Segurança e Automação em Redes SDN: Desenvolvendo uma Ferramenta de Automação de Testes de Segurança em Redes SDN

Ryan Matheus da Silva Leal

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2024

Ryan Matheus da Silva Leal

Segurança e Automação em Redes SDN: Desenvolvendo uma Ferramenta de Automação de Testes de Segurança em Redes SDN

Monografia apresentada ao curso de Ciência da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Iguatemi Eduardo da Fonseca

Novembro de 2024

Catalogação na publicação
Seção de Catalogação e Classificação

L435s Leal, Ryan Matheus da Silva.
Segurança e automação em redes SDN: desenvolvendo uma ferramenta de automação de testes de segurança em redes SDN / Ryan Matheus da Silva Leal. - João Pessoa, 2024.
43 f. : il.

Orientação: Iguatemi Eduardo da Fonseca.
TCC (Graduação) - UFPB/CI.

1. Redes SDN. 2. Automação. 3. Cibersegurança. 4. Testes de segurança. I. Fonseca, Iguatemi Eduardo da. II. Título.

UFPB/CI CDU 004.056:004.7

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computação, intitulado ***Segurança e Automação em Redes SDN: Desenvolvendo uma Ferramenta de Automação de Testes de Segurança em Redes SDN*** de autoria de Ryan Matheus da Silva Leal, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Iguatemi Eduardo da Fonseca
Universidade Federal da Paraíba

Prof. Dr. João Henrique Gonçalves Corrêa
Universidade Federal do Ceará

Prof. Dr. Waslon Terllizzie Araújo Lopes
Universidade Federal da Paraíba

Coordenador(a) do Curso de Ciência da Computação
Giorgia de Oliveira Mattos
CI/UFPB

João Pessoa, 11 de novembro de 2024

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

"Para pequenas criaturas como nós, a vastidão é suportável somente através do amor."

Carl Sagan

Agradecimentos

Primeiramente a Deus, pois sem ele, não seria nada.

Aos meus pais, José Ronaldo e Maria da Salete, pois mesmo em situações angustiantes, permaneceram firmes para que pudéssemos olhar mais longe, sendo exemplo de perseverança, parceria e fé.

Aos meus irmãos, de sangue ou não, com os quais tive a oportunidade de compartilhar momentos únicos: Adriel Cabral, Arthur Leal, Raphael Leite e Rayrone Leal.

À Sara, por ser um porto seguro em minha vida. Uma parceira para todos os momentos de ontem, hoje, amanhã e sempre.

Aos meus familiares, que apoiaram direta e indiretamente minha trajetória.

Aos amigos que fiz pelo caminho, que compartilharam ensinamentos e bons momentos durante esse percurso, lembro em especial de: Miguel Amadio, Rogério Lucas, Matheus César e Kelvyn Lenis.

Aos meus professores, que me ensinaram lições valiosas e me ajudaram a ser o profissional e ser humano que sou.

Ao prof. Dr. Iguatemi E. Fonseca, pelos ensinamentos e colaborações em diversos trabalhos durante a graduação.

Aos membros da banca examinadora, professores Dr. Iguatemi E. Fonseca, Dr. João H. G. Corrêa e Dr. Waslon T. A. Lopes, por aceitarem o convite para participar da banca e pelas sugestões e contribuições dadas ao trabalho.

À RNP (Rede Nacional de Ensino e Pesquisa) pelo apoio durante a pesquisa do projeto FAIR-5G. E ao CNPq pela oportunidade de participar do programa PIBIC.

Resumo

O avanço das redes definidas por software (SDN - *Software Defined Networks*) trouxe novos desafios e oportunidades para a segurança de redes. A importância desse tema é clara, principalmente, ao analisar a ampla adoção dessa arquitetura em, por exemplo, redes celulares de quinta geração (5G). Este trabalho apresenta o desenvolvimento de uma ferramenta de automação para testes de segurança em redes SDN, visando integrar a segurança aos ciclos de CI/CD (CI - *Continuous Integration*, CD - *Continuous Delivery*). A ferramenta reduz os erros humanos gerados pela complexidade de configuração dessas redes, de forma a testar a segurança sempre que novas configurações sejam aplicadas, reduzindo vulnerabilidades em ambiente de produção. Nos experimentos realizados, a ferramenta foi utilizada para simular ataques como inundação de pacotes TCP, UDP e ICMP, além de ataques de inundação de endereços MAC. Os resultados mostram que, mesmo em um ambiente emulado, ataques de negação de serviço podem causar indisponibilidade significativa nos controladores de redes SDN, como o ONOS, afetando tanto a conectividade entre dispositivos quanto o desempenho dos serviços oferecidos pela rede. A ferramenta proposta mostrou-se eficiente em automatizar os testes de segurança, sendo uma solução prática para ambientes que exigem agilidade e precisão nas suas operações de manutenção.

