo e frequência e podem, assim, determinar o tempo em que certos componentes de frequência estão presentes. Os sinais anômalos localizados no tempo podem ser separados dos sinais de ruído através das *Wavelets*. Ao analisar a energia de cada janela espectral, as anomalias podem ser determinadas.

Diversos trabalhos foram propostos usando o conceito de análise de *Wavelets* para detectar um ataque DoS/DDoS. Barford *et al.* [52], analisaram dados de tráfego em séries temporais distintas de tamanho médio de pacotes por segundo, fluxos por segundo e *bytes* por segundo. E então, aplicaram a análise de *Wavelets* a cada série temporal, resultando em energias espectrais de alta e média banda. Eles consideraram o conteúdo de baixa frequência uma atividade normal. Para detectar anomalias, eles ponderaram uma combinação de energias espectrais altas e médias, e então limitaram sua variabilidade. Já Li *et al.* [53], observaram que as distribuições de energia (espectro) sofrem uma acentuada alteração em suas distribuições, quando há presença de um volume intenso de tráfego, assim realizaram medidas para detectar o tráfego de ataque na rede com base nessa análise, enquanto o tráfego normal exibe uma distribuição de energia parada.

As técnicas de inteligência artificial, sugerem o desenvolvimento de algoritmos capazes de transformar dados em ações inteligentes, permitindo que seja realizada a tomada de decisões de maneira automática [42]. Atributos do tráfego como tamanho do pacote, número de pacotes são usados para treinar redes neurais e algoritmos genéticos, para mostrar a existência de possíveis ataques DoS/DDoS na rede.

- **Redes Neurais Artificiais**: as redes neurais artificiais (RNAs) são sistemas que assemelham o comportamento e a estrutura do cérebro humano [54]. Elas utilizam técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. O objetivo da rede neural é atingir uma solução generalizada para um conjunto de problemas, ou seja, fornecer valores adequados para entradas não contidas no processo de aprendizagem [55]. O treinamento da rede é realizado por meio de algoritmos de aprendizado, que permitem ajustar os pesos das conexões, de forma que os dados de entrada sejam classificados corretamente na saída. Uma das propriedades mais importantes das RNAs é a sua eficiência em classificar corretamente dados desconhecidos, isto é realizado em virtude de sua capacidade de

aprender por meio de exemplos e fazer inferências sobre o que aprendeu, melhorando gradativamente o seu desempenho.

- **Algoritmos Genéticos**: os algoritmos genéticos (AGs), são métodos de otimização inspirados em evolução, no qual os indivíduos mais adaptados sobrevivem e transmitem suas características para as gerações seguintes [56] [57]. Eles fazem parte dos algoritmos evolutivos de otimização e trabalham com uma coleção (população) de possíveis soluções. Cada solução é denominada cromossomos, e é composta por valores aleatórios para cada variável otimizadora chamados de genes, que representam uma característica independente das demais [58] [59]. Através de diversas operações eles evoluem até que se encontrem uma solução que melhor atenda a algum critério específico de avaliação. A cada geração os cromossomos são avaliados por uma função que mede o seu nível de aptidão. Os cromossomos que apresentarem a melhor aptidão são selecionados para iniciarem uma próxima geração utilizando operadores genéticos, como seleção, cruzamento e mutação. Desta forma, espera-se que através da utilização desses operadores, o conjunto de soluções de cada geração seja aprimorado, até que se obtenha uma solução que atenda o critério escolhido. A tarefa principal de um algoritmo evolutivo é buscar de forma eficiente, no conjunto de possíveis soluções de um problema, as soluções que estejam de acordo com o objetivo do problema.

Para o cenário de inteligência artificial, alguns trabalhos sugerem o uso de técnicas de redes neurais e algoritmos genéticos para a detecção de ataques DoS/DDoS. Zhao *et al.* [60], propuseram o uso de uma rede neural para treinar um enorme conjunto de dados não estruturados, para detectar possíveis comportamentos que evidenciam um trafego DDoS. Os resultados obtidos mostraram que o modelo de rede neural foi capaz de detectar um ataque DDoS a partir de uma série de 6 amostras de treinamento. Hsieh *et al.* [61], analisaram um conjunto de características, como número de pacotes, média do tamanho dos pacotes, variação de intervalo de tempo, variação do tamanho do pacote, número de bytes, taxa de pacote e taxa de bits, para treinar a rede neural e assim detectar um ataque DDoS. Os resultados mostraram que o mecanismo obteve uma acurácia de 94% na caracterização dos pacotes de ataques. Voltado para o uso de algoritmos genéticos, Guo *et al.* [62], propuseram um algoritmo genético para filtrar o tráfego de ataque em roteadores e assim, maximizar a largura de banda da rede. Seguindo a mesma linha, Wang *et al.* [63] utilizaram as estatísticas do NetFlow para alocar pesos para roteamento do tráfego, por meio de um algoritmo genético e assim garantir que a largura de banda da rede seja preservada durante um ataque DDoS, priorizando o tráfego legítimo.

Podemos observar que as técnicas apresentadas podem auxiliar na detecção de ataques DoS/DDoS, por meio delas é possível analisar um grande volume tráfego de rede e assim, propor meios para reduzir os efeitos causados por esse tipo de ataque.

A seção a seguir, apresenta os conceitos relacionados ao novo paradigma de redes, conhecido como Redes Definidas por *Software*.

2.2 Redes Definidas por Software

A concepção de Redes Definidas por *Software* (do inglês, *Software Defined Network* - SDN) nasceu com a criação do protocolo OpenFlow para a programação de *switches*, o que lhes permite expandir o acesso e controle da tabela de consulta utilizada pelo *hardware* para determinar o próximo passo de cada pacote recebido [1]. Neste tipo de rede existe uma separação lógica entre o plano de dados e o plano de controle, que define um ambiente programável. O plano de controle é definido com o *software*, enquanto o *hardware* (plano de dados) tem as tarefas relacionadas ao encaminhamento de pacotes de acordo com as regras definidas pelo *software* [9].

A ideia de redes programáveis foi proposta como um meio de remediar as limitações impostas pela atual arquitetura de uma rede tradicional, como arquitetura complexa e estática, políticas inconsistentes e dependência do fornecedor [6]. A arquitetura SDN oferece uma estrutura de gerenciamento flexível, eficiente e automatizada para lidar com a complexidade na configuração e no controle da rede. As políticas de rede em uma SDN são gerenciadas e implementadas nos controladores e depois distribuídas para os *switches*, eliminando a necessidade de gerenciar cada *switch* manualmente. A SDN fornece APIs (*Application Programming Interface*) consistentes para o gerenciamento do plano de controle e implementa interações do plano de dados por meio de protocolos padronizados, isso permite aos desenvolvedores experimentar novas ideias de forma independente do fornecedor, deixando assim a rede mais flexível e escalável com a capacidade de operar de acordo com as necessidades dos desenvolvedores [64].

A arquitetura SDN é composta por quatro pilares [65]:

1. O plano de controle e dados são desacoplados, resultando em uma arquitetura, na qual os dispositivos de rede executam tarefas simples de encaminhamento e são gerenciados pelo controlador;

2. As decisões de encaminhamento são baseadas em fluxo, em vez de baseadas em destino. Um fluxo é definido como uma sequência de pacotes de uma origem para um destino;

3. A lógica de controle é executada por uma entidade externa, o controlador SDN ou Sistema Operacional de Rede (do inglês, *Networking Operating System* - NOS);

4. A rede é programável por meio de aplicativos de *software* em execução no controlador, que interagem com os dispositivos do plano de dados.

Os quatro pilares formam uma estrutura em três camadas ou planos: (*i*) aplicação; (*ii*) plano de controle; e (*iii*) plano de dados. A Figura 2.6 apresenta a arquitetura SDN.

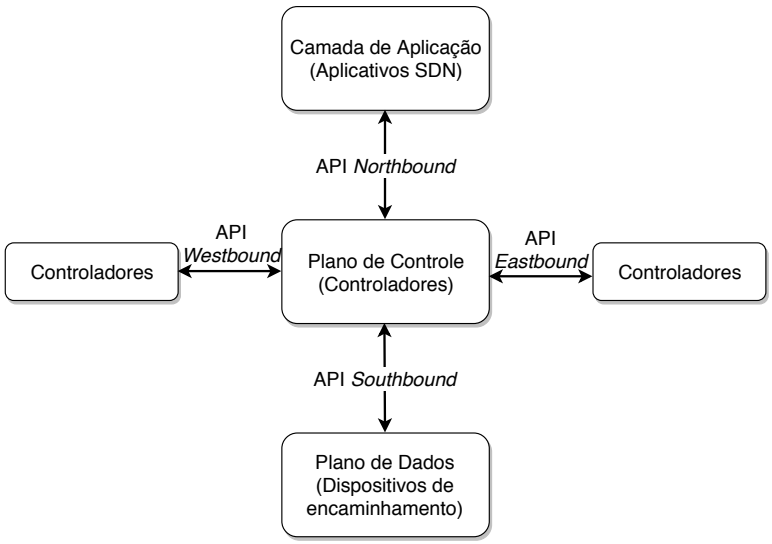


Figura 2.6: Arquitetura SDN adaptada de [1].

A camada de aplicação é formada por aplicativos SDN que fornecem vários recursos, como os relacionados ao gerenciamento de rede, implementação de políticas e serviços de segurança, como *firewall*, controle de acesso, serviço de *proxy*, balanceamento de carga e etc [66]. Esta camada auxilia a camada de controle na configuração da rede, e sua comunicação é realizada através da API *northbound*. Um exemplo desta API é a RESTful (*Representational State Transfer*) que utiliza requisições HTTP para extrair, inserir e deletar dados na rede.

O plano de controle é formado pelos controladores, que são responsáveis por programar e gerenciar o plano de dados, sendo encarregado de criar funcionalidades para realizar as decisões operacionais de como a rede deverá se comportar [7]. Ele pode ser visto como o cérebro da rede, já que toda a lógica da aplicação passa por essa camada para alcançar o plano de dados. Adicionalmente, o plano de controle ainda tem APIs horizontais à esquerda e à direita sendo as *westbound* e *eastbound*, para o gerenciamento de múltiplos controladores na rede [1]. Elas são um caso especial de interfaces requeridas por controladores distribuídos, as funções destas interfaces incluem importação e exportação de dados entre controladores, algoritmos para modelos de consistência de dados e recursos

de monitoramento e notificação, por exemplo, verificar se um controlador está ativo ou notificar uma substituição em um conjunto de dispositivos de encaminhamento.

O plano de dados compreende a infraestrutura de rede, ou seja, equipamentos como *switches*, roteadores e pontos de acessos [1]. Ele é responsável pelo encaminhamento e descarte de pacotes na rede, bem como o monitoramento de informações locais e coleta de estatísticas. A comunicação com o plano de controle é realizada pela API *southbound*, por exemplo OpenFlow [9], ForCES [67] e NetConf [68]. A implementação mais conhecida dessa API é o protocolo OpenFlow [9], que define um método genérico para permitir que o controlador interaja com os dispositivos de encaminhamentos da rede.

2.2.1 OpenFlow

O protocolo OpenFlow foi desenvolvido na Universidade de Stanford, e tem como objetivo padronizar a maneira como o controlador se comunica com os dispositivos de encaminhamento da rede em uma arquitetura SDN [8]. Ele explora a tabela de fluxo de *switches* e roteadores, para realizar ações de adicionar ou remover as entradas de fluxos por meio de uma interface padronizada e programável.

A comunicação entre o plano de controle e o plano de dados é realizada por meio de um conjunto de mensagens que são enviadas do controlador ao *switch* e um conjunto correspondente de mensagens enviadas na direção oposta, estas mensagens permitem que o controlador possa programar o *switch* para permitir um controle apurado sobre o tráfego de rede [2]. A arquitetura do protocolo consiste em três conceitos básicos: (*i*) a rede é constituída por *switches* OpenFlow que compõem o plano de dados; (*ii*) o plano de controle consiste em um ou mais controladores; e (*iii*) um canal seguro de comunicação, que conecta os *switches* com o plano de controle.

O *switch* OpenFlow possui uma tabela de fluxo, que executa pesquisa e encaminhamento de pacotes, e um canal seguro para comunicar a um controlador. O controlador gerencia o *switch* por meio do canal seguro usando o protocolo OpenFlow [2]. A Figura 2.7 exemplifica os componentes.

A tabela de fluxo do *switch* OpenFlow possui um conjunto de entradas de fluxos em que cada entrada tem uma ação associada a ela, tais como encaminhamento, descarte e envio de pacotes ao controlador [2]. Essa tabela define como os pacotes que chegam ao *switch* devem ser manipulados, na versão 1.0 do OpenFlow ela possui três campos, cabeçalho (regras), ações e contadores. Cada campo possui um conjunto de entradas específicas, por exemplo, o campo de cabeçalho é possível observar os atributos relativos aos fluxos que ingressam na rede, como IP de origem, IP de destino, porta de origem, porta de destino, protocolo, tamanho do pacote, dentre outros, já o campo de ações pode indicar o comportamento a ser tomado para determinado fluxo, como mostra a Figura 2.8.

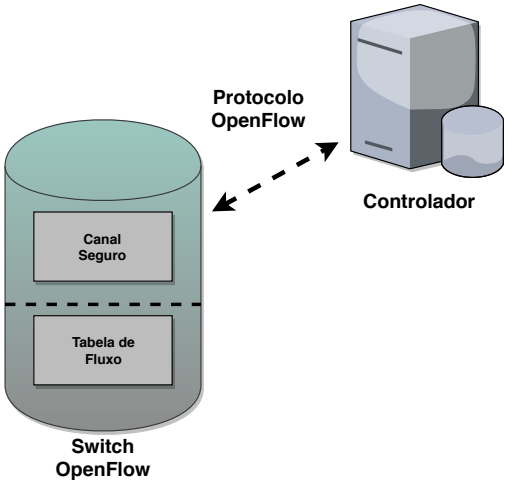


Figura 2.7: Protocolo OpenFlow adaptada de [2]

A tabela é única nessa versão do protocolo. A partir da versão 1.3, é possível ter múltiplas tabelas de fluxos, isso adiciona uma grande flexibilidade ao protocolo, pois agora é possível adicionar ações ao pacote em uma tabela, e em seguida, se necessário, encaminhá-lo para outra tabela de fluxo, onde novas ações podem ser adicionadas [5].

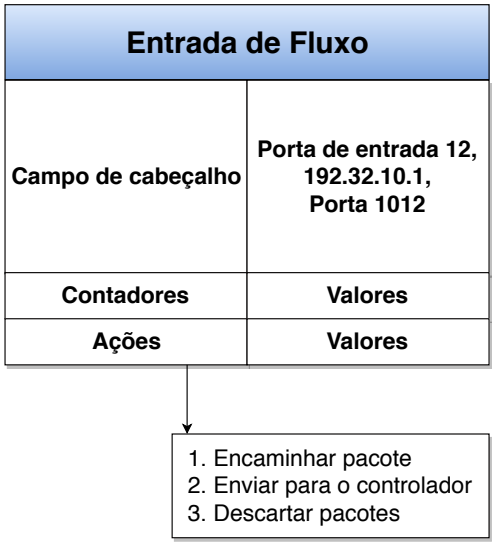


Figura 2.8: Tabela de fluxos do *switch* OpenFlow adaptada de [2]

As regras são formadas a partir de um ou mais campos do cabeçalho do pacote e correlacionados a um conjunto de ações, assim um fluxo pode ser identificado por uma combinação desses campos. Na versão 1.0 do protocolo existem 12 campos que podem ser usados no cabeçalho. As ações definem o modo como os pacotes devem ser processados.

Os contadores armazenam estatísticas referentes aos fluxos do *switch*, como o número de pacotes, tempo decorrido em que o fluxo foi instalado no *switch* e *bytes* transmitidos e recebidos. Esses dados podem ser encaminhados ao controlador de forma coordenada, por meio de um temporizador, que solicita o conjunto atual de estatísticas do *switch* para um determinado fluxo ou grupo de fluxos.

Todos os pacotes processados pelo *switch* são comparados com a tabela de fluxo. Se uma entrada correspondente for encontrada, uma determinada ação para essa entrada será executada, por exemplo, a ação de encaminhar um pacote para uma porta específica [2]. Conforme as versões do protocolo vão sendo lançadas, maiores são as flexibilidades de alterações e configurações de fluxos. A Tabela 2.1 mostra a evolução do protocolo OpenFlow, com relação ao número de campos de cabeçalhos disponíveis. A quantidade de campos do protocolo cresceu ao longo das versões de forma considerável. Esse crescimento pode ser identificado de forma positiva, como uma consequência de um bom desenvolvimento do paradigma SDN e do protocolo em si.

Tabela 2.1: Evolução do protocolo OpenFlow [5]

Versão	**Data**	**Cabeçalhos**
1.0	Dezembro 2009	12
1.1	Fevereiro 2011	15
1.2	Dezembro 2011	36
1.3	Junho 2012	40
1.4	Outubro 2013	41
1.5	Dezembro 2014	44

O canal seguro de comunicação OpenFlow é o meio utilizado para a comunicação entre o *switch* e o controlador. Por meio desta interface, o controlador configura e gerencia o *switch*. A comunicação entre os elementos da rede deve ser segura, evitando com que invasores possam capturar informações na troca de mensagens entre os dois componentes. A comunicação entre o controlador e o *switch* é realizada exclusivamente por meio do protocolo OpenFlow. Ele suporta três tipos de mensagens que podem ser trocadas entre os elementos [2]. Essas mensagens são classificadas como:

- **Controlador para *switch*:** são originadas pelo controlador e podem ou não exigir uma resposta do *switch*. Estas mensagens permitem que o controlador modifique estados e entradas de fluxos do *switch*, dentre outras características.
- **Assíncronas:** são originadas pelo *switch* sem a solicitação do controlador. Estas mensagens informam eventos na rede, chegada de pacotes, mudanças no estado de um *switch* ou erros. Um exemplo deste tipo de mensagem é a *Port-status*, que envia mensagens de status da porta do *switch* para o controlador à medida que o estado

de configuração da porta é alterado. Outro exemplo é a mensagem *packet-in*, ela é enviada sempre que um pacote não tem uma entrada correspondente na tabela de fluxo.

- **Simétricas:** são originadas sem a solicitação do *switch* ou controlador, em qualquer direção. Um exemplo deste tipo de mensagens são *Hello* e *Echo*. A primeira é trocada entre o controlador e o *switch* na inicialização de uma conexão, a segunda é usada para indicar a latência, largura de banda ou atividade de uma conexão entre o *switch* e o controlador.

Por último temos o controlador, que é responsável por programar todas as regras de correspondência e encaminhamento de pacotes nos *switches* OpenFlow [2]. Trata-se de um *software* que torna viável o gerenciamento da rede, sendo responsável por monitorar, gerenciar e tomar decisões referentes à rede, de forma que todo o processamento do fluxo no plano de dados seja baseado nas suas instruções. A Seção 2.2.2 apresentará os principais controladores que oferecem suporte ao protocolo OpenFlow.

Estabelecidos os mecanismos que compõem o protocolo, é necessário entender como eles atuam em uma rede SDN. A Figura 2.9 demonstra o funcionamento básico de um *switch* OpenFlow. A cada pacote que ingressa na rede, o *switch* OpenFlow realiza a separação dos cabeçalhos e verifica se existe alguma entrada correspondente em suas tabelas de fluxos. Caso exista, o *switch* aplica a ação correspondente. Do contrário, ele envia o pacote para o controlador por meio do canal seguro e aguarda as instruções.

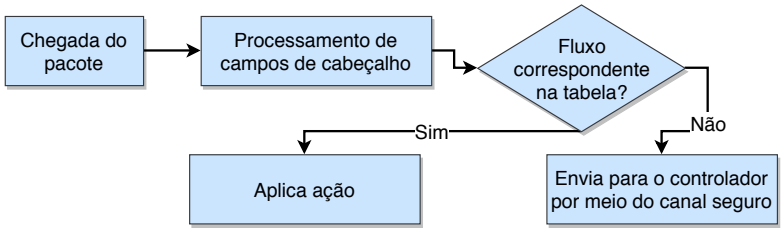


Figura 2.9: Processamento dos fluxos no *switch* adaptada de [3]

A Figura 2.10 demonstra o funcionamento básico de um controlador utilizando o protocolo OpenFlow. Ao receber o pacote, o controlador realiza a ação que foi previamente estabelecida em sua configuração para este tipo de fluxo, aplicando uma ou mais regras nos *switches*, por exemplo, encaminhar pacotes pertencentes a um determinado fluxo para uma porta específica, descartar o pacote como medida de segurança ou modificar os campos de cabeçalho do pacote.

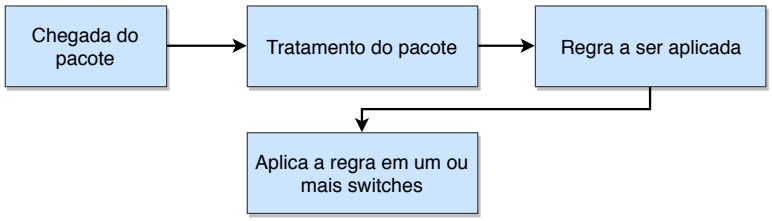


Figura 2.10: Processamento dos fluxos no controlador adaptada de [3]

2.2.2 Controladores SDN

O controlador é o principal componente de uma rede SDN, ele é o que define a natureza do paradigma, sendo responsável por coordenar e gerenciar os recursos de toda a rede, oferecendo uma visão unificada [7]. Por meio dele é possível implementar novas funcionalidades para chegar a decisões operacionais de como o sistema deve operar.

O controlador pode ser programado através de uma API, definindo o que será feito em cada um dos planos. Assim, cada controlador utiliza uma linguagem de programação, como C, C++, Java e Python para implementar comandos para auxiliar no controle da infraestrutura da rede. A seguir, são apresentados os controladores amplamente utilizados em trabalhos, pesquisas e pequenas implementações que fornecem suporte ao protocolo OpenFlow:

- **NOX**: o controlador NOX [69] foi quem deu início as pesquisas e ao desenvolvimento dos chamados sistemas operacionais de redes, desenvolvido em linguagem C++ e posteriormente implementado em Python, ele opera sobre o conceito de fluxo de dados, no qual checa o primeiro pacote de cada fluxo e procura uma entrada correspondente na tabela de fluxo para aplicar uma determinada ação. A interface do controlador gira em torno de eventos, em que um evento corresponde a mudança de fluxos, para lidar com esses eventos ele utiliza um conjunto de manipuladores para a execução de um determinado evento, o valor de retorno de um manipulador indica ao controlador se a execução deve ser interrompida ou continuar passando para o próximo manipulador. Este controlador possui um conjunto de bibliotecas, que fornece implementações de funções comuns ao gerenciamento de rede, como módulo de roteamento, classificação rápida de pacotes e uma rede filtrada por meio de políticas [69]. Este controlador obteve uma grande aceitação entre os pesquisadores, graças a existência de duas interfaces: C++ e Python, o que permite que o mesmo ambiente seja utilizado em situações que exigem alto desempenho e em casos onde é necessário um desenvolvimento que simplifica o código resultante. Ele possui suporte apenas à versão 1.0 do protocolo OpenFlow [9].