Palavras-chave: Redes SDN, Automação, Cibersegurança, Testes de Segurança.

Abstract

The advance of software-defined networks (SDN) has introduced new challenges and opportunities for network security. The importance of this topic is evident, especially when analyzing the widespread adoption of this architecture in networks such as 5G. This work presents the development of an automation tool for security testing in SDN networks, aiming to integrate security into CI/CD cycles. The tool is designed to mitigate human errors caused by the complexity of configuring these networks, allowing for security testing whenever new configurations are applied, thereby reducing vulnerabilities in production environments. In the experiments conducted, the tool was used to simulate attacks such as TCP, UDP, and ICMP packet flooding, as well as MAC address flooding attacks. The results show that, even in an emulated environment, denial-of-service attacks can cause significant downtime in SDN controllers, such as ONOS, affecting both connectivity between devices and the performance of network services. The proposed tool provides efficient in automating security testing, serving as a practical solution for environments that require agility and precision in their maintenance operations.

Keywords: SDN Networks, Automation, Cybersecurity, Security Testing.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

API - *Application Programming Interface* (Interface de Programação de Aplicações)
ARP - *Address Resolution Protocol* (Protocolo de Resolução de Endereço)
CAM - *Content Addressable Memory* (Memória de Conteúdo Endereçável)
CD - *Continuous Delivery* (Entrega Contínua)
CI - *Continuous Integration* (Integração Contínua)
CLI - *Command Line Interface* (Interface de Linha de Comando)
DDoS - *Distributed Denial of Service* (Negação de Serviço Distribuída)
DoS - *Denial of Service* (Negação de Serviço)
ICMP - *Internet Control Message Protocol* (Protocolo de Mensagem de Controle de Internet)
MAC - *Media Access Control* (Controle de Acesso ao Meio)
MTD - *Moving Target Defense* (Defesa de Alvo em Movimento)
P4 - *Pre-Packet Processor provider*
REST - *Representational State Transfer* (Transferência de Estado Representacional)
SDN - *Software-Defined Networking* (Rede Definida por Software)
SYN - *Synchronize* (Sincronizar)
TCAM - *Ternary Content Addressable Memory* (Memória de Conteúdo Endereçável Ternária)
TCP - *Transmission Control Protocol* (Protocolo de Controle de Transmissão)
UDP - *User Datagram Protocol* (Protocolo de Datagramas de Usuário)

Sumário

1 Introdução

Cada vez mais, as redes definidas por *software* ou redes SDN (*Software Defined Networks*) tem ganhado destaque devido a suas diversas aplicações em áreas como as redes 5G. Isso se deve a sua maior capacidade de controle, gerenciamento e programabilidade, quando comparada com a arquitetura tradicional das redes. Essa capacidade surge a partir de características como a centralização da lógica de controle, por meio de um elemento chamado *controlador* e também pela divisão da rede em diferentes planos. Sendo divididos em plano de dados, onde estão os dispositivos físicos da rede, plano de controle, onde está localizado o controlador da rede e o plano de aplicação, onde estão as aplicações que fazem uso dos recursos da rede [2]. Esses planos se comunicam por meio de APIs (API - *Application Programming Interface*), sendo elas nomeadas de *Northbound* e *Southbound*. A primeira é utilizada pelas aplicações para se comunicar com o controlador, por meio de API REST, interface gráfica e interface de linha de comando. Enquanto a segunda, é utilizada para comunicação entre o plano de dados e o de controle, utilizando protocolos como o Openflow e a linguagem de processamento de pacotes, P4, utilizada nas redes SDN mais modernas.

Contudo, novas vulnerabilidades também surgem, principalmente com a centralização da lógica de controle, a partir do controlador da rede, que é explorado por meio de ataques de negação de serviço. Além disso, falhas de segurança são acentuadas por erros de configuração geradas por humanos, principalmente devido ao aumento da complexidade que surge com essas novas camadas de tecnologia. A Figura 1 apresenta as cinco maiores vulnerabilidades, de acordo com os dados da OWASP [1]. É possível observar que erros de configuração de segurança continuam entre os principais problemas de segurança encontrados em diversas aplicações, não se limitando a redes SDN, mas sendo de grande relevância para uma variedade de contextos.

Uma das maneiras de evitar que erros de configuração gerem vulnerabilidades a serem exploradas por atacantes no ambiente de produção — o espaço onde o sistema opera com dados reais e é acessado por usuários finais — é manter uma rotina de execução de testes de segurança. É padrão atualmente que as aplicações façam uso de ciclos de CI/CD (CI - *Continuous Integration*, CD - *Continuous Delivery*), testando em ambiente de manutenção antes que as mudanças sejam enviadas para o ambiente de produção e são monitoradas para que novas configurações sejam criadas, com base no que foi obtido dessa análise. A ideia é que as redes SDN também façam parte desses ciclos de manutenção e sejam testadas antes de que novas configurações de rede sejam utilizadas em produção, não somente com testes de funcionalidades, mas também testes de segurança, que garantam que as novas configurações sejam seguras para o ambiente de operação [3].

Esse trabalho busca criar uma ferramenta para automação de testes de segurança,

Figura 1: Erros de configuração de segurança entre as 5 maiores vulnerabilidades da OWASP. Fonte: Adaptado de [1]

a qual pode ser utilizada em ambientes de CI/CD. Sendo simples de realizar manutenção, de forma que o usuário não precise de um conhecimento técnico profundo de programação para compreender seu funcionamento. A ferramenta é modularizada, de forma que os módulos possam ser alterados sem prejudicar o funcionamento geral do código e que seja simples de implementar novas funcionalidades, permitindo que os usuários possam utilizá-la de forma personalizada.

Como segundo objetivo, realizar testes com ataques de negação de serviço, a serem executados por meio da ferramenta e assim analisar os impactos desse tipo de ataque em um ambiente de experimentação. A emulação sendo mais próxima possível de um ambiente real, com uso de um controlador de rede reconhecido no meio acadêmico e industrial, sendo também utilizada uma ferramenta para emular a redes SDN. Dessa forma, podem ser feitas implementações de ataques para a ferramenta e analisar seu funcionamento em um teste de segurança no ambiente emulado.

1.1 Definição do Problema

As redes SDN são parte fundamental do futuro das novas aplicações, em especial em redes OpenRAN 5G, que necessitam da capacidade de personalização e controle que esse tipo de rede pode prover. Porém, com tantas capacidades e possibilidades, as equipes de manutenção são sobrecarregadas, levando a erros humanos e falhas em configurações de segurança. Fazendo com que as redes sejam expostas no ambiente de produção sem qualquer teste de segurança prévio e, consequentemente, estando vulneráveis a, por exemplo, negações de serviço. Nesse tipo de rede, os atacantes tem como foco gerar indisponibilidade no elemento controlador centralizado, criando um ponto de falha, que pode prejudicar todos os usuários dos serviços das redes conectadas ao controlador.

1.2 Objetivo geral

O objetivo geral deste trabalho é desenvolver um protótipo de ferramenta de automação de testes de segurança em redes SDN. Visando promover a prática de testes com ataques diversos durante os ciclos de manutenção e facilitar o trabalho das equipes responsáveis da rede, livrando o processo de erros humanos.

1.3 Objetivos específicos

Para alcançar o objetivo geral, buscou-se atingir os seguintes objetivos específicos:

- Realizar uma análise geral de redes SDN e suas aplicações, entendendo seu funcionamento e casos de uso;
- Pesquisar tópicos de segurança voltados para redes SDN, visando entender os principais problemas conhecidos e explorar trabalhos relacionados com automação focado em segurança em SDN;
- Planejar o ambiente de testes de redes SDN, buscando ferramentas que possam auxiliar na emulação do ambiente e na criação dos esboços da ferramenta;
- Implementar as funcionalidades da ferramenta de testes e realizar experimentos no ambiente criado para a pesquisa, gerando testes de segurança em ambiente controlado.