- **POX**: o controlador POX [70] é tido como uma evolução do NOX [69], ele foi desenvolvido em Python e fornece uma plataforma para o desenvolvimento e prototipagem rápida de aplicações de rede. Tem sua arquitetura baseada no seu antecessor o NOX, por meio dele é possível executar diferentes aplicativos, como *hub*, *switch*, balanceador de carga e *firewall*. Apesar de fornecer uma maneira eficiente de implementar o protocolo OpenFlow, ele suporta apenas a versão 1.0 do protocolo, o que limita as aplicações que podem ser desenvolvidas sob o controlador aproveitando a evolução do protocolo.

- **BEACON**: o controlador Beacon [71] é escrito em Java e suporta operações baseadas em eventos, semelhantes aos controladores NOX [69] e POX [70], e também as operações em *threads*. Trata-se de um controlador que possui uma estrutura modular, o que lhe permite realizar atualizações em tempo de execução sem interromper outras atividades de encaminhamento de pacotes. Ele fornece um conjunto de bibliotecas para o desenvolvimento de aplicativos, por exemplo, para fins de roteamento, o qual fornece uma interface chamada *IRoutingEngine*, em que é possível implementar o roteamento aplicando o algoritmo de caminho mais curto (*Shortest Path Routing*). Já a biblioteca *IDeviceManager* possibilita rastrear os dispositivos vistos na rede, incluindo seus: endereços *Ethernet* e IP (*Internet Protocol*), data da última vista e a porta vista pela última vez, o que lhe dá a capacidade de identificar quando novos dispositivos forem adicionados, atualizados ou removidos [71]. Além de possibilitar a reutilização de códigos, uma importante propriedade das linguagens orientadas a objetos, com isso ele pode criar várias instâncias de um objeto e vinculá-las em uma única entidade. Possui suporte a diferentes versões do protocolo OpenFlow, desde a versão 1.0 até a versão 1.3.

- **FLOODLIGHT**: o controlador FloodLight [72] foi desenvolvido a partir do Beacon [71] e é escrito em Java. Ele possui uma arquitetura modular, com componentes para o gerenciamento de dispositivos como o rastreamento MAC (*Media Access Control*), IP, escolha de melhor caminho e acesso ao gerenciamento via interface *web*. Ele apresenta módulos que oferecem uma API que pode ser usada pelas aplicações REST, são elas: *thread pool, packet streamer, web ui, flow cache, link discovery e floodlight provider* [72]. Dentre os módulos apresentados, o módulo principal é chamado de *Floodlight Provider*, que lida com as entradas e saídas dos *switches* e converte as mensagens OpenFlow em eventos, criando assim uma estrutura de aplicativo assíncrona orientada a eventos [72]. Além disto, o controlador incorpora um modelo de *threading* que permite que os módulos compartilhem *threads* com outros módulos. O controlador possui suporte a diferentes versões do OpenFlow,

em destaque a versão 1.3 a mais disseminada entre a comunidade SDN.

- **OPENDAYLIGHT**: o controlador OpenDaylight [73] é baseado em uma arquitetura de micro serviços, o que permite aos usuários o controle por aplicações e protocolos. Ele também é considerado um *framework* para aplicativos SDN, com o objetivo de criar um padrão de API *northbound* que de fato pode ser utilizado para programar diferentes protocolos *southbound*, incluindo o OpenFlow. É desenvolvido sob a linguagem Java e possui como características chaves: modularidade e a flexibilidade, o que permite aos desenvolvedores selecionar os recursos mais importantes para eles e criarem controladores que atendam às suas necessidades específicas. Possui suporte as versões 1.0 e 1.3 do protocolo OpenFlow [9].
- **RYU**: o controlador RYU [74] é escrito em Python e é baseado em componentes que podem ser alterados e estendidos para criar novas aplicações. Os componentes incluem um suporte ao protocolo OpenFlow desde a versão 1.0 até a versão 1.5, gerenciamento de eventos, mensagens, gerenciamento de estado na memória e uma série de bibliotecas reutilizáveis, como NetConf [68] e sFlow [7].

Através dos controladores, os administradores de rede e desenvolvedores agora são apresentados a um ambiente homogêneo mais fácil de programar e configurar toda a rede, aumentando a gama de aplicações a serem desenvolvidas. Entretanto, a programação da rede não se restringe apenas à programação do controlador, ela pode ser estendida a programação dos dispositivos de encaminhamento. A seção a seguir, aborda como isto pode ser realizado.

2.2.3 Plano de Dados Programável

As redes SDN surgiram como uma tecnologia que modifica a forma como os dados são encaminhados, introduzindo programabilidade e flexibilidade na rede. Inicialmente diversos protocolos foram propostos, dentre eles o mais difundido é o OpenFlow [2], que padroniza a maneira como o controlador se comunica com os dispositivos de redes.

O protocolo OpenFlow segmenta *switches* de função fixa do tipo ASICs (*Application Specific Integrated Circuits*) que reconhecem um conjunto predeterminado de campos de cabeçalho e que processam os pacotes usando um pequeno conjunto de ações predefinidas, o que torna a inserção de novas funcionalidades um processo engessado e dependente de novas atualizações do protocolo e dos fornecedores de *hardware* [4] [75].

Diante de tal limitação, houve a necessidade de mudanças na forma como os pacotes são processados de modo a oferecer flexibilidade a rede. A programação da rede é então estendida ao plano de dados, em que não é mais suficiente dizer o que fazer com os fluxos

de dados, é necessário também descrever como fazer. Nesse cenário, surgiu a linguagem de processamento de pacotes P4 (*Programming Protocol Independent Packet Processors*), que permite adicionar novos cabeçalhos e especificar como os pacotes de um determinado fluxo devem ser analisados e processados pelos *switches*, por meio de uma linguagem de alto nível, possibilitando criar novas funcionalidades e adicionar inteligência ao plano de dados [4].

A linguagem P4 se baseia na arquitetura PISA (*Protocol-Idenpendent Switch Architecture*), no qual os *switches* não precisam estar vinculados à nenhum protocolo de rede específico, por exemplo, OpenFlow [4]. Essa arquitetura tem duas operações básicas: configurar e preencher. A primeira, determina quais protocolos serão suportados e como os *switches* podem processar os pacotes. A segunda, adiciona (e remove) entradas a tabela de correspondência dos *switches*, estabelecendo as políticas aplicadas aos pacotes. A Figura 2.11 mostra o modelo de encaminhamento dos *switches* programáveis.

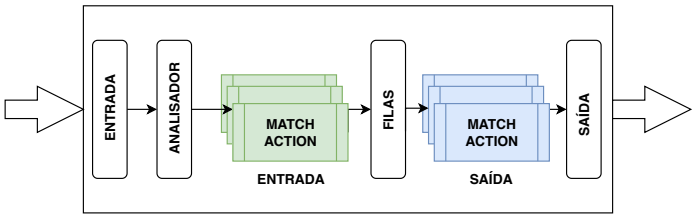


Figura 2.11: Modelo de encaminhamento dos *switches* programáveis, adaptada de [4]

Os pacotes que chegam ao *switch* são primeiro tratados pelo analisador. Ele, então, extrai os cabeçalhos e, assim, define os protocolos que o dispositivo suporta e o tratamento dado a ele. Os campos extraídos são enviados para as tabelas de correspondência (*match-action*), que são divididas em tabela de ingresso (entrada) e tabela de egresso (saída). As tabelas de ingresso, determinam as ações a serem tomadas, por exemplo, encaminhamento, replicação, rejeição e *etc.* Já as tabelas de saídas realizam modificações no cabeçalho, se necessário, por exemplo, atribuir a uma fila, alterar a porta de saída e *etc.* Entre as tabelas temos o enfileiramento que processa as especificações de saída, gera as instâncias necessárias do pacote e as envia ao *pipeline* de saída do dispositivo. Após a conclusão de todo o processamento pelo *pipeline* de saída, o pacote é transmitido.

A linguagem P4 permite utilizar estruturas para manter informações sobre determinado pacote, como registradores e contadores. Além, de possibilitar que os pacotes carreguem consigo informações adicionais por entre os estágios do *pipeline*, através dos metadados. Eles contêm informações essenciais sobre o pacote ingressante, tais como porta de entrada, a porta de saída, o *timestamp*, prioridade atribuída, fila de transmissão, ou outro tipo de dado importante para o processamento do mesmo.

Os principais componentes da linguagem P4 são:

- **Cabeçalhos:** a definição do cabeçalho descreve a sequência e estrutura de uma série de campos de *bits* definidos pelo programador. Inclui especificações de largura, restrições de tipos e valores.
- **Analisador:** especifica como identificar (reconhecer) cabeçalhos ou sequência de cabeçalhos válidos no pacote. Para isso, ele assume que o dispositivo de destino é capaz de implementar uma máquina capaz de analisar o cabeçalho de cada pacote do início ao fim, extraindo seus diversos campos um por um.
- **Tabelas:** as tabelas *match-action* definem os mecanismos para realizar o processamento dos pacotes, nas quais são associadas ações (*actions*) aos pacotes de acordo com a correspondência (*match*) realizada no analisador.
- **Ações:** define um conjunto de primitivas usadas para realizar alguma função, por exemplo, copiar um campo para outro, definir um campo específico em um cabeçalho para um valor e etc.
- **Controle:** estabelece o controle de fluxo das tabelas, sendo responsável por especificar o caminho a ser seguido pelo pacote entre uma tabela e outra.

Os componentes mencionados acima, definem um programa escrito na linguagem P4. Em geral, o analisador identifica os cabeçalhos presentes em cada pacote ingressando no dispositivo. Cada tabela *match-action* realiza uma busca em um subconjunto de campos de cabeçalho e aplica as ações correspondentes a primeira correspondência encontrada na tabela.

2.2.4 Segurança em Redes SDN

A segurança em redes SDN é um tema bastante recorrente, uma vez que a rede a passa ser programável por meio de um *software*, torna-se mais fácil criar e implementar soluções de segurança para controlar cada aspecto da rede. Porém, a centralização deste controle implica em questões para a segurança, como escalabilidade, desempenho da rede e um único ponto de falha o que pode reduzir toda a rede em caso de comprometimento da segurança [16].

A visão central de uma rede SDN permite que a lógica de controle das aplicações de segurança seja mais completa do que as atuais e, portanto, simplifica o tratamento de problemas complexos de segurança em rede [76]. A visibilidade dos recursos de rede é de suma importância, mas esses recursos não devem ser visíveis para todos os aplicativos ou para os quais não se preocupam. A criação de aplicações de segurança em SDN é um desafio, pois a própria segurança desta rede ainda é contestável [18].

Os desafios de segurança de uma rede SDN podem ser divididos em três categorias: (*i*) ausência de confiança entre componentes; (*ii*) vulnerabilidade de componentes; e (*iii*) negação de serviço [10].

1. **Ausência de confiança entre componentes:** a ausência de confiança entre os componentes da rede pode comprometer a estrutura da rede, pois as aplicações executadas sobre o controlador podem ter comportamentos maliciosos [10].

2. **Vulnerabilidade de componentes:** a vulnerabilidade dos componentes pode ocorrer a partir de duas fontes: *switch* e controlador [10]. As vulnerabilidades em um *switch* podem permitir que um invasor, que obtém acesso a um *switch* ataque o plano de controle, por exemplo, falsificar mensagens de outros *switches* para esgotar os recursos do controlador. Já as vulnerabilidades no controlador, permitem que um invasor altere toda a configuração do plano de controle ou até mesmo execute um novo aplicativo para controlar a rede.

3. **Negação de serviço:** a negação de serviço pode ocorrer tanto no plano de dados quanto no plano de controle [10]. Ela consiste em maliciosamente criar um tráfego excessivo em um espaço curto de tempo, com o objetivo de exaurir os recursos de uma vítima ou infraestrutura usada para acessar esses recursos. Causando inundação na rede e redução na largura de banda da aplicação, impedindo que a vítima seja acessada [11]. No plano de dados, um *host* malicioso pode gerar fluxos falsos para esgotar os recursos de largura de banda ou entradas de tabela de fluxo dos *switches* na rede. Enquanto no plano de controle, a negação de serviço pode ocorrer no canal de comunicação entre o controlador e os *switches*, no qual o invasor pode esgotar a capacidade de processamento do controlador, enviando muitos pacotes com cabeçalhos diferentes, que não correspondem a nenhum fluxo já definido e, portanto, são enviados ao controlador.

Estes desafios deixam claros que apesar dos inúmeros benefícios que uma rede SDN proporciona, a segurança ainda é motivo de preocupação. Podemos observar que os ataques de negação de serviço, tema abordado em redes tradicionais mantém-se presente na arquitetura SDN, principalmente devido às vulnerabilidades existentes entre as camadas (plano de dados e controle). A rede SDN também está sujeita aos mesmos ataques de redes tradicionais. Desta forma, as técnicas frequentemente empregadas em redes de arquitetura tradicional para detectar esse tipo de ataque, podem ser aplicadas ao ambiente SDN com as adequações que a nova arquitetura impõe sobre a rede.

2.2.5 Ferramentas de emulação e simulação

As ferramentas de emulação e simulação de ambientes SDN possibilitam testar as soluções de rede antes mesmo de sua utilização no mundo real. Alguns fatores, como investimento, relação custo-benefício, complexidade de gerenciamento e tempo necessário para implementação, contribuem para o uso destas ferramentas. As principais ferramentas para o desenvolvimento de uma rede SDN são:

- **NS-3**: o ns-3 é um simulador de redes de código aberto escrito em C++, desenvolvido para fornecer uma plataforma de simulação de rede voltado principalmente para pesquisa e uso educacional [77]. Ele fornece modelos de como as redes e pacotes funcionam e executam, e viabiliza um mecanismo de simulação para os usuários conduzirem experimentos de simulação. As principais características do simulador são: possibilidade de integração com bibliotecas externas e suporte a diferentes sistemas operacionais, como Linux e Windows. Embora muito utilizado para simulações de rede com arquitetura tradicional, o simulador possui suporte a versão 0.8.9 do protocolo OpenFlow, não muito usada já que o protocolo evoluiu.

- **EstiNet**: o EstiNet é um simulador e emulador de redes SDN com suporte ao protocolo OpenFlow nas versões 1.0 e 1.1, ele utiliza uma metodologia de simulação chamada de *kernel re-entering* que permite a utilização de controladores reais [78]. A ferramenta combina as vantagens das abordagens de simulação e emulação. Em uma rede simulada no EstiNet, cada *host* simulado executa um sistema operacional Linux e qualquer programa de aplicativo Unix. Por meio dele é possível utilizar os seguintes controladores: POX [70], NOX [69] e Floodlight [72]. Ele pode operar no modo emulação, no qual uma máquina executando um controlador OpenFlow real pode controlar uma rede OpenFlow em outra máquina.

- **Mininet**: o Mininet é um emulador de redes desenvolvido por pesquisadores da Universidade de Stanford, com o objetivo de auxiliar o desenvolvimento de redes SDN [79]. Ele simula uma coleção de *hosts* finais, como *switches*, roteadores, controladores e *links* em um único *kernel* Linux. Um *host* no Mininet comporta-se exatamente como uma máquina real, dessa maneira o *host* representa um *shell* de uma máquina, na qual qualquer programa pode ser conectado e executado. Esses programas personalizados podem enviar, receber e processar pacotes por meio de um *switch* virtual. O emulador permite o desenvolvimento de aplicações que utilizam o protocolo OpenFlow até a versão 1.3 e aos controladores: NOX [69], POX [70], Beacon [71], Floodlight [72], OpenDayLight [73] e Ryu [79].

A utilização dessas ferramentas permite simular ambientes de redes complexos, interagir e customizar protótipo de redes SDN, como soluções de segurança de forma a integrá-las com a rede física existente.

2.3 Considerações Finais

Este capítulo abordou os principais conceitos em torno de segurança em redes tradicionais e redes SDN, como pode ser visto estes conceitos são extremamente abrangentes e fornece um panorama acerca dos desafios de segurança em ambas as arquiteturas (tradicional e SDN). A seção a seguir, expõe uma revisão do estado da arte, com trabalhos relacionados à segurança em redes SDN.

Capítulo 3

Revisão do Estado da Arte

Neste capítulo, será apresentada uma revisão do estado da arte acerca de trabalhos relacionados à segurança em redes SDN, com ênfase no ataque SYN *Flood*. Na Seção 3.1 será utilizada a classificação proposta por Imran *et al.* [80] para as soluções apresentadas. Cada solução será classificada com base em sua metodologia para lidar com o ataque em questão. A Seção 3.2 apresenta uma discussão sobre as soluções estudadas.

3.1 Classificação das soluções de segurança em SDN

Uma solução de segurança em uma rede SDN pode ser classificada através do tipo de ataque que pretende mitigar, a metodologia para detectar o tráfego malicioso, os aspectos de segurança que ela pretende manter, ou a camada SDN em que atua. Essas classificações nos permitem observar as camadas de abstração em que cada solução atua, o tipo de ataque, qual camada ele pode garantir a segurança na arquitetura SDN, a proposta de mitigação e os aspectos de segurança específicos que elas poderiam comprometer.

As soluções de segurança apresentadas a seguir, seguem a classificação proposta por Imran *et al.* [80], no qual os autores categorizaram as soluções estudadas em três classes com base em sua metodologia para lidar com o tráfego espúrio, são elas: bloqueio, controle (atraso) e gerenciamento de recursos. Cada uma dessas classes é discutida abaixo junto com suas soluções relacionadas ao desenvolvimento de trabalho. Esta classificação nos permite reconhecer qual o modelo de técnica mais aplicado para preservar os recursos da rede.

3.1.1 Bloqueio

As soluções que compreendem a classe de bloqueio procuram mitigar o ataque bloqueando as portas às quais os *hosts* maliciosos estão conectados ou descartando o tráfego espúrio

de forma imediata. Grande parte das soluções de mitigação propostas nos últimos anos pertencem à essa classe, pois proporcionam um alívio imediato aos recursos da rede. Aqui, serão apresentadas algumas soluções que atenuam o ataque SYN *Flood*.

Shin *et al.* [15] propuseram o Avant-Guard, uma solução que adiciona proteção contra ataques SYN *Flood*, introduzindo dois módulos diretamente no *switch*: (*i*) migração de conexão e (*ii*) gatilhos de atuação. O primeiro módulo atua como um *proxy* para as conexões TCP, que intercepta as informações da conexão em uma sessão e encaminha a solicitação ao controlador apenas para conexões que foram estabelecidas, ou seja, as solicitações que executaram o *three-way handshake*, do contrário a conexão é imediatamente descartada pela solução. Este módulo é responsável por permitir ou não a migração de conexão, ou seja, a transferência dos dados para o destino. Já o segundo módulo permite que o plano de dados relate o status da rede ao controlador, para aplicar regras de bloqueio às portas do *switch*. A solução, no entanto, causa um longo atraso para estabelecer uma nova conexão para solicitações legítimas, devido ao uso do *proxy* e está sujeita a ataques DoS em que o atacante utilize um IP já validado para estabelecer as conexões.

Sonchak *et al.* [81] desenvolveram o OFX (*OpenFlow Extension Framework*), um *framework* que permite o desenvolvimento de soluções de segurança em módulos incrementais. A solução permite que diferentes soluções sejam carregadas dinamicamente por meio de módulos de *software* diretamente nos *switches* OpenFlow. A estrutura da solução consiste em um conjunto de bibliotecas de módulos que carregam recursos adicionados por outros aplicativos. Para demonstrar a eficiência do OFX, os autores desenvolveram um módulo de segurança que fornece aos dispositivos de encaminhamento duas funções adicionais. Alerta, que informa o *switch* para enviar um alerta ao controlador se o pacote ou a taxa de *bytes* do tráfego destinado a um *host* monitorado exceder o limiar proposto. E validação de conexão TCP, que permite que o *switch* classifique as conexões em válidas ou inválidas, por meio de um *proxy* no *switch*, similar ao Avant-Guard [15]. A solução está sujeita a conflitos entre os módulos instalados, em que um módulo pode sobrescrever as regras implementadas pelo outro e também as limitações impostas ao Avant-Guard.

Fichera *et al.* [14] apresentaram uma solução chamada Operetta, que transforma o controlador em um *proxy* para verificar a autenticidade de um *host* de origem. Sempre que a solução recebe um novo pacote SYN, o controlador responde com um SYN-ACK, como se fosse um servidor *web*. Se o cliente responder a este pacote com um ACK, o controlador realiza a validação da conexão, antes de encaminhar ao destino. Ao validar a conexão, ele então envia um pacote RST, indicando que o cliente deve tentar novamente a solicitação de conexão. Após estabelecida a relação de confiança, a solução instala regras de encaminhamento direto no *switch* para esse cliente, para que, quando o cliente tentar novamente a solicitação de conexão, sua solicitação SYN não seja encaminhada

para o controlador e sim ao seu destino. Se o cliente não enviar o pacote ACK de volta, o MAC de origem do cliente será colocado em uma lista temporária e seu contador SYN correspondente será incrementado até um limite predefinido. O contador é usado para indicar o número de conexões simultâneas semi-abertas reservadas para um *host* específico identificado através de um determinado endereço MAC, se o número de solicitações SYN exceder o limite definido, o controlador instala uma regra no *switch* para realizar o bloqueio do *host*, assim rejeitando solicitações maliciosas. Embora, a solução detecte um ataque SYN *Flood*, ela pode se tornar vulnerável em situações que o atacante conclua o primeiro TCP *handshake*, pois uma vez que a entrada é instalada no *switch*, ele pode sobrecarregar o servidor com um grande volume de novas solicitações oriundas do primeiro pacote validado.

Moreno *et al.* [82] propuseram uma solução chamada Lineswitch. A solução utiliza o conceito de intermediação probabilística, isto é, ela faz o *proxy* apenas da primeira conexão de um determinado endereço IP no *switch*, enquanto as demais conexões subsequentes do mesmo endereço IP são intermediadas apenas com uma certa probabilidade, reduzindo a carga de trabalho do *switch*. Conexões definidas como maliciosas, ou seja, aquelas que não conseguem realizar o TCP *handshake* completo, comum para o ataque SYN *Flood* são adicionadas a uma *blacklist* e posteriormente descartados pela solução para evitar o estouro da tabela de fluxo e preservar a memória do *switch*. Contudo, ao utilizar um IP real, para que a probabilidade de endereços IP duplicados seja muito baixa, a solução pode ficar vulnerável ao ataque em que um IP já validado pode ocasionar a saturação dos dispositivos de encaminhamento.

Reza *et al.* [13] desenvolveram o Slicots, uma solução que monitora todas as conexões entre cliente e servidor e adiciona regras temporárias de curto prazo. Desta forma, quando um pacote SYN-ACK é encaminhado pelo servidor para o cliente, no momento em que o cliente realiza o pedido de abertura de conexão (SYN), o controlador instala uma regra de encaminhamento temporário de curto prazo, para que ela seja removida rapidamente e dê espaço para os novos fluxos. Assim que o processo de conexão é finalizado entre cliente e servidor e tem como resultado uma conexão bem-sucedida, o controlador modifica a regra temporária, transformando-a em uma regra permanente de encaminhamento para o cliente e servidor. Caso contrário, a conexão entre o *host* solicitado e o servidor de destino permanece meio aberta por um determinado período. A solução monitora o número de conexões semi-abertas para um *host* solicitado específico, caso ele exceda um limite predefinido, a solução instalará uma regra de encaminhamento para bloquear do *host* malicioso, para proteger a rede de uma eventual saturação aos dispositivos da rede. Embora, a solução impeça os usuários de atividades maliciosas, a instalação de regras de encaminhamento temporário se torna uma tarefa tediosa para um grande número de

usuários o que pode ocasionar a saturação dos dispositivos de encaminhamento.

Ubale *et al.* [83] criaram uma solução em que o controlador analisa o volume de pacotes SYN recebidos para identificar possíveis ataques à rede e bloquear o número de entradas de fluxo instaladas no *switch*. O processo é feito da seguinte forma, assim, que um novo pacote ingressa na rede, a solução extrai o endereço IP e o MSS (*Maximum Segment Size*) do pacote e calcula um valor *hash*. Este valor definirá a ordem de retirada das regras de fluxos instaladas no *switch* para cada novo fluxo, no momento do ataque. O tamanho do valor do *hash* é definido com base no tamanho do MSS, pacotes com MSS menores, receberão valores menores. Se o volume de pacotes que chegam ao controlador exceder o limite predefinido, a solução remove as entradas cuja o valor de *hash* são menores e instala regras nos *switches* para o bloqueio dos fluxos. No entanto, o uso de MSS maiores pode fazer com que a solução classifique erroneamente fluxos espúrios em fluxos legítimos. Assim, o atacante pode sobrecarregar as entradas de encaminhamento do *switch*.

Dang *et al.* [84] propuseram uma solução que transforma o controlador em um *proxy*, porém sem armazenar o estado dos fluxos, ou seja, após o TCP *handshake*, as entradas de fluxo criadas dentro do *switch* são removidas rapidamente, o que contribui para reduzir o tempo médio das entradas de fluxos que ocupam os recursos do *switch*. Para detectar um ataque SYN *Flood*, a solução analisa o tempo de espera, desde o ponto em que o controlador (*proxy*) recebe a solicitação de conexão (SYN) até o ponto em que recebe o primeiro pacote de confirmação (ACK). Além disso, determina um tempo limite para que o controlador possa remover as conexões semi-abertas iniciadas devido as solicitações de conexões, liberando assim o recurso do servidor. O tempo limite das entradas de fluxo está relacionado ao tempo de espera do servidor. O tempo é definido com base nas características estatísticas de um serviço e no estado atual do servidor, no qual é analisado o número médio de conexões semi-abertas que o servidor processa durante um contexto de tráfego normal. Esse valor possui um coeficiente de ajuste dinâmico, em que aumentará ou diminuirá o tempo com base nas requisições recebidas, o que refletirá no tempo em que as entradas de fluxos serão removidas pela solução. Para conexões legítimas, ou seja, aquelas em que há o processo completo do TCP *handshake*, a solução migra os demais pacotes do fluxo para serem trocados diretamente entre o cliente o servidor sem a intervenção do *proxy*. Caso o fluxo seja considerado malicioso, ou seja, o *switch* não recebeu a confirmação de conexão após o intervalo de tempo predefinido, o controlador é alertado para descartar a conexão semi-aberta correspondente do fluxo no servidor e assim, instala uma regra para o descarte do pacote no *switch*. Durante um ataque, quando o atacante força o *switch* a instalar novas entradas de fluxo, a remoção rápida dessas entradas reduz o impacto do ataque na rede. Contudo, o atacante só precisa concluir a primeira conexão TCP para sobrecarregar a tabela de encaminhamento do *switch*.

3.1.2 Controle

As soluções que compreendem a classe de controle, procuram controlar (atrasar) o tráfego espúrio atribuindo a ele um valor de baixa prioridade, redução da largura de banda ou redução no número de mensagens enviadas ao controlador e não realizam nenhum bloqueio ou descarte do tráfego, como as soluções discutidas na classe anterior. Algumas abordagens relacionadas a essa classe são discutidas abaixo.

Piu *et al.* [85] propuseram um método de detecção usando o gráfico de controle CUSUM (Soma Cumulativa), para a detecção de alterações bruscas no fluxo de tráfego, com base nas estatísticas de rede dos *switches* OpenFlow. A solução consulta dados estatísticos em intervalos regulares de tempo dos *switches* para obter os valores dos pacotes recebidos. A partir desses valores é calculada a taxa de pacotes durante um intervalo de tempo determinado, após a coleta é aplicado o algoritmo CUSUM no conjunto de dados para detectar qualquer anomalia na rede. Sob condições normais, o algoritmo obtém valores negativos e positivos. Quando a taxa de pacotes aumenta abruptamente, o algoritmo começa a obter apenas valores positivos. Se o valor da soma acumulada exceder o limiar definido, há indícios de um ataque em andamento. Para reduzir os impactos causados pelo ataque, a solução monitora o número de pacotes que o *switch* encaminha para o controlador, desta forma ela define um contador no *switch* para contabilizar todos os pacotes recebidos e assim, coleta periodicamente esses valores e calcula a proporção entre o número total de pacotes e o número total de pacotes recebidos para cada intervalo, que não deve exceder o limiar definido. Se essa proporção de pacotes exceder o limiar, o controlador instrui o *switch* a enviar um número menor de pacotes, desta maneira a solução consegue atrasar alguns fluxos, enquanto os demais são atendidos pelo controlador. Todavia, a solução instrui o *switch* a enviar menos pacotes ao controlador, porém ela não realiza nenhuma distinção do tipo de fluxos a ser atrasado (legítimo ou espúrio), logo fluxos legítimos poderão ser atrasados por esta solução.

Kuerban *et al.* [86] propuseram o FlowSec para mitigar um ataque SYN *Flood*, aplicando um limite dinâmico de taxa no número de pacotes enviados ao controlador. A solução coleta as estatísticas de fluxos dos *switches*, com base na taxa de pacotes e *bytes*, para medir e controlar a taxa de pacotes enviados ao controlador. Com base nos resultados extraídos é definido o limiar de detecção, caso os fluxos ultrapassem o limiar estabelecido a solução inicia o processo para minimizar o impacto causado pelo ataque DoS. Calcula-se dinamicamente a largura de banda de controlador e assim, instrui a porta do *switch* responsável pelo volume de tráfego enviados ao controlador a diminuir sua velocidade. Este processo causa um atraso nos pacotes, evitando o estouro da tabela de fluxo e o gargalo na comunicação entre o plano de dados e controle. A solução reduz o número de fluxos enviados ao controlador, porém não realiza a distinção do tipo de fluxo

a ser atrasado, para priorizar os fluxos legítimos durante o processo de mitigação.

3.1.3 Gerenciamento de Recursos

As soluções que compreendem a classe de gerenciamento de recursos, tentam evitar um ataque DoS configurando e gerenciando a rede por meio de recursos adicionais presentes na infraestrutura da rede. Algumas delas são discutidas abaixo.

Drid *et al.* [12] incluíram uma solução chamada SDN-Guard para mitigar o impacto do ataque SYN *Flood* em uma rede SDN atuando em conjunto com um *Intrusion Detection System (IDS)*. A solução comunica-se constantemente com um IDS, que analisa as mensagens de pacotes e alerta a solução sobre a probabilidade de ameaça de cada novo fluxo. A probabilidade de ameaça é dada com base nas estatísticas dos fluxos coletados pelo IDS, assim a solução pode tomar uma decisão sobre o roteamento de cada um dos fluxos e o tempo limite para suas entradas na tabela de fluxo dos *switches*. Para fluxos considerados maliciosos, ou seja, se a probabilidade de ameaça estiver acima de um limiar predefinido, a solução atribui um valor de tempo limite grande e seleciona o enlace menos utilizado em termo de consumo de largura de banda para garantir que esse fluxo não concorra com fluxos legítimos e cause impacto em seu desempenho. Quando a probabilidade de ameaça do fluxo é baixa, ou seja, dentro do limiar predefinido, o fluxo é considerado legítimo. A solução então o encaminha pelo caminho mais curto e atribui um valor de tempo limite regular, permitindo que ele seja atendido mais rapidamente, de modo a garantir atrasos mínimos no atendimento dos fluxos, reduzindo os impactos causados por um ataque DoS. No entanto, a solução exige comunicação contínua com um agente externo (IDS) para obter as estatísticas da rede, o que pode causar um longo atraso na detecção do ataque.

Nugraha *et al.* [87] propuseram uma solução que atua em conjunto com uma ferramenta de amostragem *sFlow* para a detecção e mitigação do ataque SYN *Flood*. A solução gerencia o *sFlow* a partir do controlador, que realiza a captura de dados estatísticos de tráfego do dispositivo monitorado em um determinado intervalo de tempo. O algoritmo de detecção realiza uma soma de pacotes de ataques de forma cumulativa e assim, quando a soma estiver acima de um limiar há uma incidência de ataque. A soma é realizada da seguinte forma, a solução analisa o número total de *switches* presentes na rede, o número de pacotes em cada *switch* e o consumo médio de recursos do *switch*. Por exemplo, assumimos que, se houver 14.000 pacotes SYN chegando por segundo, considerando a rede com 4 *switches*, o sistema detectará o ataque quando a soma dos *switches* participantes ultrapassar 14.000, o que é muito mais rápido e consome menos recursos de cada *switch* em comparação com uma abordagem tradicional. Após o ataque ser detectado, o controlador aplica uma regra de bloqueio no *switch* para reduzir os impactos do ataque na rede. No entanto, a solução não apresenta como é feita a distinção de fluxos legítimos e espúrios,

logo pode gerar um grande número de falsos positivos (tráfego legítimo considerado como ataque).

3.2 Discussão

As soluções apresentadas na seção anterior representam o estado da arte com relação as principais soluções de segurança, que propõem métodos para reduzir os impactos causados por um ataque SYN *Flood* em uma rede SDN. Embora, elas funcionem bem para o propósito para o qual foram projetadas, ainda podem haver pontos fracos nessas soluções, principalmente devido a maneira como lidam com tráfego espúrio. Podendo afetar a funcionalidade, a eficiência e o custo da rede. Uma solução pode ter um ou mais pontos fracos, que são principalmente relacionados:

- **Atraso:** é o tempo exigido pela solução para reduzir os impactos causados por um ataque DoS. Em geral, esse atraso está ligado a maneira como é realizado o processo de mitigação, por exemplo, migração de conexão ou sistema de *proxy*.
- **Sobrecarga:** está relacionada ao consumo intensificado dos recursos computacionais (CPU e memória) do controlador e dos *switches*, além do uso normal para o encaminhamento de pacotes. Soluções que realizam detecção baseada em anomalias, que necessitam de uma comunicação contínua e coleta de dados para analisar o comportamento da rede, podem acarretar em um consumo além do previsto.
- **Equipamento adicional/especial:** algumas soluções de mitigação podem exigir o uso de equipamentos adicionais ou especiais. Este equipamento pode ser do mesmo tipo que já é usado na infraestrutura da rede, como controlador, *switches* e roteadores ou pode ser de algum tipo especial, como módulos adicionais ou um novo componente, que fique entre o controlador e os *switches*. Esse equipamento pode aumentar a complexidade e o custo da rede em geral.
- **Perda de pacotes legítimos:** as soluções podem produzir alertas falsos, como resultado alguns pacotes legítimos podem ser descartados. Isso afetará a confiabilidade da solução proposta. Taxa de limitação, atraso e o uso de limiares de detecção são os principais motivos para a perda de pacotes legítimos.

A Tabela 3.1 mostra a comparação de todas as soluções discutidas anteriormente. Para proteger a rede de diferentes maneiras elas podem exigir algum custo, como atraso, sobrecarga computacional, equipamento adicional/especial ou perda de pacotes legítimos.

Cerca de 45% das soluções apresentam um atraso considerado médio. Tal fato ocorre, quando adicionamos qualquer processamento além do padrão da rede SDN para a mani-

Tabela 3.1: Comparação das soluções de mitigação

Soluções		**Camada SDN**	**Atraso**	**Sobre-carga**	**Equipamento adicional/ especial**	**Perda pacotes legítimos**
Bloqueio	Shin *et al.* [15]	Dados	alto	alta	sim	baixa
	Sonchak *et al.* [81]	Dados	alto	média	não	média
	Fichera *et al.* [14]	Controle	alto	alta	não	baixa
	Moreno *et al.* [82]	Dados	médio	alta	sim	baixa
	Reza *et al.* [13]	Controle	baixo	média	não	média
	Dang *et al.* [84]	Controle	baixo	média	não	média
	Ubale *et al.* [83]	Controle	médio	alta	não	média
Controle	Piu *et al.* [85]	Controle	médio	alta	não	média
	Kuerban *et al.* [86]	Controle	médio	alta	não	média
Gerenciamento Recursos	Drid *et al.* [12]	Controle	alto	alta	sim	média
	Nugraha *et al.* [87]	Controle	médio	alta	sim	média

pulação dos fluxos. Porém, quando considerado muito alto, principalmente em soluções que realizam a intermediação entre origem e destino, pode fazer com que a solução leve um tempo além do normal para detectar possíveis mudanças no tráfego ou apenas para encaminhar uma solicitação normal de conexão com o destino. Porém, esta intermediação garante que a solução tenha uma perda menor de pacotes legítimos, quando comparado com soluções que utilizam limiares de detecção, estas por sua vez, acabam exigindo uma comunicação intensa entre *switch* e controlador, principalmente para a coleta de dados estatísticos.

Das soluções que atuam no plano de controle, cerca de 47% delas, requerem uma comunicação continua com o plano de dados para obter informações da rede, e assim, estabelecer meios para reduzir os impactos causados pelo ataque SYN *Flood*, como o uso de limiares de detecção. No entanto, a comunicação excessiva entre os planos (dados e controle) pode resultar em atrasos (gargalos), tempo de processamento mais elevados e sobrecarga ao controlador que, por sua vez, podem afetar o tempo necessário para detectar um ataque e tomar as medidas necessárias. Uma alternativa é o desenvolvimento de soluções que atuem diretamente no plano de dados, de modo a reduzir a comunicação excessiva com o controlador e que tome as ações necessárias diminuindo o tempo de processamento e a sobrecarga do controlador para detectar eventuais mudanças na rede.

Podemos observar também nas soluções propostas que o fluxo uma vez validado não passa por nenhum tipo de revalidação, situação está em que a solução pode ficar vulnerável a um novo tipo de ataque, já que o atacante pode utilizar um fluxo legítimo para causar saturação na rede. Outro ponto a ser destacado é a ausência de métodos que priorizem o tráfego legítimo nos dispositivos de encaminhamento, ou seja, não há uma manipulação do tráfego nos dispositivos de encaminhamento que possa realizar o gerenciamento de maneira ágil dos fluxos recebidos. Esta ausência, promove com que algumas soluções

fiquem expostas aos ataques DoS, como a solução proposta por Shin *et al.* [15], que atua no plano de dados com o objetivo de proteger o controlador. Desta forma, os ataques DoS ainda podem ocasionar gargalos na comunicação entre o plano de dados e plano de controle, exaurindo a capacidade de processamento dos dispositivos de encaminhamento. Já que não há uma distinção de fluxos legítimos e espúrios e todos seguem pela mesma fila com a prioridade equivalente. Fato este que não confere qualquer vantagem aos fluxos legítimos. Logo, são necessários meios para revalidar a conexão e gerenciar os fluxos, assegurando com que o tráfego legítimo seja atendido com prioridade, reduzindo assim, o congestionamento de fluxos nos dispositivos de encaminhamento, ocasionados por um ataque DoS.

3.3 Considerações Finais

Este capítulo apresentou o estado da arte acerca das soluções de segurança em redes SDN com ênfase no ataque SYN *Flood*, que propõem alternativas para reduzir os impactos causados por este ataque. Foram apontadas algumas lacunas existentes na literatura, que podem deixar a rede vulnerável a este tipo de ataque, causando falhas na infraestrutura de rede. No capítulo seguinte será apresentada uma proposta de detecção e mitigação, a fim de preencher as lacunas encontradas e aumentar a segurança da rede.

Capítulo 4

Proposta de Detecção e Mitigação

Neste capítulo, apresentamos a solução proposta de detecção e mitigação chamada DoS-SEC (do inglês, *Denial of Service Security*), desenvolvida integralmente no plano de dados. Em contrapartida com outras soluções apresentadas no capítulo anterior, a solução reduz a sobrecarga do controlador e a comunicação extensiva entre este e o plano de dados para obter informações da rede. Em particular, começamos a discutir os elementos fundamentais da solução proposta, como é realizado o processo de detecção utilizando técnicas de análises estatísticas. Em seguida, apresentamos o processo de mitigação, para reduzir os impactos causados pelo ataque SYN *Flood*.

4.1 Proposta de Trabalho

O paradigma SDN trouxe inúmeros benefícios, como o controle centralizado da rede e a capacidade de operá-la de acordo com às necessidades do usuário [1]. Porém, a segurança em uma rede SDN ainda é motivo de preocupação [18]. Os ataques de negação de serviço (DoS) são um dos principais desafios dessa rede, devido as vulnerabilidades existentes entre as camadas. Embora, existam inúmeras soluções, como SDN-Guard [12], Slicots [13], Operetta [14] e Avant-Guard [15] que procuram reduzir os impactos causados por este tipo de ataque, ainda permanecem algumas lacunas que podem deixar a rede suscetível a este tipo de ataque.

A proposta deste trabalho é detectar e mitigar o ataque SYN *Flood* em uma rede SDN. Este tipo de ataque possui como característica básica, o envio de um grande volume de tráfego para a vítima em um espaço curto de tempo [25]. Um dos grandes problemas enfrentados pelas soluções de segurança em SDN consiste em distinguir durante esse tipo de ataque, o tráfego legítimo do tráfego espúrio de maneira rápida e precisa, tendo em vista o volume de tráfego analisado, sem que haja uma carga excessiva de trabalho nos dispositivos que compõem a rede [65]. O emprego de métodos de análise estatística, como

qui-quadrado e entropia, são exemplos de técnicas capazes de auxiliar na identificação de desvios no padrão de comportamento do tráfego que possam caracterizar um ataque DoS, apresentando resultados satisfatórios para esse fim [88].

A proposta de detecção do ataque SYN *Flood* consiste em utilizar uma técnica de análise estatística. A partir das estatísticas geradas pelos *switches* que compõem a rede, pode-se observar o comportamento do tráfego e assim detectar eventuais ataques. Após a detecção é preciso tomar medidas para reduzir os impactos causados por esse tipo de ataque. Uma das lacunas encontradas durante a revisão do estado da arte acerca das soluções de segurança em SDN é a ausência de métodos que priorizem o tráfego nos dispositivos de encaminhamento, o que pode deixar a rede suscetível a ataques DoS agora voltados para exaurir a capacidade de gerenciamento de fluxos nos dispositivos de encaminhamento. Uma das soluções analisadas foi o trabalho proposto por Shin *et al.* [15] que oferece proteção à rede, mediante uma validação intermediária de conexão entre origem e destino antes de encaminhá-la ao seu destino final. Apesar de proteger o controlador, no entanto a ausência de métodos que possam priorizar o tráfego legítimo nos dispositivos de encaminhamento pode provocar um congestionamento de fluxos. Logo, são necessários meios para gerenciar os fluxos e fazer com que o tráfego legítimo seja atendido de forma prioritária.

A proposta de mitigação, consiste em aplicar um método baseado em reputação, que utiliza filas para denotar o nível de confiança dos fluxos. Assim, será possível classificar e atribuir uma prioridade para cada novo fluxo que ingressa na rede. Desta forma, os fluxos validados terão prioridades mais altas e serão movidos para uma fila apropriada, enquanto os demais serão deslocados para filas com prioridades menores. Deste modo, garantimos com que os fluxos legítimos sejam os primeiros atendidos pela solução, reduzindo o congestionamento de fluxos e minimizando os impactos causados pelo ataque SYN *Flood*.

O DoSSEC é formado por dois componentes: detecção e mitigação. A funcionalidade de detecção coleta as estatísticas de fluxos e fornece como saída as alterações no volume do tráfego que indicam um possível ataque em andamento, para que as medidas de mitigação reduzam o impacto do ataque. A aplicabilidade de mitigação, tem como objetivo priorizar os fluxos já validados por meio de um método baseado em reputação, que utiliza filas nos *switches* para denotar o nível de confiança dos fluxos. A descrição detalhada da solução proposta será apresentada nas subseções a seguir.

4.1.1 Etapa de Detecção do Ataque

A etapa de detecção do ataque é realizada com base nas estatísticas geradas pelos *switches* que compõem a rede, para tal implementamos uma arquitetura capaz de permitir que

diferentes métodos estatísticos possam ser aplicados, como qui-quadrado e entropia, por exemplo. Tais técnicas permitem caracterizar eventuais ataques à rede de maneira rápida e eficaz, ou seja, com baixa complexidade computacional para que possa ser realizado em uma rede em produção [46] [89].

Para o desenvolvimento da solução diretamente no plano de dados, utilizamos a linguagem P4, que permite adicionar novos cabeçalhos e especificar como os pacotes devem ser analisados e processados pelos *switches*, transformando o *switch* em um dispositivo totalmente programável [4]. No entanto, as soluções que adotam esta linguagem ainda precisam lidar com um conjunto restrito de funcionalidades para análise do tráfego, tornando limitado o escopo das soluções de segurança que estão sendo desenvolvidas no plano de dados [90] [91]. Uma das limitações é o não suporte a operações matemáticas que resultam em números de ponto flutuante. Para superar essas restrições, propusemos o uso de um componente auxiliar ao lado do *switch* programável para operações de ponto flutuante que retorna as informações referentes aos fluxos para o *switch* P4 em tempo real. Para esse fim, a estrutura do Apache Thrift [92] foi utilizada, que permite a comunicação entre diferentes linguagens de programação via RPC (*Remote Procedure Call*).