1.4 Estrutura da monografia

O restante do texto está dividido da seguinte forma:

- **Capítulo 2:** Exploram-se os conceitos relacionados a redes SDN, suas aplicações e pontos importantes da segurança desse tipo de rede, trazendo ainda uma seção de trabalhos relacionados voltados para automação em segurança de redes SDN;
- **Capítulo 3:** É apresentada a metodologia desenvolvida no processo de criação da ferramenta, do ambiente emulado e do cenário de testes, com diagramas e explicações do funcionamento prático e processo de desenvolvimento;
- **Capítulo 4:** É feita uma análise aprofundada dos testes realizados, observando cada caso e suas implicações para o ambiente de testes, com imagens que demonstram esse processo;
- **Capítulo 5:** São feitas as conclusões do trabalho, apresentando alguns dos resultados obtidos na pesquisa e novas ideias para trabalhos futuros.

2 Conceitos Gerais e Revisão da Literatura

2.1 Redes SDN

Redes Definidas por Software ou redes SDN (SDN - *Software Defined Networks*) é uma arquitetura de redes de computadores que difere das redes tradicionais ao adicionar conceitos e funcionalidades que promovem maior controle e gerenciamento no uso dos recursos da rede [4]. Dentre as principais características das redes SDN, destacam-se a programabilidade de aplicações de rede, permitindo o uso dos recursos de maneira dinâmica e controlada, além da distinção de funcionalidades por meio de planos. Características que permitem flexibilidade no gerenciamento da rede SDN, a qual é dividida em plano de aplicação, plano de controle e plano de dados.

O ponto chave se encontra no plano de controle, onde existe um elemento controlador que gera um nível maior de personalização e inteligência à rede, permitindo a programação de funcionalidades, assim, possibilitando que regras e funções personalizadas sejam criadas, favorecendo as atividades de gerenciamento da rede [5]. O plano de dados é responsável pelas funções de análise de regras e roteamento dos pacotes, é nesse plano que os dispositivos físicos da rede, como roteadores e *switches* são encontrados. Com a chegada de novos pacotes, novas regras de encaminhamento são necessárias, assim caso seja necessário, são enviados os cabeçalhos dos pacotes ao plano de controle, visando estabelecer uma nova regra que o enquadre, preenchendo as tabelas de fluxos dos *switches*. O plano de aplicação é onde são executados os aplicativos que fazem uso dos recursos da rede e que tem a possibilidade de modificar e personalizar com base nas necessidades de uso desses recursos, promovendo adaptabilidade e automação para os serviços que fazem uso da rede [6].

Para que os planos possam trabalhar em conjunto, a comunicação entre os planos é essencial, assim, os controladores de redes SDN possuem duas *APIs* para comunicação, *Southbound* - onde são realizadas as interações com os dispositivos do plano de dados e *Northbound* - por onde é realizada a comunicação com a camada de aplicação. Por sua vez, no plano de dados, atualmente, utilizam-se duas principais tecnologias, o Openflow, que é um protocolo amplamente utilizado, sendo um dos primeiros padrões para redes SDN. Ele é simples de entender e utilizar, o que acaba sendo muito importante para entender o fluxo dentro de aplicações em SDN. Porém, a linguagem de processamento de pacotes P4 (P4 - *Pre-Packet Processor Provider*) tem ganhado cada vez mais espaço. Principalmente, devido sua robustez e por adicionar a capacidade de programação da lógica do plano de dados, isso permite que a personalização das redes seja ainda maior, gerando aplicações cada vez mais robustas e seguras. Porém, por ser mais recente e mais complexo, por enquanto, sua adoção é menor, o que deve mudar nos próximos anos, visto que é considerada uma tecnologia da próxima geração de redes SDN [7].

A Figura 2 mostra a arquitetura das redes SDN por meio de um diagrama, contendo as diferentes camadas e suas interfaces.