O Apache Thrift atua como uma ponte de *software*, enviando instruções que são convertidas em ações a serem interpretadas pelo *switch* P4 em tempo de execução. A Figura 4.1 mostra a arquitetura proposta. O *switch* programável está conectado à estrutura do Apache que executa cálculos estatísticos e também relacionados às filas e retorna essas informações ao *switch* que gera uma API para preencher tabelas e campos correspondentes dos pacotes recebidos. O P4 OpenFlow Agent mapeia essas tabelas para tabelas OpenFlow, juntamente com seus campos de correspondência. Dessa forma, é possível promover a comunicação com o controlador via protocolo OpenFlow e, assim, combinar *switches* programáveis com *switches* de função fixa à rede. Observe que esse suporte ao protocolo OpenFlow aumenta a variedade de dispositivos que podem se beneficiar da arquitetura proposta.

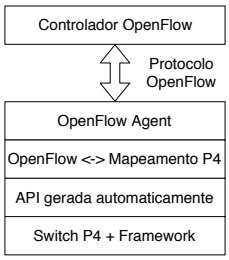


Figura 4.1: Arquitetura de Detecção

Para o desenvolvimento da etapa de detecção utilizando técnicas de análise estatísticas é necessário definir quatro elementos: campo a ser observado, janela de observação,

método estatístico e o limiar de detecção.

O campo a ser observado refere-se aos campos de cabeçalho disponíveis do fluxo, que podem incluir IP de origem, IP de destino, porta de origem, porta de destino, tamanho do pacote e assim por diante. A partir deste campo, é possível extrair informações do cabeçalho do pacote do fluxo que permitam caracterizar o tráfego.

A janela de observação corresponde ao número de pacotes (tamanho da janela l) que serão analisados pela solução. Dentro de cada janela de observação w_i, a frequência com que o campo a ser observado é calculada. A probabilidade de ocorrência do campo observado é medida da seguinte forma:

$$p_j = \frac{F_j}{l}, \tag{4.1}$$

onde F_j, $(1 \leq j \leq l)$, é definido como a frequência do campo observado em w_i, $|w_i| = l$ é o número de pacotes em w_i e p_j é a probabilidade de ocorrência do campo em w_i.

O método estatístico consiste em aplicar um dos métodos definidos pelo administrador da solução: qui-quadrado ou entropia, por exemplo. Nesta etapa, os cálculos são realizados de acordo com a Equação (2.1) ou Equação (2.2) nos dados contidos na janela de observação w_i. Os valores calculados são armazenados em uma lista que contém os valores calculados anteriormente. Esses valores servirão de base para definir o limiar de detecção.

O limiar de detecção T_{w_i}, é utilizado para detectar algum comportamento fora dos padrões estabelecidos e assim informar a possível existência de um ataque. O cálculo do limiar é dado como:

$$T_{w_i} = \begin{cases} T_{w_i}, & i = 1 \\ T_{w_i}\alpha + (1-\alpha)T_{w_{i-1}}, & i > 1 \end{cases} \tag{4.2}$$

onde T_{w_i} representa o valor do método aplicado (qui-quadrado ou entropia) calculado, T_{w-1} representa o valor obtido no período anterior w_1, onde $i > 1$ é o índice que representa a janela de observação em condições normais (sem ataque) w e α é o coeficiente de suavização. O coeficiente α assume valores entre $(0 \leq \alpha \leq 1)$ de forma que quanto mais próximo de 1, maior é o peso dado às informações recentes e quanto mais próximo de 0 menor é o peso dado às informações recentes e maior é o peso dado às informações passadas. Dessa maneira, o limiar pode ser ajustado de acordo com as características da rede para reduzir a incidência de alarmes falsos. Vale ressaltar que o limiar para T_{w_i} é aplicado somente quando o T_{w-1} não é classificado como ataque. Do contrário, a janela atual não é considerada para o cálculo.

Com base nos elementos discutidos acima, apresentamos o modelo estatístico de detecção na Figura 4.2 presente no plano de dados. De início à medida que os pacotes

ingressam na fila de entrada do *switch*, é extraído o campo a ser observado pela solução proposta, que corresponde ao campo de cabeçalho escolhido para análise (IP de origem), para compor a janela de observação w_i ($i \geq 1$). Assume-se que na primeira janela não há ataque. Quando a janela de observação é preenchida, ou seja, quando $|w_i| = l$, onde l é o tamanho da janela de observação, a frequência do campo observado em w_i é calculada de acordo com a Equação (4.1). Depois disso, é calculado a pontuação do método estatístico (α_{w_i}: qui-quadrado (Equação.(2.1)) ou entropia (Equação.(2.2))) para a janela de observação $|w_i|$. Em seguida, o limiar de detecção é calculado T_{w_i}, de acordo com a Equação (4.2) para avaliar o comportamento da rede. Se o valor do método estatístico da janela de observação atual estiver abaixo do limiar de detecção, o tráfego será considerado legítimo; caso contrário, há uma forte indicação de um possível ataque em andamento, sendo acionado o alarme de detecção para que as ações de mitigação sejam realizadas.

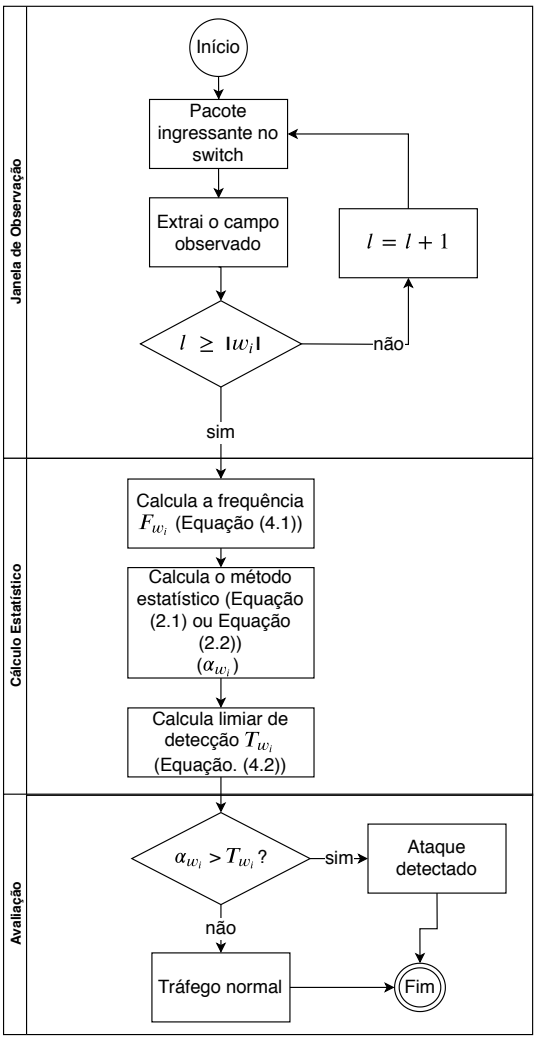


Figura 4.2: Modelo de detecção no *switch*

No curso de um ataque SYN *Flood*, esperamos que o valor do método estatístico

proposto saia da sua distribuição normal ultrapassando o limiar estabelecido, conforme os pacotes espúrios comecem a ocupar as janelas de observação.

4.1.2 Etapa de Mitigação do Ataque

A etapa de mitigação tem como objetivo reduzir os impactos causados pelo ataque SYN *Flood*, preservando e priorizando um conjunto benigno do tráfego, através de um método baseado em reputação, que utiliza filas para denotar o nível de confiança dos fluxos no *switch* para garantir a qualidade de serviço (do inglês, *Quality of Service - QoS*) e melhorar o desempenho da rede.

O DoSSEC classifica e atende os fluxos que ingressam na rede com base em sua prioridade atribuída, no contexto da linguagem P4. O gerenciamento de filas do *switch* P4 é similar ao dos *switches* que utilizam o protocolo OpenFlow [2] para a manipulação de filas, em que uma ou mais filas podem ser configuradas para cada interface de saída e usadas para mapear entradas de fluxos nelas. As entradas de fluxos mapeadas para uma fila específica, serão tratadas de acordo com a configuração da fila em termos de taxa de serviço.

O *switch* P4 realiza a manipulação de informações através de metadados, que carregam informações adicionais no fluxo entre os estágios do *pipeline* do *switch* [4]. Estão disponíveis até 8 níveis de prioridades, onde 0 é a prioridade mais baixa e 7 a prioridade mais alta para a fila [17]. Cada fila possui um identificador e uma prioridade que pode ser atribuída, no qual os fluxos seguirão estritamente as marcações indicadas de acordo com a identificação da fila dentro do *pipeline* do *switch*. A seguir, é apresentada a motivação da solução proposta, através de alguns casos de entrada para melhor compreensão do funcionamento do DoSSEC.

Motivação

O DoSSEC pode ser classificado em duas classes de mitigação: controle (atraso) e bloqueio. Optamos por não eliminar todo o tráfego proveniente da origem do ataque para garantir que os fluxos considerados espúrios, mas que na verdade são legítimos, tenham a chance de alcançar o seu destino, porém com um pequeno atraso com relação aos fluxos já validados.

O atraso é em decorrência da redefinição das solicitações de conexão TCP para os novos fluxos quando a rede está sob ataque. Este processo pode ocasionar um leve atraso, porém ele não dá a oportunidade de fluxos provenientes do ataque chegarem ao controlador, fazendo com ele não instale regras de encaminhamento desnecessárias e consuma os recursos computacionais processando os fluxos. Como resultado da ação, o ataque não

causará falhas na infraestrutura da rede, o que permitirá preservar um conjunto benigno do tráfego da rede.

O bloqueio é realizado de forma temporária, apenas para os fluxos que apresentarem comportamento fora dos padrões estabelecidos dentro das filas. Isto, dá a oportunidade de o fluxo ser atendido novamente pela solução, na situação que seja um fluxo legítimo e tenha sido penalizado de forma indevida.

Ao propormos um gerenciamento de filas no *switch*, atribuímos prioridades aos fluxos e garantimos o atendimento de acordo com a precedência estabelecida. Portanto, asseguramos que o tráfego legítimo seja atendido de maneira prioritária, reduzindo assim o congestionamento de fluxos nos *switches*, garantindo a qualidade do serviço à rede.

Por fim, a adoção da solução no plano de dados reduz a comunicação contínua do controlador com os *switches* para a obtenção de dados estatísticos para a etapa de detecção. Com isso, temos uma detecção com atrasos menores, o que reduz significativamente o tempo e o volume de pacotes a serem analisados para detectar qualquer mudança no tráfego de rede.

A seguir, são apresentados alguns casos de entrada para melhor compreensão do funcionamento do DoSSEC.

- **Estabelecimento de conexão com IP legítimo:** A Figura 4.3 mostra as funcionalidades do DoSSEC para um fluxo legítimo. Ao tentar estabelecer uma conexão com o servidor TCP, primeiramente o pacote de abertura de conexão do fluxo (SYN) é enviado para o destino (servidor), ele é encaminhado primeiramente pelo gerenciamento padrão de filas do *swtich*, sendo categorizado inicialmente como Não Classificado. A partir de então, a solução realiza o procedimento de acompanhamento da conexão, que é responsável por observar e classificar as solicitações de conexão dos fluxos. Por se tratar de um novo fluxo, não haverá nenhuma entrada correspondente na tabela de fluxo do *switch*, logo ele é encaminhado ao controlador que calcula a melhor rota e instala uma regra de encaminhamento no *switch*, contendo uma ação indicando o percurso que o fluxo deverá seguir. O pacote é então encaminhado ao servidor, que responde com um SYN-ACK e aguarda o pacote ACK para estabelecer a conexão. Estabelecida a conexão, o fluxo tem a sua conexão classificada em sucesso e é vinculado a *Graylist*, no qual é atribuído o nível de confiança intermediário. Desta forma as novas solicitações do fluxo serão atendidas diretamente pela fila em questão com uma prioridade associada. Ao manter um bom comportamento, por realizar inúmeras conexões com sucesso, o fluxo pode ser promovido a *Whitelist*, que indica o nível de confiança mais alto adotado pela solução;

- **Tentativa de estabelecer conexão com IPs espúrios:** A Figura 4.4 mostra as funcionalidades do DoSSEC para impedir a tentativa de um ataque SYN *Flood*. O

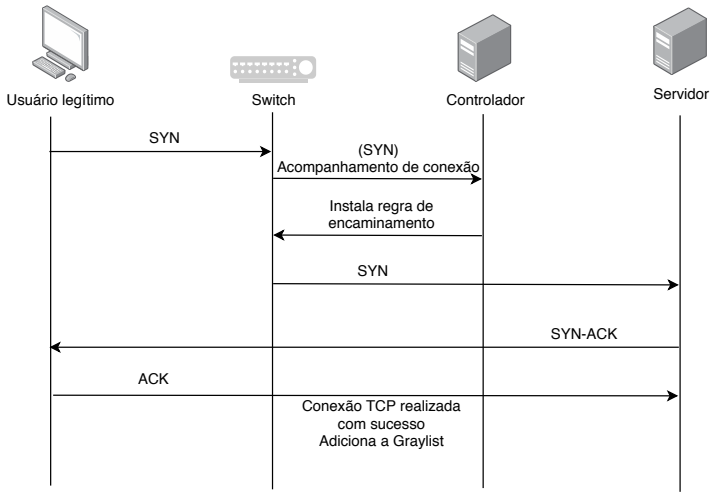


Figura 4.3: Estabelecimento de conexão com IP legítimo

atacante envia um grande volume de pacotes para o estabelecimento de uma conexão com o servidor, por meio de endereços espúrios que não correspondem a nenhum fluxo já definido, ou seja, que não tenha passado pela rede. O limiar de detecção é acionado ao notar um comportamento fora do padrão para o tráfego legítimo. Assim, as solicitações de conexões dos novos fluxos são redefinidas (atrasadas) pela solução, impedindo que elas cheguem ao controlador e consuma recursos desnecessários da rede. As novas solicitações não são colocadas em nenhuma fila de prioridade. Devido a característica do ataque, estes endereços IPs não retornam a rede, fazendo com que novas entradas de encaminhamento na tabela do *switch* não sejam instaladas, garantindo com que os recursos da rede sejam preservados pela solução e assim priorizar os fluxos já validados.

- **IP espúrio disfarçado de legítimo:** A Figura 4.5 mostra as funcionalidades do DoSSEC para impedir a tentativa de um ataque SYN *Flood*, no qual o atacante utiliza um IP legítimo e partir dele inicia o ataque com o objetivo de exaurir os recursos dos dispositivos de encaminhamento. O atacante estabelece uma conexão com o servidor TCP e é vinculado inicialmente a *Graylist*. Estabelecida a conexão, o fluxo inicia o ataque enviando um grande volume de pacotes de abertura de conexão (SYN) em direção ao servidor. Ao identificar um comportamento fora do padrão estabelecido dentro da fila para os fluxos, ele tem a sua conexão bloqueada de forma temporária, ou seja, é adicionado a *Blacklist*. Desta forma, as suas solicitações de abertura de conexão serão descartadas de forma imediata pela solução, aliviando os recursos da rede.

A seguir, é apresentado como é realizado o método de gerenciamento de filas no *switch*

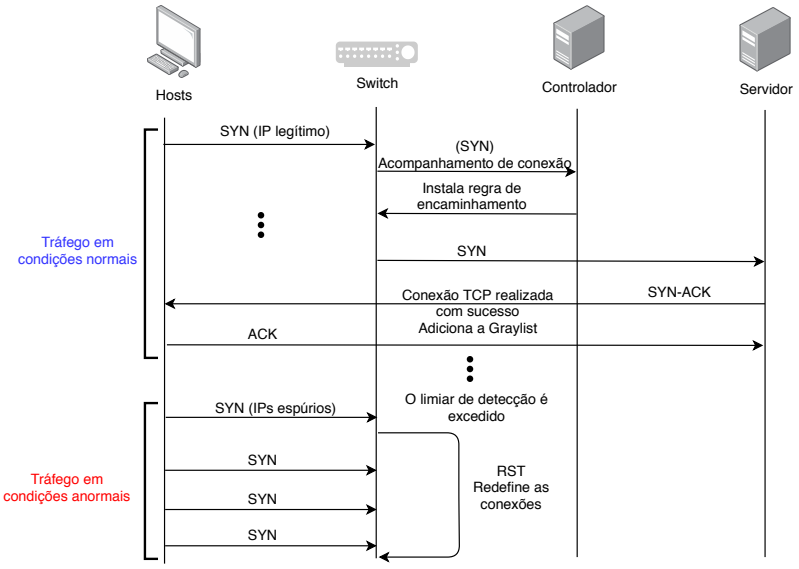


Figura 4.4: Tentativa de estabelecer conexão com IPs espúrios

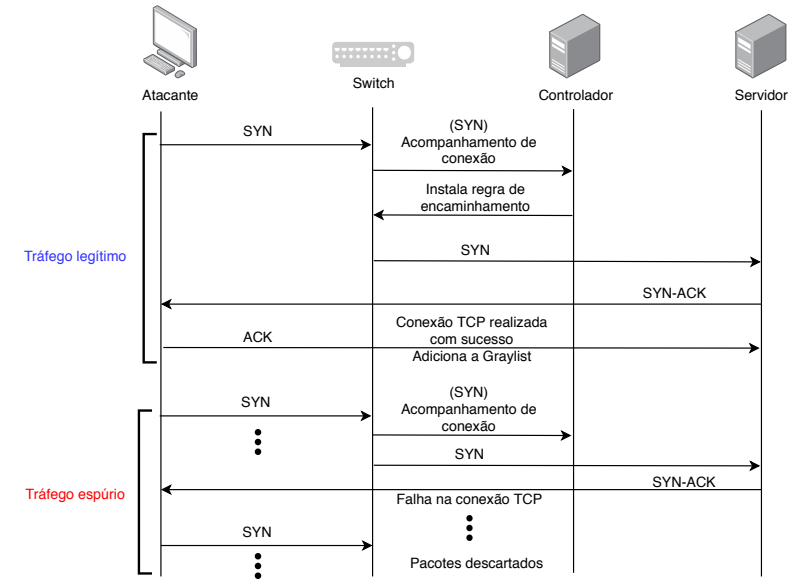


Figura 4.5: Tentativa de iniciar o ataque a partir de um IP legítimo

P4 para a solução proposta, de modo a atribuir os fluxos as suas respectivas filas de acordo com o nível de confiança estabelecido.

Níveis de confiança dos fluxos

O DoSSEC opera sobre os seguintes tipos de níveis de confiança: *Whitelist*, *Graylist*, *Blacklist* e Não Classificado.

- ***Whitelist:*** retrata os fluxos que possuem o nível de confiança mais alto, nos quais são atribuídos a prioridade mais alta, o que lhes permitem serem atendidos primeiro na rede em detrimento dos demais fluxos. Este nível é conferido aos fluxos considerados mais regulares e que possuem um bom comportamento na rede ao longo do seu período de atividade, de acordo com os parâmetros definidos pela solução para o gerenciamento das filas. Os fluxos pertencentes a esta classe são identificados pela solução através do campo de cabeçalho referente ao Tipo de Serviço (*Type of Service* - ToS), marcado com o valor 7.

- ***Graylist:*** corresponde aos fluxos que possuem o nível de confiança intermediário, ou seja, aqueles que tiveram um bom comportamento inicialmente na rede. Eles possuem uma prioridade mais baixa com relação a *Whitelist*, tendo o campo de cabeçalho ToS, definido com o valor 6.

- ***Blacklist:*** retrata os fluxos não confiáveis, ou seja, bloqueados pela solução. Aqueles que tiveram comportamentos fora dos padrões estabelecidos pela solução, considerados irregulares. Os fluxos pertencentes a este nível, não são atendidos pela solução e não possuem nenhuma prioridade associada. Eles detêm o campo de cabeçalho ToS, marcado com o identificador 0. Em especial, para a solução proposta, este identificador não terá nenhuma prioridade associada e servirá apenas para categorizar os fluxos pertencentes a *Blacklist*.

- **Não Classificado**: retrata os fluxos considerados novos pela solução, ou seja, aqueles que ainda não realizaram nenhuma solicitação de conexão TCP com o destino. Para este nível, os fluxos não possuem nenhuma prioridade associada, isto é, nenhum nível de confiança foi estipulado pela solução.

A Figura 4.6 mostra o processo de transição entre os níveis de confiança dos fluxos, de acordo com o seu comportamento na rede. Os fluxos novos que ingressam a rede são tratados previamente pela solução como Não Classificado, o que significa que eles não possuem nenhuma prioridade associada a priori. O fluxo pertencente a esta categoria é encaminhado primeiramente ao seu destino através do gerenciamento padrão de filas do *switch* P4, sem nenhuma prioridade relacionada. A prioridade só é concebida, a partir do momento em que o fluxo completa a sua primeira conexão TCP com o seu destino, no qual passará a compor inicialmente a *Graylist*. A partir de então, ele pode ser transferido para o nível de maior confiabilidade (*Whitelist*) em decorrência de um bom comportamento. Pode ser bloqueado, ou seja, movido para (*Blacklist*), quando tem o seu comportamento caracterizado como irregular pela solução ou pode simplesmente perder a sua classificação inicial, sob a condição de fluxo suspeito voltando a condição de Não Classificado. Ao ser migrado para a *Whitelist*, ele também está sujeito a perder o nível de confiança que lhe

foi atribuído, com base em seu comportamento podendo retornar a categoria de Não Classificado ou ser bloqueado pela solução (*Blacklist*). Por fim, o fluxo transferido para a *Blacklist*, é bloqueado por um período de tempo, o que lhe permite retornar a condição de Não Classificado, após ultrapassar o tempo limite de permanência definido pela solução.

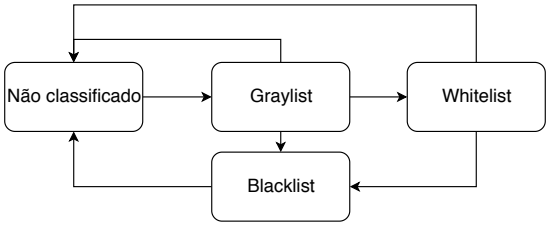


Figura 4.6: Transição dos fluxos entre os níveis de confiança

Os níveis de confiança possibilitam a solução categorizar o fluxo, e conceder prioridade no seu atendimento em detrimento dos demais fluxos presentes na rede. Este processo atua em conjunto com a etapa de detecção, para preservar os recursos da rede e garantir à qualidade do serviço. A Figura 4.7 mostra como as etapas de detecção e mitigação cooperam entre si. Inicialmente, ao receber uma solicitação de conexão TCP do fluxo f_i, é verificado qual o nível de priorização desta solicitação de acordo com o nível de confiança estabelecido para o fluxo. Se o fluxo pertencer a *Blacklist*, ele tem a sua conexão descartada de forma imediata pela solução, sendo posteriormente analisado pelo procedimento da *Blacklist*. Do contrário, é verificado qual o nível de confiabilidade do fluxo para as filas de prioridade *Whitelist* ou *Graylist*, respectivamente, para estabelecer a sua ordem de atendimento (encaminhamento), com relação a sua solicitação de conexão. A solicitação de conexão é observada por meio do procedimento de acompanhamento de conexão, que é responsável por classificar o tipo de conexão estabelecida para a solicitação do fluxo. Se a solicitação de conexão do fluxo não pertencer a nenhuma das filas de prioridade, trata-se de uma solicitação de conexão TCP de um novo fluxo a rede. Para este cenário, há dois tipos de tratamentos a serem realizados, conforme os resultados dos limiares de detecção. O primeiro cenário refere-se quando a rede não está sob ataque, neste caso a solicitação do fluxo é encaminhada normalmente ao seu destino sem nenhuma prioridade relacionada, sendo observada pelo procedimento de acompanhamento de conexão. O segundo cenário retrata quando a rede está sob ataque, para esta situação a primeira tentativa de estabelecer uma conexão TCP do fluxo com o destino é sempre rejeitada (redefinida) pela solução. No qual, a solução envia um pacote de redefinição (RST) para o fluxo em questão, o que faz com que conexão seja anormalmente fechada, e isso permite com que a solicitação de conexão do fluxo possa retornar a rede. Ao redefinir a conexão, a solução carimba a solicitação de conexão do fluxo em uma lista de conexões pendentes, que contém as conexões não concluídas para os fluxos e aguarda o seu retorno. Desta forma, é possível

identificar se a solicitação de conexão do fluxo refere-se a uma solicitação já redefinida ou não pela solução. Ao retornar a rede durante o ataque, ele é retirado da lista de conexões pendentes e tem a sua solicitação de conexão TCP encaminhada normalmente ao seu destino, observada pelo procedimento de acompanhamento de conexão.

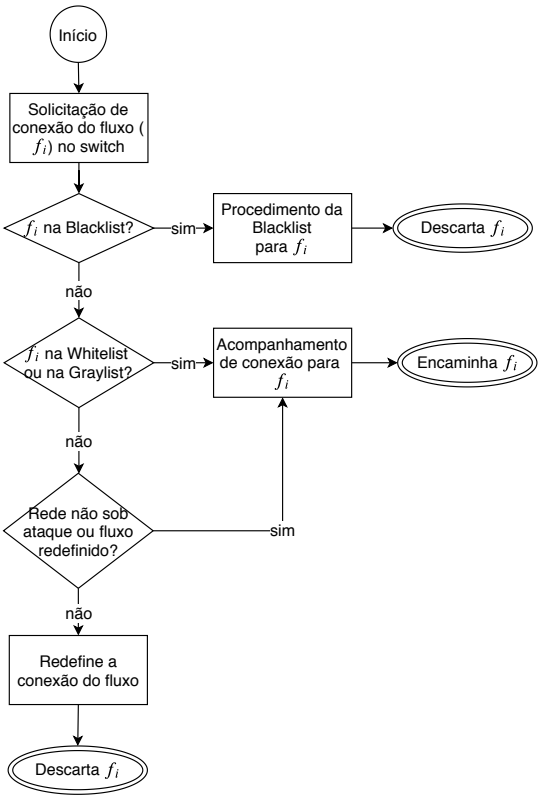


Figura 4.7: Detecção e Mitigação

Acompanhamento de Conexão

O acompanhamento de conexão, tem como objetivo observar as solicitações de conexão TCP dos fluxos e classificá-las em sucesso ou falha. A classificação é realizada através do acompanhamento das *flags* TCP, trocadas entre origem e destino para o estabelecimento de conexão do fluxo (*three-way handshake*). Para tal, os seguintes campos são analisados: As *flags* (SYN, SYN-ACK e ACK), IP de origem, IP de destino, porta de origem e porta de destino de forma a identificar o fluxo solicitante.

No sentido de analisar o processo do *three-way handshake* do fluxo, precisamos conhecer o período de tempo limite típico de estabelecimento de uma conexão TCP, considerando um ambiente comum sem a presença de fluxos espúrios na rede. Analisou-se então, o tempo de espera desde o ponto em que o *switch* recebe o primeiro pacote SYN, até o ponto em que recebe o primeiro pacote ACK para concluir a conexão TCP. Para então

definir o tempo médio de espera e assim, concluir se a conexão foi de fato estabelecida ou não. O tempo do fluxo no *switch* é dado pela seguinte equação:

$$T(f_i) = T_{stamp}(f_i) - T_{in}(f_i), \tag{4.3}$$

onde T_{stamp} indica o tempo atual do sistema e T_{in} representa o tempo em que o primeiro pacote ACK do fluxo f_i é registrado pelo *switch*.

A Figura 4.8 mostra quais os tratamentos a serem realizados, de acordo com o resultado do procedimento de acompanhamento de conexão para o fluxo. De início, ao receber a solicitação de conexão TCP do fluxo f_i, o procedimento de acompanhamento de conexão estabelece o tempo médio de espera e verifica se a conexão será estabelecida ou não com o destino. Se o fluxo não enviar o pacote ACK dentro do tempo limite estabelecido, indica que a conexão não foi estabelecida. Para este cenário, o fluxo tem o seu respectivo contador para o número de conexões não estabelecidas S_{f_i}, atualizado e comparado ao limiar β, que é escolhido de forma a identificar uma troca entre desempenho e eficiência. De fato, a escolha de um valor grande para o limiar β resultaria em um atraso na detecção de um possível ataque e, portanto, na eficiência precária. Ao contrário, ao escolher um valor muito baixo de β, isso causaria uma detecção precoce de um fluxo que poderia ser legítimo. Pois, o ACK pode ser perdido devido a muitas razões, por exemplo, congestionamento, tempo limite decorrido, perda de conectividade e entre outros. Consequentemente, se o limiar for muito baixo, o risco de expor a rede a muitos ataques é reduzido, melhorando assim o desempenho. Se o número de conexões não estabelecidas for menor que o limiar β, a conexão é simplesmente descartada pela solução, no qual o controlador é alertado para aplicar uma regra de descarte do pacote. Do contrário, as solicitações de conexões do fluxo serão bloqueadas, caracterizando a sua inserção na *Blacklist*. Contudo, se a conexão TCP do fluxo for realizada com sucesso e ele ainda não foi classificado ou está na *Graylist*, será atendido de acordo com o procedimento adotado para a fila em questão. Senão, terá o seu atendimento com base nos procedimentos da *Whitelist*.

Procedimento Graylist

O procedimento da *Graylist*, tem por objetivo atender e gerenciar os fluxos que possuem o nível de confiança intermediário. Os fluxos pertencentes a esta fila tem o seu comportamento analisado, através da taxa de pacotes e o grau de reputação da fila.

A taxa de pacotes da fila T_G, tem como meta detectar quando um fluxo utiliza de forma demasiada os recursos disponíveis da fila, causando a perda de pacotes dos demais fluxos presentes na fila. Assim, a taxa é calculada a cada intervalo de tempo, de acordo com a seguinte equação:

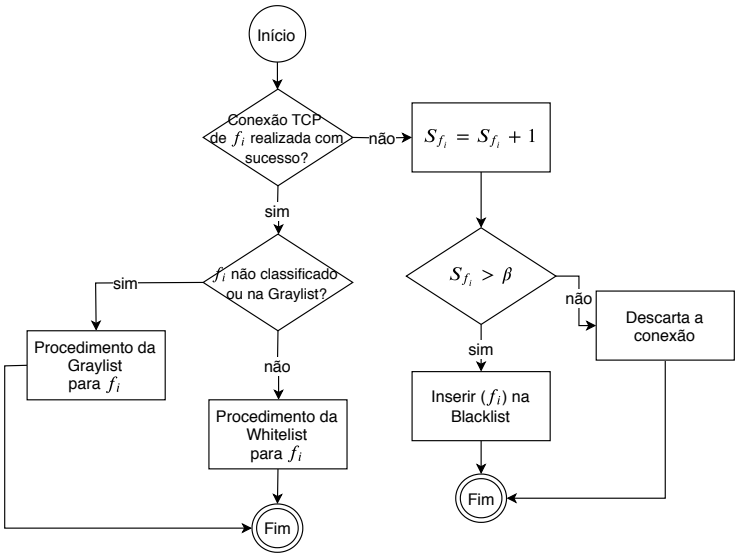


Figura 4.8: Acompanhamento de Conexão

$$T_G = \frac{\sum_{i=1}^{m} Tp_i}{m}, \tag{4.4}$$

onde m é a quantidade de fluxos na fila e Tp é o volume total de pacotes da fila. A definição de um intervalo de tempo excessivamente longo para as verificações pode possibilitar com que a fila fique muito sobrecarregada. Por outro lado, o uso de um tempo muito curto, exigirá mais processamento por parte da solução para a verificação das filas. Logo, é preciso propor um intervalo de tempo proporcione uma verificação de maneira justa, sem causar grandes sobrecargas a solução.

O grau de reputação R_G, permite avaliar o comportamento do fluxo ao longo do tempo em que se manteve presente na fila, permitindo com que o fluxo tenha o seu nível de confiança elevado. Ele leva em consideração duas premissas, o número de conexões TCP realizadas com sucesso do fluxo e o seu tempo de permanência na fila. O número médio de conexões TCP realizadas com sucesso da fila é calculado a cada intervalo de tempo, através da seguinte equação:

$$R_G = \frac{\sum_{i=1}^{m} Tc_i}{m}, \tag{4.5}$$

onde m é a quantidade de fluxos presentes na fila e Tc_i é o volume total de conexões TCP realizadas com sucesso da fila.

A Figura 4.9 mostra o fluxograma para o procedimento da *Graylist*. Primeiramente, é questionado se a solicitação de conexão TCP do fluxo f_i que chegou a rede, pertence a categoria de não classificado ou de algum fluxo presente na *Graylist*. Na condição de não classificado, o fluxo é adicionado a fila e a solução guarda as seguintes informações:

tempo em que foi enfileirado t_{f_i}, contadores para o número de conexões TCP realizadas com sucesso R_{f_i} e não realizadas S_{f_i}, volume total de pacotes (taxa) trafegados pelo fluxo T_{f_i} e por fim, o número de vezes que o fluxo é retirado da fila N_{f_i}. Na situação de ser uma solicitação de conexão TCP de um fluxo já presente na fila, ele tem os seus respectivos contadores atualizados conforme o seu comportamento na rede ao longo do tempo. A taxa de pacotes e a reputação da fila são calculadas de acordo com as Equações (4.4) e (4.5), respectivamente. Se a taxa do fluxo estiver acima da taxa de pacotes da fila, ou seja, o fluxo está gerando um enorme volume de pacotes a curto prazo acima do limiar estabelecido, ele é retirado da fila perdendo a classificação que lhe foi atribuída inicialmente e pode ser adicionado a *Blacklist*, caso ultrapasse o limiar estabelecido para o número de retiradas permitidas. Ao ser retirado da fila, o fluxo tem o seu número de retiradas N_{f_i} atualizado e comparado com o limiar δ, que é escolhido de forma a preservar a rede para a defesa de fluxos com comportamentos considerados suspeitos. Este valor, deve ser definido de maneira a não causar a incidência de falsos positivos a rede e que também não possa comprometer a solução, de forma que a própria função possa ser utilizada como alvo de um ataque. Se o número de vezes em que o fluxo foi retirado da fila estiver acima do limiar δ, significa que ele terá o seu bloqueio determinado pela solução, no qual é adicionado a *Blacklist*. Do contrário, ele apenas perderá a classificação que lhe foi atribuída inicialmente, devendo realizar todo o procedimento de reingresso a rede, para obter o primeiro nível de confiabilidade. Contudo, se ele estiver abaixo da taxa de pacotes e com um volume de conexões TCP bem-sucedidas acima da reputação da fila R_G e ter o maior tempo de permanência na fila, ele é promovido a *Whitelist*, no qual possui o seu nível de confiabilidade atualizado. Do contrário, permanecerá na fila até que atinja o grau de reputação necessário para a sua progressão.

Procedimento Whitelist

O procedimento da *Whitelist*, tem por objetivo atender e gerenciar os fluxos que possuem o maior nível de confiabilidade da rede. Os fluxos pertencentes a esta fila tem o seu comportamento analisado, através da taxa de pacotes da fila.

A taxa de pacotes T_W da *Whitelist*, tem como finalidade constatar quando um fluxo utiliza de forma demasiada os recursos disponíveis da fila, causando a perda de pacotes dos demais fluxos presentes na fila. Deste modo, a taxa é calculada a cada intervalo de tempo de acordo com a seguinte equação:

$$T_W = \frac{\sum_{i=1}^{m} Tp_i}{m}, \tag{4.6}$$

onde m é a quantidade de fluxos na fila e Tp_i é o volume total de pacotes da fila.

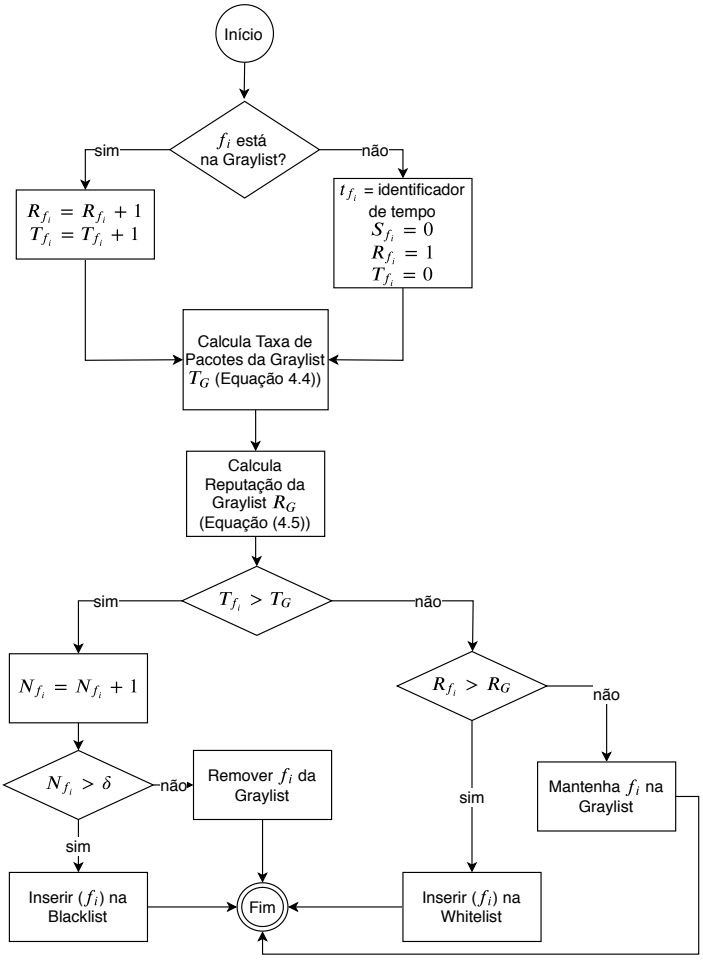


Figura 4.9: Procedimento da *Graylist*

A Figura 4.10, mostra o fluxograma para o procedimento da *Whitelist*. Inicialmente é perguntado se a solicitação de conexão TCP do fluxo f_i, pertence a fila em questão ou não. Na condição de não pertencer a fila, ou seja, foi migrado para a fila, porém não realizou nenhuma nova solicitação de conexão TCP, a solução armazena as seguintes informações do fluxo: tempo em que foi enfileirado t_{f_i}, contadores para o número de conexões TCP realizadas com sucesso R_{f_i} e não realizadas S_{f_i}, volume total de pacotes trafegados pelo fluxo T_{f_i} e por fim, o número de vezes que o fluxo é retirado da fila N_{f_i}. Na circunstância de ser um fluxo já pertencente a fila, ele tem seus respectivos contadores atualizados a cada nova conexão TCP iniciada. O comportamento da fila é observado através da taxa de pacotes T_W, que é calculada de acordo com a Equação (4.6). Se a taxa de pacotes do fluxo estiver acima da taxa de pacotes da fila, ele é retirado da *Whitelist*, ou seja, retorna à categoria de um fluxo não classificado e pode ser bloqueado temporariamente, caso ultrapasse o limiar δ, que possui as mesmas propriedades do procedimento da *Graylist*.

Do contrário, permanecerá na fila com a prioridade que lhe foi concedida.

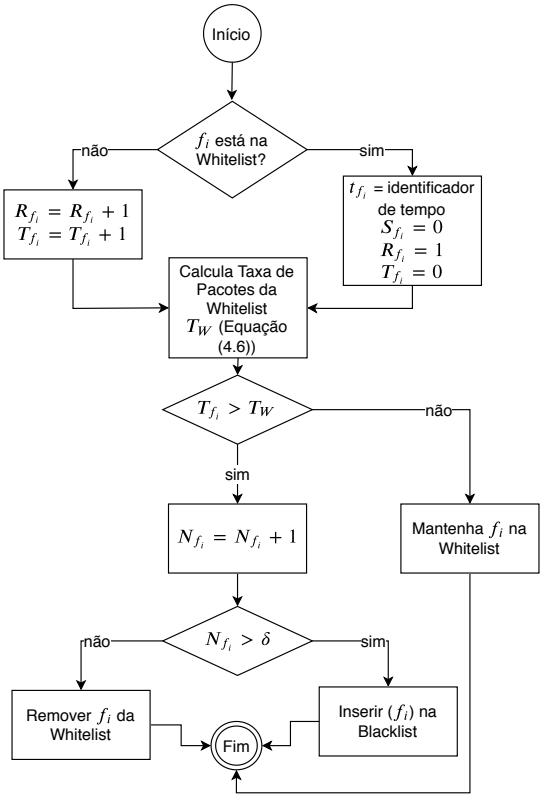


Figura 4.10: Procedimento da *Whitelist*

Procedimento Blacklist

O procedimento da *Blacklist*, tem por intuito gerenciar os fluxos não confiáveis. Os fluxos pertencentes a esta fila tem o seu comportamento analisado, através do tempo limite de permanência.

O tempo limite de permanência da fila μ_{f_i}, indica o período máximo em que o fluxo permanecerá bloqueado pela solução. O tempo de permanência do fluxo é calculado com base no seu tempo de enfileiramento. O cálculo é realizado através da seguinte equação:

$$\mu_{f_i} = t_a - t_{f_i}, \tag{4.7}$$

onde t_a representa o tempo atual da rede (*timestamp*) e t_{f_i} o tempo em que o fluxo f_i foi adicionado a fila.

A Figura 4.11 mostra o fluxograma para o procedimento da *Blacklist*. Primeiramente, quando um fluxo f_i é encaminhado para a *Blacklist* é verificado se é a sua primeira ocorrência ou não para a fila. Para a primeira ocorrência, ele tem o seu tempo de enfileiramento registrado pela solução. Do contrário, o seu tempo de permanência é calculado, por meio

da Equação (4.7) e comparado com o limiar θ, que é definido de maneira a preservar os recursos da rede, ou seja, garantir o atendimento aos demais fluxos que tiveram um bom comportamento ao longo de sua trajetória. O limiar θ deve ser adaptado às necessidades específicas e ao histórico da rede, sob consideração para a máxima eficácia. Se o tempo de permanência do fluxo μ_{f_i}, for maior que o limiar estabelecido θ. Ele é retirado da fila, no qual a solução redefine todos os dados relacionados a marcação do campo de cabeçalho ToS, o número de retiradas das filas de prioridades, o volume de conexões TCP realizadas pelo fluxo e o volume de pacotes trafegados para vazio e alerta o controlador sobre a alteração. Tal função, permitirá com que o fluxo possa ser atendido novamente pela solução. Do contrário, o fluxo permanecerá na fila até atingir o limiar estabelecido.

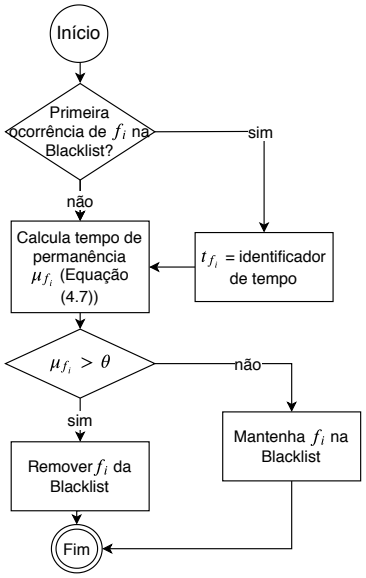


Figura 4.11: Procedimento da *Blacklist*

4.2 Considerações Finais

Este capítulo apresentou a solução de segurança em redes SDN desenvolvida neste trabalho, chamada DoSSEC. A solução tem como objetivo reduzir os impactos causados pelo ataque SYN *Flood*, ela permite a detecção e mitigação diretamente nos dispositivos que compõem o plano de dados. O próximo capítulo apresenta uma análise experimental da solução proposta.

Capítulo 5

Experimentos

Este capítulo apresenta informações relacionadas à avaliação de desempenho da solução proposta no Capítulo 4. Analisamos o seu desempenho quando comparada com o Avant-Guard [15], uma solução que atua no plano de dados e que procura reduzir os efeitos causados pelo ataque SYN *Flood*. As seções a seguir apresentam o ambiente experimental, a metodologia, as métricas consideradas em nossa análise e os resultados observados nos experimentos.

5.1 Ambiente de teste e configuração

Os experimentos foram realizados em um computador com sistema operacional Ubuntu 16.04 LTS, com processador $i5$ 7400 3.0 GHz CPU e 4Gb de memória RAM. Utilizamos o Mininet [79] como emulador de rede, ele utiliza a virtualização baseada em processos para executar diversos *hosts* e *switches* em um único *kernel* do sistema operacional. O emulador permite a configuração de conexões de rede entre dispositivos e a configuração dos parâmetros de *links* necessários.

A topologia de rede consiste em 4 *switches*, 9 *hosts* e 1 controlador, como mostra a Figura 5.1. Tal topologia foi definida por permitir a distribuição do tráfego de maneira a avaliar a solução proposta. A largura de banda de cada enlace é de 100 Mbps. Nas experiências, supõe-se que o invasor controle um dos *hosts* da rede. O invasor não tem controle sobre o *switch*, controlador ou canal de controle. Como pode ser visto na Figura 5.1, o *host* da vítima ($h1$) está localizado no *switch* $S2$ e o atacante inicia o ataque a partir de um *host* pertencente à rede do *switch* $S4$. A vítima é um servidor *web*, responsável por receber solicitações dos clientes.