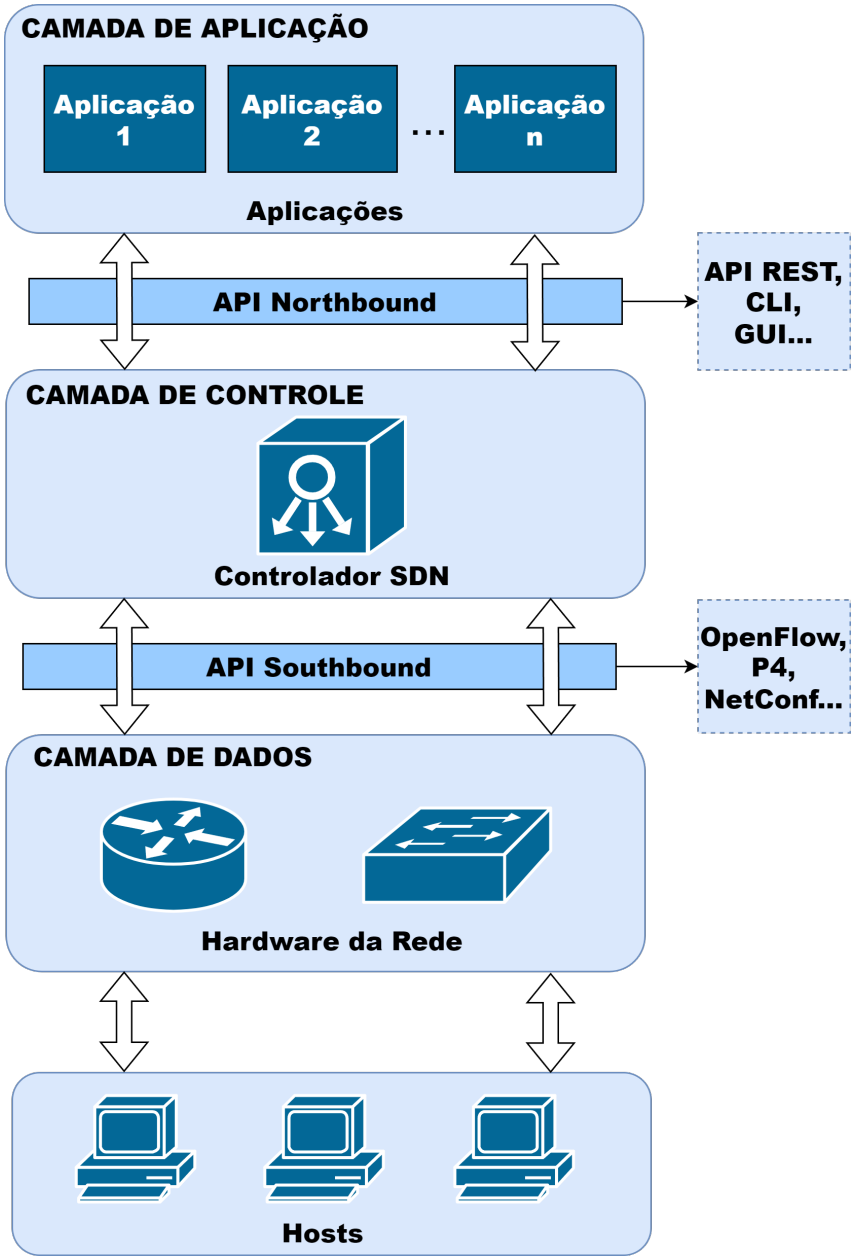

Figura 2: Diagrama da arquitetura das Redes SDN. Fonte: O autor

As redes SDN utilizam memórias TCAM (TCAM - *Ternary Content Addressable Memory*), que são mais rápidas que as memórias CAM (CAM - *Content Addressable Memory*), seu maior diferencial vem da utilização de 3 valores possíveis: 1, 0 e um valor coringa. Isso permite que regras mais abrangentes possam ser criadas. Porém, o custo energético e financeiro desse tipo de memória é mais alto, por isso, é utilizada em menor quantidade nos *switches* SDN [8]. A possibilidade de comunicação entre os planos e a capacidade de programabilidade de aplicações, tornam as redes SDN mais condizente com os requerimentos dos ciclos de desenvolvimento atuais [9]. Principalmente, porque

práticas de CI/CD necessitam da capacidade de personalização e controle para gerar aplicações capazes de se adaptar as necessidades, possibilitando que fluxos de trabalho sejam automatizados por meio de esquemas de orquestração [10], o que torna as atividades diárias de gerenciamento de recursos da rede mais simples.

2.2 Controladores de Redes SDN

Com a centralização da lógica de controle, os controladores de redes SDN desempenham um papel central no gerenciamento de redes, permitindo maior flexibilidade, automação e personalização na administração. Em um ambiente tradicional, o controle da rede é distribuído pelos dispositivos físicos, como switches e roteadores. No entanto, em uma arquitetura SDN, o controlador atua como uma entidade lógica central que possui uma visão global de toda a rede e comunica diretamente com os dispositivos no plano de dados para executar políticas e decisões de encaminhamento de pacotes [11]. A principal função do controlador SDN é tomar decisões inteligentes sobre como os fluxos de dados serão tratados pelos dispositivos da rede. São utilizadas interfaces de comunicação entre os planos, as APIs *Southbound* e *Northbound*, o que permite melhor gerenciamento dos recursos e proporciona uma rede mais ágil e responsiva às necessidades dinâmicas de tráfego e segurança [12].