Analisamos o desempenho da solução proposta, comparando uma solução presente na literatura que também atua no plano de dados e procura reduzir os efeitos causados pelo ataque SYN *Flood*: Avant-Guard [15], que adiciona inteligência aos dispositivos que

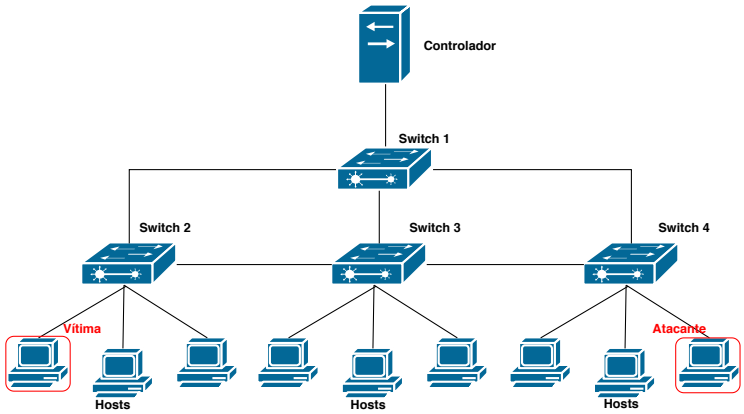


Figura 5.1: Topologia de rede

compõem o plano de dados e com uma solução denominada SDN Padrão, que refere-se a uma implementação padrão que utiliza o protocolo OpenFlow [2] sem nenhum método de defesa contra o ataque SYN *Flood*. Os testes contemplam dois cenários, antecipamos que no segundo cenário é o que diferencia a nossa solução das demais soluções apresentadas no estado da arte.

- **Cenário 1:** o atacante envia um grande número de pacotes SYN com IP de origem falsificado em direção ao servidor, com o objetivo de exaurir o canal de comunicação entre o plano de dados e o controlador.
- **Cenário 2:** o atacante realiza uma conexão TCP de maneira completa e a partir dela inicia o ataque ao servidor, por meio de um grande volume de pacotes SYN, com objetivo de esgotar as entradas da tabela de fluxo do *switch*. Apesar do Avant-Guard [15] não tratar este tipo de cenário, nós pensamos em um adversário que conhece o método de mitigação empregado pela solução.

A solução proposta é implementada nos *switches* programáveis: $S2$, $S3$ e $S4$. Devido a dificuldade de ser obter traços reais de ataque SYN *Flood* como os avaliados neste trabalho, optou-se por simular o tráfego de rede. Dois tipos de tráfegos são gerados: tráfego legítimo e tráfego espúrio. O tráfego legítimo é criado usando a ferramenta Scapy [93], que permite criar pacotes TCP para a geração de tráfegos de rede de maneira simples e rápida. Todos os *hosts* clientes geram tráfego em direção ao servidor. O servidor apenas recebe e processa o tráfego gerado pelos *hosts* clientes. O tráfego espúrio é gerado usando a ferramenta Hping [94], que simula o ataque volumétrico SYN *Flood*. O tráfego legítimo é usado para calcular valores do método estatístico em condições normais e, assim, definir o limiar de detecção de acordo com a Equação (4.2). Depois que o limiar é definido, o tráfego espúrio é injetado para analisar o comportamento da solução proposta.

Os seguintes parâmetros são usados nos experimentos por apresentarem resultados satisfatórios em simulações preliminares: a janela de observação é configurada para 60 ($|w_i| = 60$) pacotes, o campo a ser observado é o IP de origem, coeficiente de suavização é definido para 0,5 ($\alpha = 0{,}5$) e a técnica estatística utilizada é o qui-quadrado (x^2). Definimos a capacidade das filas para 5.000 pacotes. O tempo total da simulação de cada experimento é de 60s. O tráfego espúrio começa no tempo 20s e tem duração de 20 segundos.

As simulações foram realizadas sob diferentes cargas de tráfego para avaliar o desempenho da solução proposta, com base na capacidade total da rede. Para esse fim, a carga geral da rede foi mantida constante nos experimentos, enquanto a carga do tráfego espúrio varia de 20%, 40%, 60% e 80% em relação ao tráfego legítimo. Essa configuração fornece 4 cargas de trabalhos o que corresponde à diferentes taxas de ataque.

5.2 Métricas

Avaliamos os resultados experimentais usando as seguintes métricas de desempenho:

- **Tempo de resposta HTTP:** é o tempo gasto por um *host* legítimo para obter a resposta HTTP de volta do servidor. A equação é definida como:

$$T_{total} = T_{init} - T_{resp}, \tag{5.1}$$

  onde T_{init} é o tempo em que o *host* legítimo envia o pedido de conexão ao servidor e T_{resp} é o tempo de resposta do servidor para a solicitação do *host*.

- **Número de entradas instaladas no *switch*:** indica o número de entradas de encaminhamento que foram instaladas nos *switches* pelo controlador.

- **Número de conexões ativas:** percentual de conexões atendidas durante o ataque a rede. A equação é definida como:

$$N_{con} = \frac{T_{Conexões} - C_{Atendidas}}{T_{Conexões}} \cdot 100, \tag{5.2}$$

  onde $T_{Conexões}$ o volume de solicitações de conexão e $C_{Atendidas}$ o volume do conexões atendidas.

- **Precisão da solução:** obtida através de uma matriz de confusão para avaliar a eficiência da solução proposta. A Tabela 5.1 exemplifica o modelo de matriz utilizado. A matriz é composta por valores previstos (resultados obtidos) e valores reais (resultados esperados) do tráfego espúrio e tráfego normal, com base na taxa

de falso positivo (FP), falso negativo (FN), verdadeiro positivo (VP) e verdadeiro negativo (VN).

Tabela 5.1: Matriz de confusão

Previsto	**Real**	
	Ataque	**Normal**
Ataque	VP	FP
Normal	FN	VN

A taxa de falso negativo é dada pelo número de pacotes classificados, como normais (negativos) que são confirmados como pacotes de ataques (positivos). Enquanto, a taxa de falso positivo é dada pelo número de pacotes classificados, como pacotes de ataque (positivos), onde são confirmados como pacotes normais (negativo). A taxa de verdadeiro positivo, representa o número de pacotes classificados corretamente como, ataques (positivo). A taxa de verdadeiro negativo, indica o número de pacotes classificados corretamente como normal (negativo).

Por meio das taxas de falso positivo, falso negativo, verdadeiro positivo e verdadeiro negativo, calculamos a precisão, *recall* e o $F1$ score do mecanismo de detecção proposto. A precisão representa a proporção de identificações positivas que estavam corretas, medindo a qualidade dos resultados. A equação é definida como:

$$Precisão = \frac{VP}{(VP + FP)}, \tag{5.3}$$

onde VP é a taxa de verdadeiro positivo e FP é a taxa de falso positivo.

O *recall* é a proporção de positivos que foram identificados corretamente, medindo a completude do método. A equação é definida como:

$$Recall = \frac{VP}{(VP + FN)}, \tag{5.4}$$

onde VP é a taxa de verdadeiro positivo e FN é a taxa de falso negativo.

O $F1$ *score* é definido como a média harmônica da precisão e do *recall*, com a finalidade de trazer o equilíbrio entre ambos, para indicar o desempenho geral do método. Se o F1 *score* for próximo de 1, a precisão do método de detecção é boa. A equação é definida como:

$$F1\ score = \frac{(2 \cdot VP)}{(2 \cdot VP + FN + FP)}, \tag{5.5}$$

onde VP é a taxa de verdadeiro positivo, FN é a taxa de falso negativo e FP é a taxa de falso positivo.

- **Consumo de recursos:** consumo médio de recursos da CPU pela solução proposta.

5.3 Resultados

Nesta seção, discutimos e analisamos os resultados obtidos para a solução proposta. Primeiramente, investigamos o tempo de resposta HTTP para estabelecer uma conexão com o servidor TCP em um cenário sem nenhum atacante, o resultado pode ser visto na Figura 5.2.

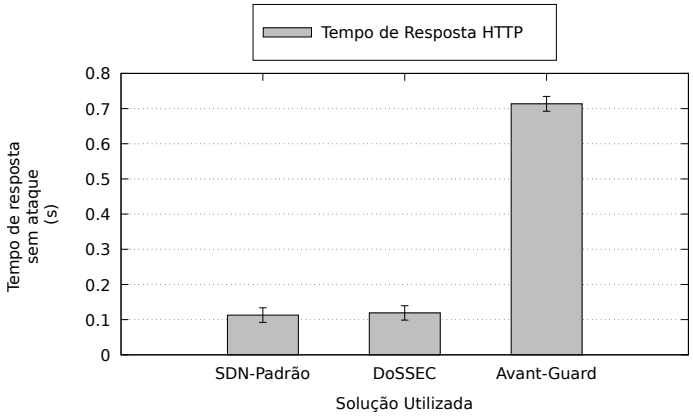


Figura 5.2: Tempo de resposta sem ataque

Podemos observar que o Avant-Guard impõe um tempo considerável em comparação as demais soluções, SDN Padrão e o DoSSEC. Tal fato ocorre porque a solução causa um longo atraso para estabelecer uma nova conexão para solicitações legítimas dos fluxos, independente do cenário em questão, seja ele sob ataque ou em condições normais. Pois, há uma intermediação entre os *hosts* (origem e destino) através do módulo de migração de conexão. No qual, primeiro é realizado a troca de informações entre a origem e o *switch*, após a conclusão desta conexão, o controlador autoriza o processo de migração de conexão, em que o *switch* inicia a conexão com o destino, para que os dados sejam finalmente transferidos. Ao invés disso, o DoSSEC não quebra semântica de ponta a ponta do TCP durante um ambiente em condições normais, o que reduz em média 82%, o tempo de resposta quando comparado ao Avant-Guard para um cenário sem a presença de nenhum atacante a rede.

5.3.1 Cenário 1

Para o cenário 1, o atacante envia um grande número de pacotes SYN com IPs de origem falsificados e aleatórios em direção ao servidor, com o objetivo de exaurir o canal de comunicação entre o plano de dados e o controlador.

Analisamos o tempo de resposta HTTP, para o cenário em que a rede está sob ataque, o resultado pode ser visto na Figura 5.3. A medida que aumentamos o volume do ataque, o tempo de resposta HTTP aumenta significativamente para a solução SDN Padrão, devido à ausência de qualquer mecanismo defensivo, o controlador recebe e processa cada solicitação recebida, independentemente do tipo de fluxo inserido, seja ele legítimo ou espúrio. Como resultado, uma alta taxa de ataque, causa um tempo de resposta mais longo. O Avant-Guard por sua vez, faz uso do módulo de migração de conexão, que transfere para o destino apenas as conexões que foram validadas e autorizadas pelo controlador, porém a intermediação entre origem e destino e o atendimento aos fluxos pela mesma fila acaba interferindo no tempo de resposta. Em contrapartida, o DoSSEC ao detectar a um possível ataque em andamento, realiza a redefinição das conexões TCP dos novos fluxos sejam eles provenientes do ataque ou não. Este processo acaba causando um atraso aos novos fluxos legítimos, porém garante com que os fluxos validados sejam priorizados e tenha um tempo de resposta menor reduzindo o congestionamento de fluxos no *switch*, proporcionando uma redução em média de 30% em comparação com o Avant-Guard. Em comparação com o tempo de resposta sem ataque, o Avant-Guard aumentou 16% em virtude de atender todos os fluxos pela mesma fila e o DoSSEC 9% em razão da redefinição das novas solicitações de conexão para os fluxos considerados legítimos.

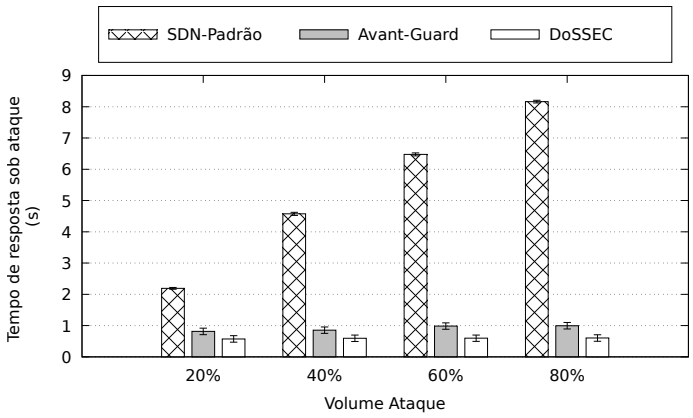


Figura 5.3: Tempo de resposta durante o ataque

A Figura 5.4 mostra o número de entradas de encaminhamento instaladas no *switch* pelo controlador para cada solicitação de conexão TCP durante o cenário proposto. Conforme o volume do tráfego espúrio aumenta o número de entradas de encaminhamento

seguem a mesma direção. Nota-se que na solução SDN Padrão, como não há nenhuma política de defesa, o controlador instala as entradas de encaminhamento para cada nova solicitação de conexão TCP recebida e, subsequentemente, o número de entradas aumenta causando um gargalo na comunicação entre o controlador e o *switch*. Por outro lado, o Avant-Guard e o DoSSEC preservam o canal de comunicação, onde o número de entradas de encaminhamento instaladas pelo controlador reflete apenas para as solicitações de conexão dos fluxos legítimos, de acordo com a sua taxa. O Avant-Guard efetua o bloqueio das solicitações provenientes do ataque e o DoSSEC realiza a redefinição delas, impedindo que o controlador instale regras desnecessárias. O DoSSEC apesar ter um pequeno número de entradas acima do Avant-Guard, tal incremento não afeita a eficácia da solução proposta.

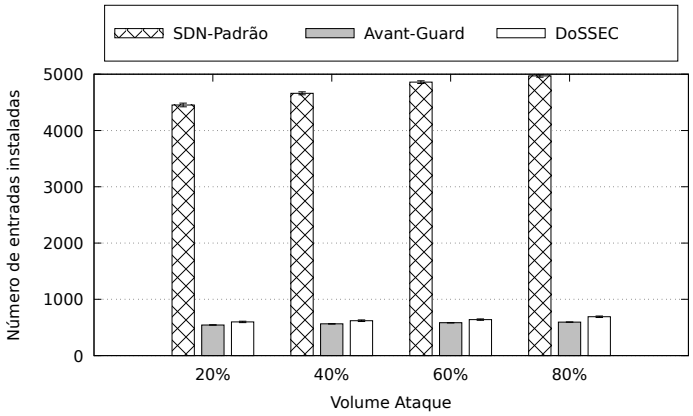
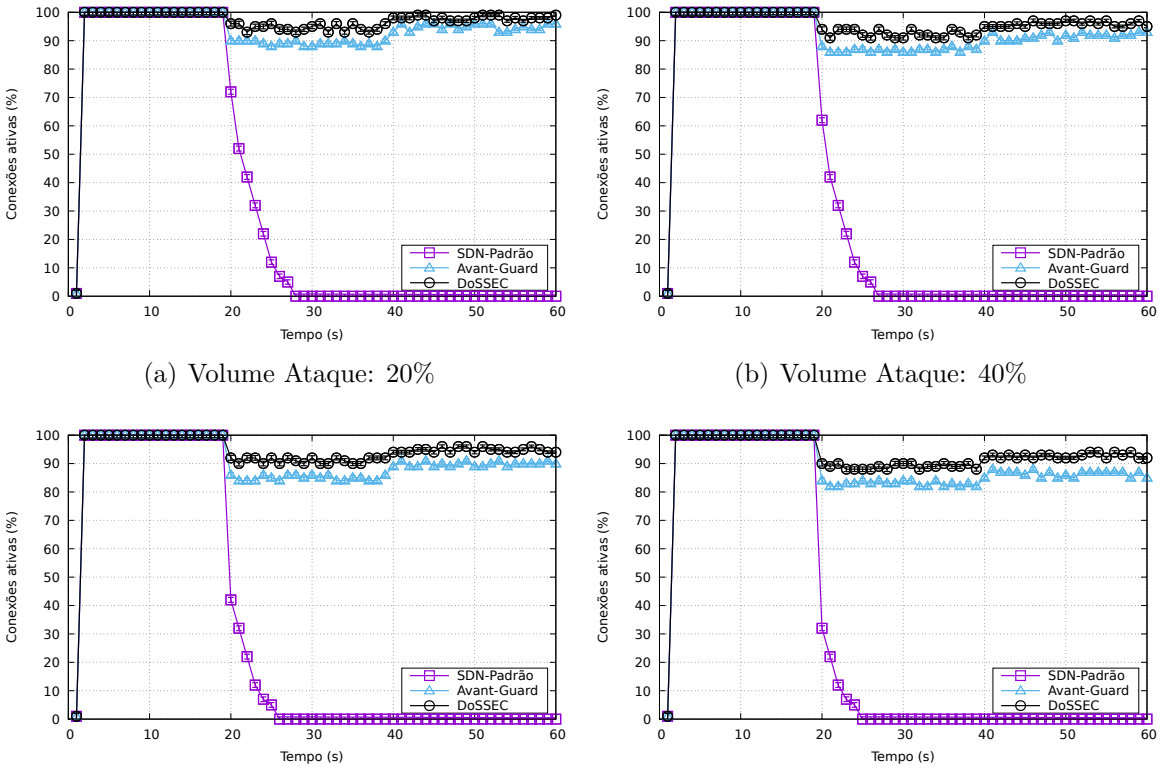


Figura 5.4: Número de entradas de encaminhamento instaladas

A Figura 5.5 mostra a taxa de conexões ativas para os diferentes volumes de tráfego espúrio para o cenário em questão. O percentual de conexões ativas diminui significativamente à medida que o volume do ataque aumenta. Por exemplo, para o cenário de 80% do tráfego espúrio, na solução SDN Padrão a taxa diminui rapidamente de 100% para 30% nos primeiros instantes do ataque, este efeito é em decorrência do grande volume de tráfego que consome todos os recursos da rede em um intervalo de tempo mais curto, causando uma baixa nas conexões ativas, até chegar ao ponto em que nenhuma conexão é atendida pela solução. O Avant-Guard e o DoSSEC têm melhor desempenho, pois possuem mecanismos para mitigar as solicitações oriundas do ataque. No entanto, o Avant-Guard possui um desempenho mais baixo com relação ao DoSSEC, já que todos os pacotes são tratados pela mesma fila sem nenhum tipo de distinção de prioridade, o que permite com que fluxos espúrios possam competir com fluxos legítimos para chegar ao seu destino, causando um congestionamento, mesmo que haja uma validação entre origem e destino (migração de conexão). O DoSSEC por sua vez, obteve um aumento cerca de 7,2% no número de pacotes legítimos entregues ao destino em comparação com o

Avant-Guard. Isto nos mostra que o gerenciamento de filas de prioridade nos dispositivos de encaminhamento é capaz de aumentar a qualidade do serviço, conferindo vantagem aos fluxos já validados, mesmo a rede estando sob forte ataque, permitindo assim que mais pacotes de fluxos legítimos sejam entregues ao destino.



(a) Volume Ataque: 20%

(b) Volume Ataque: 40%

(c) Volume Ataque: 60%

(d) Volume Ataque: 80%

Figura 5.5: Percentual de conexões ativas no cenário 1 com injeção de tráfego espúrio entre 20 e 40s

A Figura 5.6 mostra o resultado com relação a precisão de detecção do tráfego espúrio para as soluções analisadas, por meio do F1 *score*. A solução SDN Padrão não possui nenhum mecanismo de detecção, logo a sua taxa de detecção é nula. O Avant-Guard possui um bom desempenho, chegando a taxa de 94% para o cenário de 80% do tráfego espúrio. Este resultado está relacionado ao módulo de migração de conexão e aos gatilhos de atuação que retorna as informações com base nas decisões do controlador para o bloqueio do tráfego espúrio. O DoSSEC por sua vez, conforme o tráfego espúrio se torna mais volumoso, isto é, assume uma proporção maior quando comparado ao tráfego legítimo, a precisão da detecção alcança taxas cada vez mais altas (excedendo mais de 93%). Essa precisão é profundamente influenciada pela variação do tráfego, que se torna mais concentrada e, consequentemente, mais fácil de detectar. Essa facilidade é intrínseca à detecção de ataques baseados em anomalia e é exacerbada quando se considera proporções

menores do tráfego, o que resulta em estimativas menos precisas das técnicas de análise estatística.

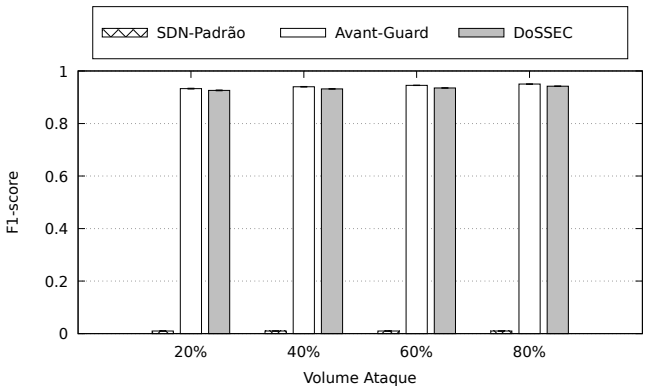
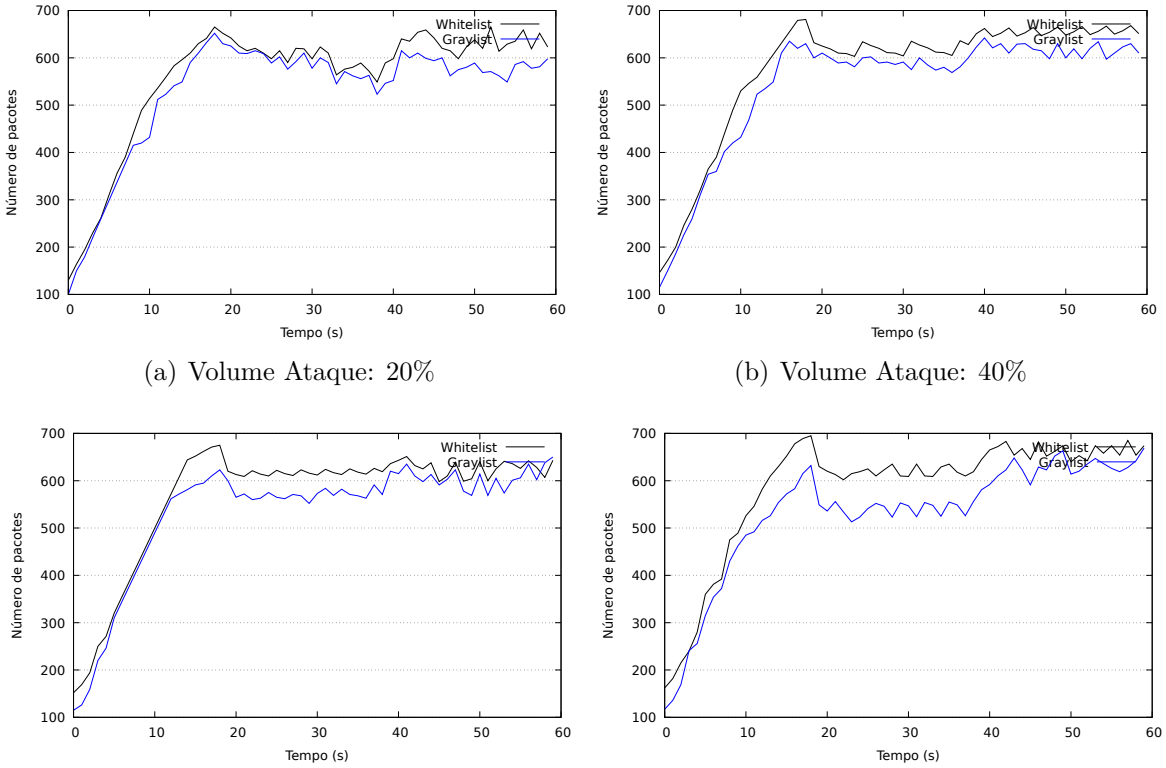


Figura 5.6: F1 score para o cenário 1

A Figura 5.13 mostra o comportamento das filas *Graylist* e *Whitelist* para as diferentes cargas de tráfego espúrio, para o cenário em questão. Nota-se que há uma pequena diminuição no número de pacotes presentes na fila, tal queda reflete-se na intensidade do ataque a rede, diminuindo o número de pacotes legítimos trafegados na rede. Por exemplo, para o cenário de 80% há uma redução de 6% no volume de pacotes na fila, em decorrência da grandeza do ataque. Mesmo com esta pequena queda a solução garante com que os fluxos presentes nas filas tenham seus pacotes priorizados e atendidos.

A Figura 5.8 mostra o consumo de recursos da CPU das soluções para o cenário em questão. A solução SDN Padrão, não possui nenhum mecanismo para impedir o invasor e, como resultado, os recursos da rede são exauridos causando a sobrecarga da solução à medida em que o ataque avança, pois, o controlador e o *switch* precisam processar cada vez mais pacotes oriundos do ataque. Por outro lado, o Avant-Guard e o DoSSEC impedem que os recursos da rede sejam exauridos. O módulo de migração de conexão do Avant-Guard, apesar de eficiente, introduz uma notável sobrecarga ao dispositivo para a manipulação das conexões, em média o seu consumo chega a ser de 44%. Isto porque, a solução utiliza um algoritmo de SYN *cookie* [95] para a codificação dos pacotes e geração de processamento de *hash*, com a finalidade de validar a origem e assim migrar a solicitação de conexão até o seu destino. Enquanto o DoSSEC, apesar de utilizar um módulo adicional para o processamento de cálculos estatísticos e das filas, tem o consumo em média de 23,25%, cerca de 20,75% a menos quando comparado com o Avant-Guard. O motivo deste consumo é graças a adoção de um algoritmo leve para o acompanhamento das conexões e também ao uso da conexão Thrift [92], por meio de chamadas RPC para a comunicação com o módulo adicionado ao *switch*, que provê uma conexão eficiente entre linguagens de programação, por exemplo, *Python* e P4, usando uma ponte de *software*



(a) Volume Ataque: 20%

(b) Volume Ataque: 40%

(c) Volume Ataque: 60%

(d) Volume Ataque: 80%

Figura 5.7: Filas de prioridade para o cenário 1

transparente e de alto desempenho. Tais fatores contribuem para que a solução consuma poucos recursos computacionais para a manipulação dos fluxos.

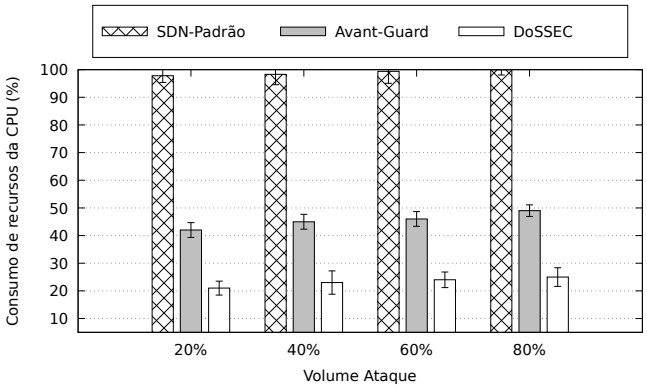


Figura 5.8: Consumo de recursos da CPU para o cenário 1

5.3.2 Cenário 2

Para o cenário 2, o atacante completa uma conexão TCP e a partir dela inicia o ataque ao servidor, por meio de um grande volume de aberturas de conexão (SYN), com objetivo de

esgotar as entradas da tabela de fluxo do *switch*, causando saturação do plano de dados.

A Figura 5.9 mostra o resultado para o tempo de resposta HTTP. Podemos observar que as soluções SDN Padrão e Avant-Guard possuem um aumento abrupto no tempo de resposta, ao passo que o volume do tráfego gerado pelo atacante cresce, causando uma degradação mais rápida aos recursos da rede. Para a primeira solução esse comportamento é absolutamente normal, pois como já havíamos mencionado não há nenhum mecanismo de defesa implementado. Por outro lado, vimos que o Avant-Guard não é efetivamente hábil em lidar com este tipo de ataque, no qual o atacante completa a primeira conexão TCP (*three-way handshake*) e logo em seguida inunda a rede com um grande volume de pacotes SYN, a partir da primeira conexão validada para este fluxo. O aumento no tempo de resposta é em decorrência do número de entradas de encaminhamento estabelecidas na tabela de fluxo do *switch*, causando a saturação aos dispositivos que compõem o plano de dados, causando a perda de pacotes de fluxos legítimos em decorrência do volume do ataque. Portanto, pudemos perceber um atraso considerável no tempo de resposta para as solicitações de conexão dos fluxos. O DoSSEC demonstrou ser eficaz, pois o acompanhamento das conexões TCP do fluxo pela solução, mesmo após ele ser adicionado as filas de prioridades, permite controlar o seu comportamento ao longo do seu tempo de vida na rede, monitorando comportamentos suspeitos. O tempo de resposta para o DoSSEC é em média 76% mais rápido, quando comparado com o tempo de resposta do Avant-Guard para processar as solicitações de abertura de conexão TCP.

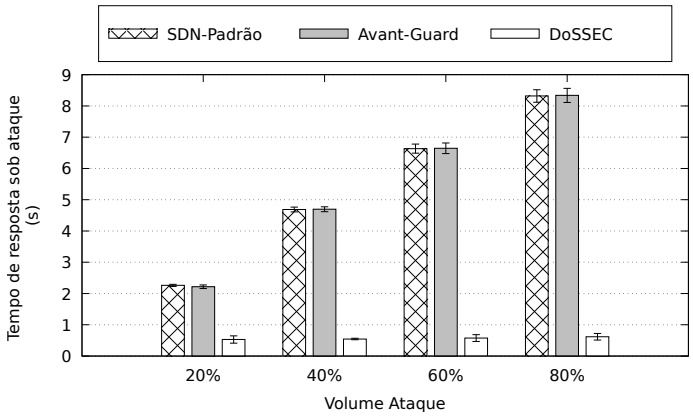


Figura 5.9: Tempo de resposta sob ataque

A Figura 5.10 mostra o número de entradas de encaminhamento instaladas no *switch* pelo controlador, para o cenário em questão. Nota-se que à medida em que o atacante inicia o ataque o número de entradas nas soluções SDN Padrão e Avant-Guard, aumentam de forma significativa. Isto porque, a entrada instalada pelo controlador para o endereço do atacante é atualizada à medida em que o *switch* remove a regra para o fluxo questão, quando ele atinge o tempo limite ocioso na tabela de encaminhamento, porém devido

ao volume do ataque este processo é repetido várias vezes e acaba causando um volume de entradas e o estouro na tabela de encaminhamento do *switch*. Diferentemente, o DoSSEC possui um pequeno aumento nas entradas instaladas, já que o atacante completa a primeira conexão TCP e é adicionado a fila, gerando inicialmente a entrada no *switch*. Porém, em virtude da taxa de uso da fila (monitoramento das filas), o fluxo responsável por gerar esse aumento imediatamente ultrapassa o limiar estabelecido e é retirado da fila, perdendo a prioridade que lhe foi atribuída inicialmente e posteriormente bloqueado. Desta maneira, o número de entradas instaladas é controlada preservando a tabela de encaminhamento do *switch*, no qual as entradas correspondentes refletem apenas para as solicitações de conexão TCP dos fluxos legítimos.

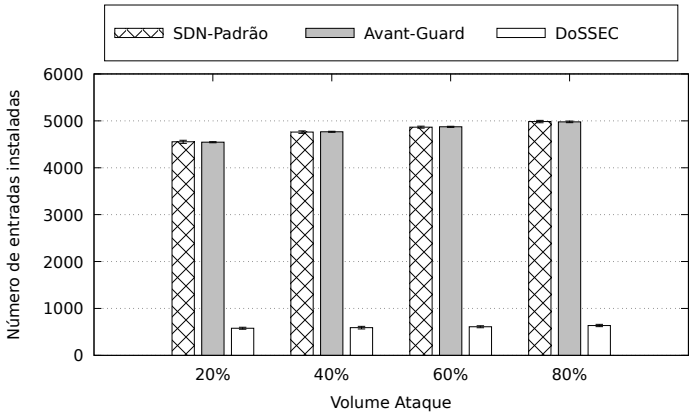
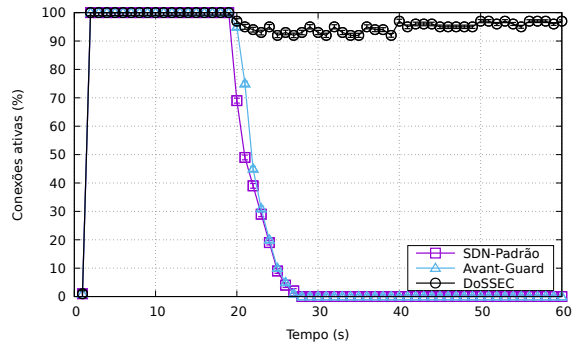


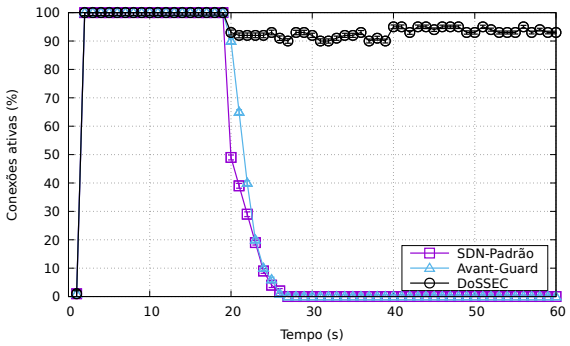
Figura 5.10: Número de entradas de encaminhamento instaladas

A Figura 5.11 mostra a taxa de conexões ativas para os diferentes volumes de tráfego espúrio. A taxa de conexão diminui significativamente à medida que o volume do ataque aumenta para as soluções SDN Padrão e Avant-Guard. O Avant-Guard por não dispor de nenhuma técnica que permite reavaliar o fluxo, tem o número de conexões ativas prejudicado, sofrendo uma redução ao longo do período em que o atacante se mantém presente na rede, o que causa o esgotamento das entradas da tabela de fluxo do *switch* para o encaminhamento das conexões, e como resultado temos uma diminuição do número de conexões dos fluxos legítimos. Há uma redução de 20% nos primeiros instantes do ataque para o Avant-Guard, chegando a zero rapidamente. Pois as solicitações de conexão provenientes do atacante consomem todos os recursos da rede. O DoSSEC por sua vez, possui o procedimento de acompanhamento de conexão para todos os fluxos, isto significa que mesmo após serem validados, eles possuem o seu comportamento analisado pela solução, em média 92% dos pacotes são entregues ao seu destino. Ao identificar o fluxo responsável pelo ataque, ele é retirado da fila de prioridade. No qual, terá que realizar uma nova tentativa de conexão TCP completa, para que lhe seja atribuída algum tipo de

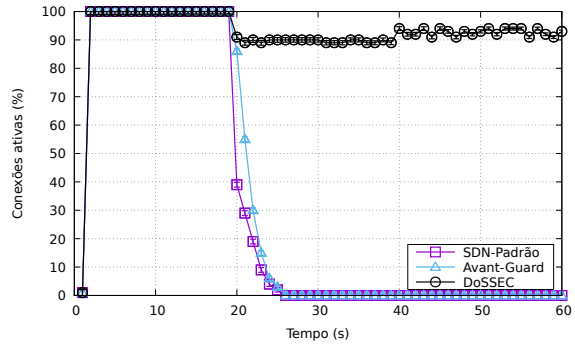
prioridade, aliviando de forma imediata os recursos da rede e garantindo a qualidade de serviço para os demais fluxos.



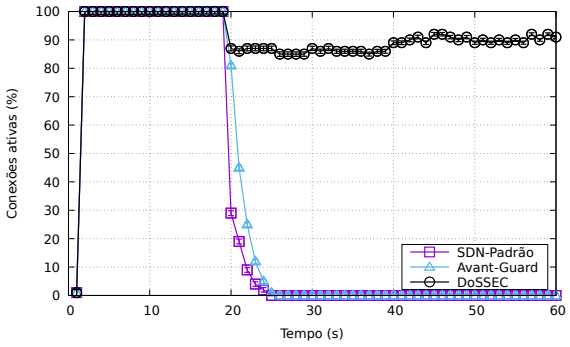
(a) Volume Ataque: 20%



(b) Volume Ataque: 40%



(c) Volume Ataque: 60%



(d) Volume Ataque: 80%

Figura 5.11: Percentual de conexões ativas no cenário 2 com injeção de tráfego espúrio entre 20 e 40s

A Figura 5.12 mostra o resultado no que diz respeito à precisão das soluções analisadas para a detecção do tráfego espúrio, mediante o F1 *score*. O Avant-Guard para o cenário em questão mostra que o ataque passa despercebido pela solução. Isto porque, o atacante completa a primeira conexão, ou seja, consegue ter a sua conexão migrada para o destino, o que lhe permite contato direto com o destino (servidor). Com o caminho livre ele inunda as entradas da tabela do *switch* com um volume de pacotes de abertura de conexão, causando um esgotamento da tabela e impedindo que a vítima seja acessada. O DoSSEC por sua vez, mostra que está preparado para este tipo de situação. Conforme a taxa de tráfego espúrio aumenta, o mecanismo torna-se mais eficiente ao detectar e remover o fluxo, pois o volume de conexões alcança o limiar de forma mais acentuada, contribuindo para uma detecção mais precisa. Por exemplo, para o cenário de 80% do tráfego espúrio, o F1 *score* aumentou cerca de 4% quando comparado com o volume de 20%.

A Figura 5.13 mostra o comportamento das filas *Graylist* e *Whitelist* para as diferentes cargas de tráfego espúrio, para o cenário em questão. Podemos observar que, o aumento

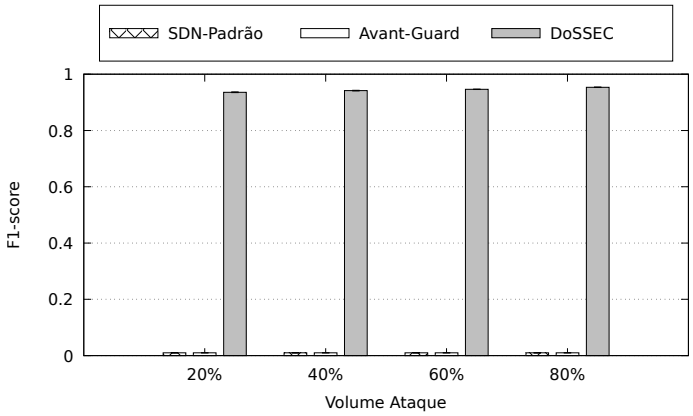
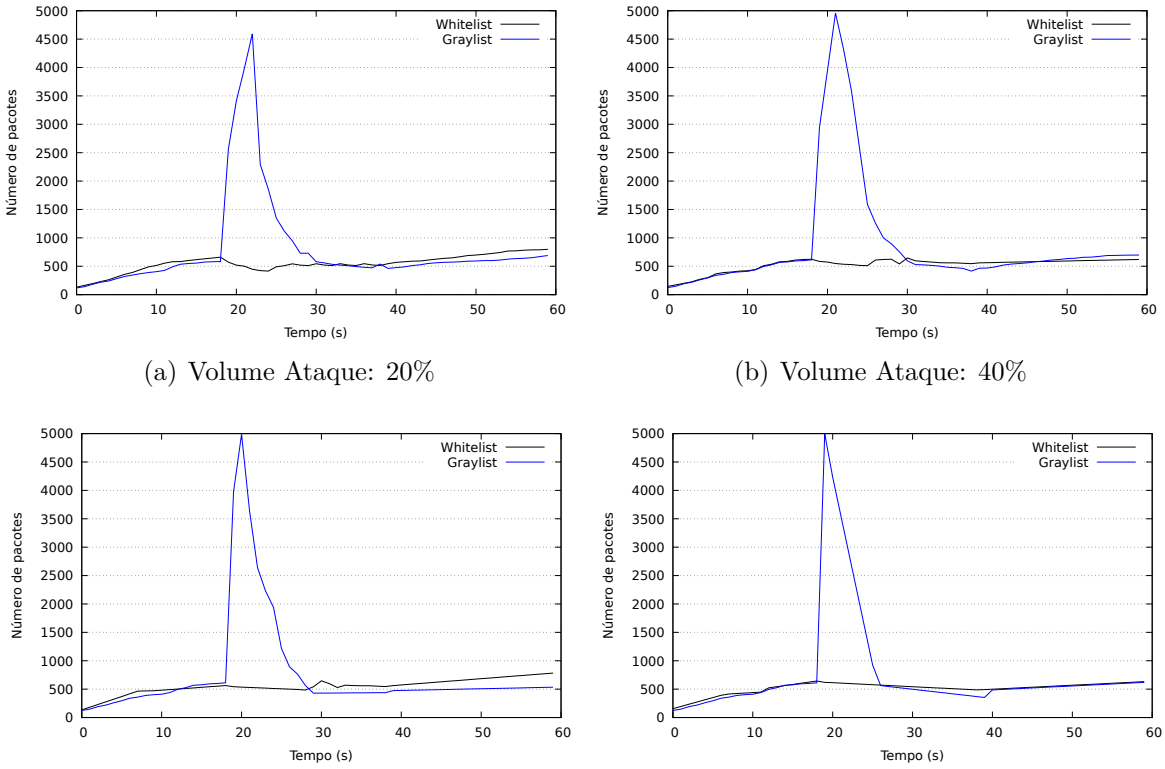


Figura 5.12: F1 score para o cenário 2

no volume de pacotes na *Graylist* é em decorrência do atacante realizar a conexão TCP de maneira completa e a partir dela iniciar o ataque ao *switch*, causando descarte de alguns pacotes legítimos na fila. Porém, a taxa de uso da fila, responsável por analisar o comportamento dos fluxos nas filas de prioridade, identifica esse volume anormal de pacotes e realiza o processo de retirada o fluxo da fila, fazendo com que o número de pacotes espúrios diminua gradativamente. Nota-se que a medida em que o volume do tráfego espúrio aumenta, a taxa de uso da fila age mais rapidamente, preservando um maior número de pacotes legítimos presentes na fila. Por exemplo, para o cenário de 80%, a retirada do fluxo espúrio ocorre de forma mais rápida, sendo quatro vezes mais rápido em comparação com o cenário de 20%, ou seja, preservando um número maior de pacotes.

Por fim, a Figura 5.14 mostra o consumo de recursos gerais disponíveis da solução para o cenário em questão. A solução SDN-Padrão não possui nenhum mecanismo para lidar com as solicitações de conexão TCP do atacante, o que faz com que ela tenha que processar a cada nova solicitação recebida. Ao aumentar a taxa do ataque, a solução terá que lidar com um maior volume de pacotes, o que aumentará a sobrecarga da solução, chegando a consumir 100% dos recursos disponíveis. Para este cenário, o atacante gera um grande número de conexões do mesmo endereço IP de origem. Neste ponto, o Avant-Guard precisa armazenar o estado e processar cada uma dessas novas conexões, como resultado, temos um aumento drástico no consumo de recurso da CPU por parte da solução. Já o DoSSEC para o cenário em questão, possui uma média de consumo de 22,4%. O motivo da redução na utilização dos recursos é através das ações de acompanhamento das filas de prioridade e a taxa de uso, que detectam o comportamento suspeito e realizam ações para aliviar os recursos, como resultado, o consumo de CPU mantém-se estável durante o ataque.



(a) Volume Ataque: 20%

(b) Volume Ataque: 40%

(c) Volume Ataque: 60%

(d) Volume Ataque: 80%

Figura 5.13: Filas de prioridade no cenário 2

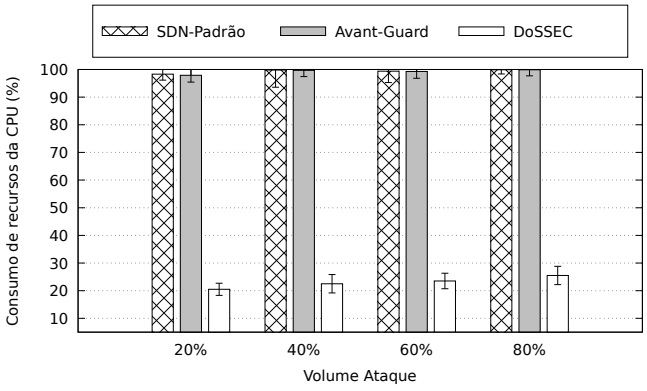


Figura 5.14: Consumo de recursos da CPU para o cenário 2

Capítulo 6

Conclusão

O paradigma de Redes Definidas por *Software*, apesar de oferecer muitas vantagens, proporciona também diversos desafios. Dentre eles, está em como reduzir os impactos causados pelos ataques de negação de serviço, em especial o SYN *Flood*. Logo, é preciso propor soluções que sejam capazes de garantir a escalabilidade e resiliência da rede, de modo a torná-la mais segura e estável.

Ao longo do projeto pudemos presenciar as principais características deste novo paradigma de redes e suas contribuições para a comunidade de segurança e como as soluções presentes na literatura procuram preservar os dispositivos que compõem a rede, diante de um ataque SYN *Flood*. A revisão do estado da arte nos permitiu identificar lacunas que ainda deixam a rede vulnerável a este tipo de ataque. De modo, a suprir estas brechas encontradas, aproveitamos o recente conceito de plano de dados programável para tornar os dispositivos que compõem o plano de dados em agentes capazes de realizar ações que possam minimizar os impactos causados pelo ataque em questão, com a mínima intervenção do controlador e assim reduzir a sobrecarga imposta ao mesmo e tornar a rede mais segura.

A solução desenvolvida por este trabalho foi chamada de DoSSEC. Esta solução permite que diferentes métodos estatísticos sejam aplicados para detectar alterações no volume do tráfego e assim detectar eventuais ataques à rede. Além de fornecer suporte ao principal protocolo de comunicação entre o plano de dados e controle, o OpenFlow, permitindo a combinação de *switches* programáveis e os dispositivos OpenFlow na rede. Reduzimos o impacto causado pelo ataque SYN *Flood*, através de um gerenciamento de filas nos dispositivos de encaminhamento, realizando a priorização de fluxos, garantindo que fluxos legítimos sejam atendidos pela solução de maneira prioritária.

A avaliação da solução proposta foi realizada por meio de simulações executadas no emulador Mininet, onde os resultados foram comparados com o Avant-Guard [15] e com uma solução sem qualquer tipo de proposta de segurança. Foram simulados dois cenários,

com diferentes taxas de tráfego espúrio. Os resultados experimentais mostraram que, o DoSSEC reduz em média 30% o tempo de resposta a requisições HTTP quando a rede está sob ataque, quando comparado ao Avant-Guard. Além disso, melhora a taxa de conexões TCPs realizadas com sucesso e diminui o congestionamento de fluxos mediante o gerenciamento dos níveis de prioridades, reduzindo os impactos causados pelo ataque SYN *Flood*.

O estudo apresentado nesta dissertação teve como objetivo principal fornecer uma solução de detecção e mitigação do ataque SYN *Flood* em redes SDN. Este objetivo foi alcançado, através de uma solução que utiliza métodos estatísticos para detectar eventuais ataques a rede e provê a priorização de fluxos por meio de um método baseado em reputação, para denotar o nível de confiança dos fluxos e assim preservar um conjunto benigno de fluxos.

6.1 Trabalhos Futuros

Por fim, como trabalhos futuros, propõe-se aprimorar as técnicas de detecção, criando um ambiente colaborativo entre os planos (dados e controle). Permitindo assim, um processo de filtragem em dois estágios: no plano de dados uma detecção mais grossa (uso do qui-quadrado) e no plano de controle uma detecção mais fina (uso da entropia). Além de realizar o compartilhamento de informações sobre o gerenciamento das filas nos dispositivos de encaminhamento para os demais controladores presentes na rede. Ampliar o leque de mitigação para ataques DDoS, que explorem o protocolo TCP e UDP, ataques de reflexão e amplificação, baseados em diferentes protocolos como DNS e NTP.

Referências

[1] Kreutz, D., F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky e S. Uhlig: *Software-defined networking: A comprehensive survey.* Proceedings of the IEEE, 103(1):14–76, janeiro 2015. x, 1, 18, 19, 20, 41

[2] Foundation, Open Network: *Openflow switch specification 1.0.* 1:1–44, dezembro 2009. x, 20, 21, 22, 23, 26, 46, 60

[3] Foundation, Open Network: *Openflow switch specification 1.1.* fevereiro 2011. x, 23, 24

[4] P. Bosshart, D. Daly, G. Gibb M. Izzard N. McKeown J. Rexford C. Schlesinger D. Talayco A. Vahdat G. Varghese e D. Walker: *P4: Programming protocol-independent packet processors.* SIGCOMM Comput. Commun. Rev., 44(3):87–95, julho 2014, ISSN 0146-4833. `http://doi.acm.org/10.1145/2656877.2656890`. x, 26, 27, 43, 46

[5] Foundation, Open Network: *Openflow switch specification 1.5.* dezembro 2014. xii, 21, 22

[6] Foudantion, Open Network: *Software-defined networking (sdn) definition*, 2018. `https://www.opennetworking.org/sdn-definition/`. 1, 18

[7] Casado, M., T. Koponen, R. Ramanathan e S. Shenker: *Virtualizing the network forwarding plane.* WORKSHOP ON PROGRAMMABLE ROUTERS FOR EXTENSIBLE SERVICES OF TOMORROW, PRESTO, 2010. 1, 19, 24, 26

[8] McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker e J. Turner: *Openflow: Enabling innovation in campus networks.* SIGCOMM Comput. Commun. Rev., 38(2):69–74, 2008, ISSN 0146-4833. `http://doi.acm.org/10.1145/1355734.1355746`. 1, 20

[9] A. Lara, A. Kolasani e B. Ramamurthy: *Network innovation using openflow: A survey.* IEEE communications surveys & tutorials, 16(1):493–512, 2014. 1, 18, 20, 24, 26

[10] Mattos, D. M. F. e O. C. M. B. Duarte: *Authflow: authentication and access control mechanism for software defined networking.* Em *Ann. Telecommun*, volume 71, páginas 607 – 615, dezembro 2016. 1, 29

[11] Lau, F., S. H. Rubin, M. H. Smith e L. Trajkovic: *Distributed denial of service attacks*. Em *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0*, volume 3, páginas 2275–2280 vol.3, outubro 2000. 1, 2, 7, 29

[12] Dridi, L. e M. F. Zhani: *Sdn-guard: Dos attacks mitigation in sdn networks*. Em *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, páginas 212–217, outubro 2016. 2, 37, 39, 41

[13] Mohammadi, R., R. Javidan e M. Conti: *Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks*. IEEE Transactions on Network and Service Management, 14(2):487–497, junho 2017, ISSN 1932-4537. 2, 34, 39, 41

[14] Fichera, S., L. Galluccio, S. Grancagnolo, G. Morabito e S. Palazzo: *Operetta: An openflow-based remedy to mitigate tcp synflood attacks against web servers*. Comput. Netw., 92(P1):89–100, 2015, ISSN 1389-1286. `http://dx.doi.org/10.1016/j.comnet.2015.08.038`. 2, 33, 39, 41

[15] Shin, S., Y. Vinod, P. Phillip e G. Guofei: *Avant-guard: Scalable and vigilant switch flow management in software-defined networks*. Em *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, páginas 413–424, New York, NY, USA, 2013. ACM, ISBN 978-1-4503-2477-9. `http://doi.acm.org/10.1145/2508859.2516684`. 2, 33, 39, 40, 41, 42, 59, 60, 74

[16] Heller, B., S. Rob e N. McKeown: *The controller placement problem*. Em *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, páginas 7–12, New York, NY, USA, 2012. ACM, ISBN 978-1-4503-1477-0. `http://doi.acm.org/10.1145/2342441.2342444`. 2, 28

[17] Consortium, The P4 Language: *P4 - language specification*. `https://p4.org/`. 2, 46

[18] Kreutz, D., F. M. V. Ramos e P. E. Verissimo: *Towards secure and dependable software-defined networks*. Em *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, páginas 55–60, New York, NY, USA, 2013. ACM, ISBN 978-1-4503-2178-5. `http://doi.acm.org/10.1145/2491185.2491199`. 2, 28, 41

[19] Carvalho, R., J. L. Bordim e E. A. P. Alchieri: *Entropy-based dos attack identification in sdn*. 21st Workshop on Advances in Parallel and Distributed Computational Models, 2019. 4

[20] Carvalho, R., L. R. Costa, J. L. Bordim e E. A. P. Alchieri: *Enhancing an sdn architecture with dos attack detection mechanisms*. Advances in Science, Technology and Engineering Systems Journal, 2020. 4

[21] Carvalho, R., L. R. Costa, J. L. Bordim e E. A. P. Alchieri: *New programmable data plane architecture based on p4 openflow agent*. 34th Advanced Information Networking and Applications, 2020. 4

[22] Singh, J., S. Kaur, G. Kaur e G. Kaur: *A detailed survey and classification of commonly recurring cyber attacks*. International Journal of Computer Applications (0975 – 8887), 141(10):15–19, 2016. 6

[23] J. Criscuolo, Paul: *Distributed denial of service: Trin00, tribe flood network, tribe flood network 2000, and stacheldraht ciac-2319*. página 19, fevereiro 2000. 6, 8, 9

[24] Carl, G., G. Kesidis, R. Brooks e R. Rai: *Denial-of-service attack-detection techniques*. IEEE Internet Computing, 10(1):82–89, 2006, ISSN 1089-7801. `http://dx.doi.org/10.1109/MIC.2006.5`. 6

[25] Specht, S. e L. Ruby: *Distributed denial of service: Taxonomies of attacks, tools, and countermeasures.* páginas 543–550, janeiro 2004. 6, 41

[26] Gondim, João J.C., Robson de Oliveira Albuquerque e Ana Lucila Sandoval Orozco: *Mirror saturation in amplified reflection distributed denial of service: A case of study using snmp, ssdp, ntp and dns protocols*. Future Generation Computer Systems, 2020, ISSN 0167-739X. `http://www.sciencedirect.com/science/article/pii/S0167739X19322745`. 7

[27] Feinstein, L., D. Schnackenberg, R. Balupari e D. Kindred: *Statistical approaches to ddos attack detection and response*. Em *Proceedings DARPA Information Survivability Conference and Exposition*, volume 1, páginas 303–314 vol.1, abril 2003. 7, 13

[28] Postel, J.: *Transmission Control Protocol*. RFC 793 (Standard), 1981. `http://www.ietf.org/rfc/rfc793.txt`, Updated by RFCs 1122, 3168. 7

[29] Kavisankar, L. e C. Chellappan: *A mitigation model for tcp syn flooding with ip spoofing*. Em *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, páginas 251–256, junho 2011. 7

[30] *Domain names - concepts and facilities*. RFC 1034, novembro 1987. `https://rfc-editor.org/rfc/rfc1034.txt`. 9

[31] Reilly, Denis, Harlan Stenn e Dieter Sibold: *Network Time Protocol Best Current Practices*. RFC 8633, julho 2019. `https://rfc-editor.org/rfc/rfc8633.txt`. 9

[32] Sattar, U., T. Naqash, M. R. Zafar, K. Razzaq e F. bin Ubaid: *Secure dns from amplification attack by using modified bloom filters*. Em *Eighth International Conference on Digital Information Management (ICDIM 2013)*, páginas 20–23, setembro 2013. 9

[33] Mockapetris, P: *RFC 1034 Domain Names - Concepts and Facilities*, 1987. `http://tools.ietf.org/html/rfc1035http://tools.ietf.org/html/rfc1034`. 9

[34] Rossow, Christian: *Amplification hell: Revisiting network protocols for ddos abuse*. janeiro 2014, ISBN 1-891562-35-5. 9

[35] Rudman, L. e B. Irwin: *Characterization and analysis of ntp amplification based ddos attacks*. Em *2015 Information Security for South Africa (ISSA)*, páginas 1–5, agosto 2015. 10

[36] Paxson, Vern: *An analysis of using reflectors for distributed denial-of-service attacks*. Computer Communication Review, 31, julho 2001. 10

[37] Yevsieieva, O. e S. M. Helalat: *Analysis of the impact of the slow http dos and ddos attacks on the cloud environment*. Em *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S T)*, páginas 519–523, outubro 2017. 11

[38] Shafieian, S., M. Zulkernine e A. Haque: *Cloudzombie: Launching and detecting slow-read distributed denial of service attacks from the cloud*. Em *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, páginas 1733–1740, outubro 2015. 11

[39] Garg, A. e P. Maheshwari: *A hybrid intrusion detection system: A review*. Em *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, páginas 1–5, janeiro 2016. 12

[40] S. Lee, H. Kim, J. Na e J. Jang: *Abnormal traffic detection and its implementation*. Em *The 7th International Conference on Advanced Communication Technology, 2005, ICACT 2005.*, volume 1, páginas 246–250, fevereiro 2005. 12

[41] You, Y., M. Zulkernine e A. Haque: *Detecting flooding-based ddos attacks*. Em *2007 IEEE International Conference on Communications*, páginas 1229–1234, junho 2007. 13

[42] Beitollahi, H. e G. Deconinck: *Analyzing well-known countermeasures against distributed denial of service attacks*. Computer Communications, 35(11):1312 – 1332, 2012, ISSN 0140-3664. `http://www.sciencedirect.com/science/article/pii/S0140366412001211`. 13, 14, 16

[43] Shannon, C.E: *A mathematical theory of communication*. 27:379–423, 1948. 13, 14

[44] Leu, F. e C. Pai: *Detecting dos and ddos attacks using chi-square*. Em *2009 Fifth International Conference on Information Assurance and Security*, volume 2, páginas 255–258, agosto 2009. 14

[45] Oshima, S., A. Hirakawa, T. Nakashima e T. Sueyoshi: *Dos/ddos detection scheme using statistical method based on the destination port number*. Em *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, páginas 206–209, setembro 2009. 14

[46] Tritilanunt, S., S. Sivakorn, C. Juengjincharoen e A. Siripornpisan: *Entropy-based input-output traffic mode detection scheme for dos/ddos attacks*. Em *2010 10th International Symposium on Communications and Information Technologies*, páginas 804–809, outubro 2010. 14, 43

[47] Kurihara, K. e K. Katagishi: *A simple detection method for dos attacks based on ip packets entropy values*. Em *2014 Ninth Asia Joint Conference on Information Security*, páginas 44–51, setembro 2014. 14

[48] Lima, M. S.: *Métodos adaptativos para detecção de clusters no espaço-tempo*, 2015. 14

[49] Walter, W. e O. Carvalho: *Aplicação individual e combinada dos gráficos de controle shewhart e cusum: uma aplicação no setor metal*. Gestão Produção, 20:271 – 286, junho 2013. 14

[50] Haining Wang, Danlu Zhang e K. G. Shin: *Change-point monitoring for the detection of dos attacks*. IEEE Transactions on Dependable and Secure Computing, 1(4):193–208, outubro 2004, ISSN 1545-5971. 15

[51] Alenezi, M. e M. J. Reed: *Denial of service detection through tcp congestion window analysis*. Em *World Congress on Internet Security (WorldCIS-2013)*, páginas 145–150, dezembro 2013. 15

[52] P. Barford, J. Kline, D. Plonka e A. Ron: *A signal analysis of network traffic anomalies*. Em *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, páginas 71–82, New York, NY, USA, 2002. ACM, ISBN 1-58113-603-X. `http://doi.acm.org/10.1145/637201.637210`. 16

[53] Li, L. e G. Lee: *Ddos attack detection and wavelets*. Em *Proceedings. 12th International Conference on Computer Communications and Networks (IEEE Cat. No.03EX712)*, páginas 421–427, outubro 2003. 16

[54] A. P. Braga, A. Po. de Leon F. de Carvalho e T. B. Ludemir: *Redes Neurais Artificias - Teoria e Aplicações*, volume 1. LTC, 2007. 16

[55] Bonifacio, J. M., A. M. Cansian, A. C. P. L. F. De Carvalho e E. S. Moreira: *Neural networks applied in intrusion detection systems*. Em *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, volume 1, páginas 205–210 vol.1, Maio 1998. 16

[56] Holland, J. H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. 1975, 1ª edição. 17

[57] Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1ª edição, 1989, ISBN 0201157675. 17

[58] Costa, C.B.B., A.C.R Elmer, M. C. Alves F. Rezende, M. R. W. Maciel e R. M. Filho: *Prior detection of genetic algorithm significant parameters: Coupling factorial design technique to genetic algorithm*. Chemical Engineering Science, 62(17):4780 – 4801, 2007, ISSN 0009-2509. `http://www.sciencedirect.com/science/article/pii/S0009250907003168`. 17

[59] Medeiros, A. L.: *Aplicabilidade de algoritmos genéticos para calibração de redes viárias urbanas microssimuladas*. Tese de Mestrado, Universidade Federal do Ceará, http://www.repositorio.ufc.br/handle/riufc/7992, dezembro 2012. 17

[60] Zhao, T., D. C. Lo e K. Qian: *A neural-network based ddos detection system using hadoop and hbase*. Em *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, páginas 1326–1331, agosto 2015. 17

[61] Hsieh, C. e T. Chan: *Detection ddos attacks based on neural-network using apache spark*. Em *2016 International Conference on Applied System Innovation (ICASI)*, páginas 1–4, maio 2016. 17

[62] Guo, R., G. Chang, R. Hou, Y. Qin, B. Sun, A. Liu, Y. Jia e D. Peng: *Research on counter bandwidth depletion ddos attacks based on genetic algorithm*. Em *Third International Conference on Natural Computation (ICNC 2007)*, volume 4, páginas 155–159, agosto 2007. 17

[63] Wang, S. e R. Guo: *Ga-based filtering algorithm to defend against ddos attack in high speed network*. Em *2008 Fourth International Conference on Natural Computation*, volume 1, páginas 601–607, outubro 2008. 17

[64] Gong, Yili, Wei Huang, Wenjie Wang e Yingchun Lei: *A survey on software defined networking and its applications*. Frontiers of Computer Science, 9(6):827–845, dezembro 2015, ISSN 2095-2236. `https://doi.org/10.1007/s11704-015-3448-z`. 18

[65] Ahmad, I., S. Namal, M. Ylianttila e A. Gurtov: *Security in software defined networks: A survey*. IEEE Communications Surveys Tutorials, 17(4):2317–2346, Fourthquarter 2015, ISSN 1553-877X. 18, 41

[66] Taha, A.: *Software-defined networking and its security*, 2014. 19

[67] Chowdhury, D.: *Forces: An elastic routing architecture for nextgen sdn.* agosto 2016. 20

[68] Stancu, A., A. Avram, M. Skorupski, A. Vulpe e S. Halunga: *Enabling sdn application development using a netconf mediator layer simulator*. Em *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, páginas 658–663, July 2017. 20, 26

[69] N. Gude, T. Koponen, J. Pettit B. Pfaff M. Casado N. McKeown e S. Shenker: *Nox: Towards an operating system for networks*. Computer Communication Review, 38:105–110, janeiro 2008. 24, 25, 30

[70] Murpyh, M.: *Nox*. `https://github.com/noxrepo/pox`. 25, 30

[71] Erickson, D.: *The beacon openflow controller*. Em *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, páginas 13–18, New York, NY, USA, 2013. ACM, ISBN 978-1-4503-2178-5. `http://doi.acm.org/10.1145/2491185.2491189`. 25, 30

[72] Khondoker, R., A. Zaalouk, R. Marx e K. Bayarou: *Feature-based comparison and selection of software defined networking (sdn) controllers*. Em *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, páginas 1–7, janeiro 2014. 25, 30

[73] Projects, The Linux Foundation: *Open day light - platform overview*. `https://www.opendaylight.org/what-we-do/odl-platform-overview`. 26, 30

[74] RYU: *Ryu sdn framework*. `https://osrg.github.io/ryu/`. 26

[75] Miano, S., F. Risso e H. Woesner: *Partial offloading of openflow rules on a traditional hardware switch asic*. Em *2017 IEEE Conference on Network Softwarization (NetSoft)*, páginas 1–9, July 2017. 26

[76] Shin, S., P. Porras, V. Yegneswaran, M. Fong, G. Gu e M. Tyson: *Fresco: Modular composable security services for software-defined networks*, 2013. 28

[77] NSnam: *Ns-3 - network simulator*. `https://www.nsnam.org/`. 30

[78] Estinet: *Estinet - the network simulator and emulator that supports sdn/openflow*. `http://www.estinet.com/ns/`. 30

[79] Team, Mininet: *Mininet - an instant virtual network on your laptop (or other pc)*. `http://mininet.org/`. 30, 59

[80] Imran, M., H. Durad, F. Khan e A. Derhab: *Toward an optimal solution against denial of service attacks in software defined networks*. Future Generation Computer Systems, 92, setembro 2018. 32

[81] Sonchack, J., Adam J. Aviv, Eric Keller e Jonathan Smith: *Enabling practical software-defined networking security applications with ofx*. janeiro 2016. 33, 39

[82] Ambrosin, M., M. Conti, F. De Gaspari e R. Poovendran: *Lineswitch: Tackling control plane saturation attacks in software-defined networking*. IEEE/ACM Transactions on Networking, 25(2):1206–1219, abril 2017, ISSN 1063-6692. 34, 39

[83] Ubale, T. e A. Jain: *Srl: An tcp synflood ddos mitigation approach in software-defined networks*. páginas 956–962, março 2018. 35, 39

[84] Dang, V. T., T. T. Huong, N. H. Thanh, P. N. Nam, N. N. Thanh, A. Marshall e S. Furnell: *Sdn-based syn proxy—a solution to enhance performance of attack mitigation under tcp syn flood*. The Computer Journal, 62(4):518–534, April 2019. 35, 39

[85] Bera, P., A. Saha e S. K. Setua: *Denial of service attack in software defined network*. Em *2016 5th International Conference on Computer Science and Network Technology (ICCSNT)*, páginas 497–501, Dec 2016. 36, 39

[86] Kuerban, M., Y. Tian, Q. Yang, Y. Jia, B. Huebert e D. Poss: *Flowsec: Dos attack mitigation strategy on sdn controller*. Em *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, páginas 1–2, Aug 2016. 36, 39

[87] M. Nugraha, I. Paramita, A. Musa D. Choi e B. Cho: *Utilizing openflow and sflow to detect and mitigate syn flooding attack*. Journal of Korea Multimedia Society, 8(8), agosto 2014. http://dx.doi.org/10.9717/kmms.2014.17.8.988. 37, 39

[88] Kalkan, K., G. Gür e F. Alagöz: *Filtering-based defense mechanisms against ddos attacks: A survey*. IEEE Systems Journal, 11(4):2761–2773, dezembro 2017, ISSN 1932-8184. 42

[89] Tajer, J., A. Makke, O. Salem e A. Mehaoua: *A comparison between divergence measures for network anomaly detection*. Em *2011 7th International Conference on Network and Service Management*, páginas 1–5, outubro 2011. 43

[90] Lapolli, A. C., J. A. Marques e L. P. Gaspary: *Offloading real-time ddos attack detection to programmable data planes*. Em *IFIP/IEEE International Symposium on Integrated Network Management (IM 2019)*, 2019. 43

[91] Sviridov, G., M. Bonola, A. Tulumello, P. Giaccone, A. Bianco e G. Bianchi: *Lodge: Local decisions on global states in prograna naable data planes*. Em *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, páginas 257–261, June 2018. 43

[92] *Apache thrift*. https://thrift.apache.org/. 43, 67

[93] Scapy: *Packet crafting for python2 and python3*. https://scapy.net/. 60

[94] Sanfilippo, S.: *Hping - active network security*. http://www.hping.org/. 60

[95] Bernstein, D. J.: *Syn cookies*. http://cr.yp.to/syncookies.html. 67