Existem diferentes tipos de controladores SDN, sendo ONOS, OpenDaylight e Ryu alguns dos mais utilizados. O ONOS foi desenvolvido com o foco em alta escalabilidade e resiliência, voltado para grandes ambientes de produção. Já o OpenDaylight, também um controlador open-source, é altamente modular e flexível, sendo amplamente adotado pela indústria por sua capacidade de integrar diversos componentes e protocolos de rede [13]. O OpenDaylight oferece suporte a uma ampla gama de tecnologias, tornando-o uma solução adequada para diversos cenários de rede. O Ryu, por sua vez, é um controlador mais leve e simples, voltado para a experimentação e desenvolvimento acadêmico, sendo frequentemente utilizado em laboratórios e pesquisas de SDN [14]. Cada um desses controladores apresenta vantagens específicas conforme o contexto de uso, com o ONOS sendo mais indicado para redes distribuídas de grande porte, o OpenDaylight para ambientes heterogêneos, e o Ryu para redes experimentais e de menor escala.

Além das diferenças de desempenho e arquitetura, há também variações significativas no suporte a funcionalidades de segurança. O ONOS, por exemplo, se destaca pela implementação robusta de mecanismos de alta disponibilidade, essenciais para a segurança e continuidade do serviço em ambientes críticos. O OpenDaylight oferece uma maior personalização para implementar políticas de segurança, devido a sua modularização, enquanto o Ryu, devido à sua simplicidade, pode exigir mais intervenções manuais para a implementação de políticas de segurança sofisticadas [15]. Devido a essas características de centralização do controle e flexibilidade, as arquiteturas SDN têm sido amplamente ado-

tadas em diversos setores, especialmente em ambientes de nuvem e data centers, onde há uma necessidade crescente de eficiência operacional e respostas rápidas às mudanças nas demandas de rede [16]. A Tabela 1 traz uma comparação com alguns aspectos principais dos controladores ONOS, OpenDaylight e Ryu.

Tabela 1: Comparação entre controladores SDN.

Característica	**ONOS**	**OpenDaylight**	**Ryu**
Escalabilidade	Alta, adequado para redes de grande porte	Alta, com foco em ambientes heterogêneos	Limitada, mais utilizado em redes pequenas
Arquitetura	Distribuída com alta disponibilidade	Modular, com suporte a múltiplos protocolos	Monolítica, simples e leve
Uso Principal	Telecomunicações, data centers	Corporativo, telecomunicações, redes híbridas	Pesquisas acadêmicas, desenvolvimento
Desempenho	Ótimo em redes de alta demanda	Flexível e adaptável a diferentes cenários	Adequado para experimentação, menor demanda
Segurança	Foco em alta disponibilidade e mitigação	Flexível para políticas personalizadas	Requer personalização manual
Suporte a Protocolos	OpenFlow, Netconf, P4	OpenFlow, Netconf, BGP, MPLS	OpenFlow

2.3 Controlador ONOS

O *Open Network Operating System* (ONOS) é um controlador SDN open-source desenvolvido para fornecer uma plataforma escalável e resiliente, projetada principalmente para redes de grande porte, como operadoras de telecomunicações e data centers. Ele se destaca pela capacidade de lidar com redes de produção em larga escala, oferecendo alta disponibilidade, desempenho e flexibilidade na gestão de tráfego [17]. O ONOS foi desenvolvido com foco em três principais pilares: escalabilidade, alta disponibilidade e abstração da rede. Um dos diferenciais do ONOS é a sua arquitetura distribuída, que permite a implementação de múltiplos controladores trabalhando em conjunto, garantindo redundância e continuidade do serviço, mesmo em casos de falha de algum nó. Essa abordagem garante que, ao enfrentar uma falha, o sistema continue funcionando sem interrupções significativas [18]. Além disso, o ONOS suporta uma ampla gama de interfaces, incluindo OpenFlow, Netconf e mais recentemente, a tecnologia P4, as quais permitem o controle direto dos dispositivos de rede. A Figura 3 traz a arquitetura do controlador e mostra seu foco na distribuição, permitindo uma ampla gama de serviços com seus próprios protocolos, mas compartilhando dos núcleo distribuído do ONOS.

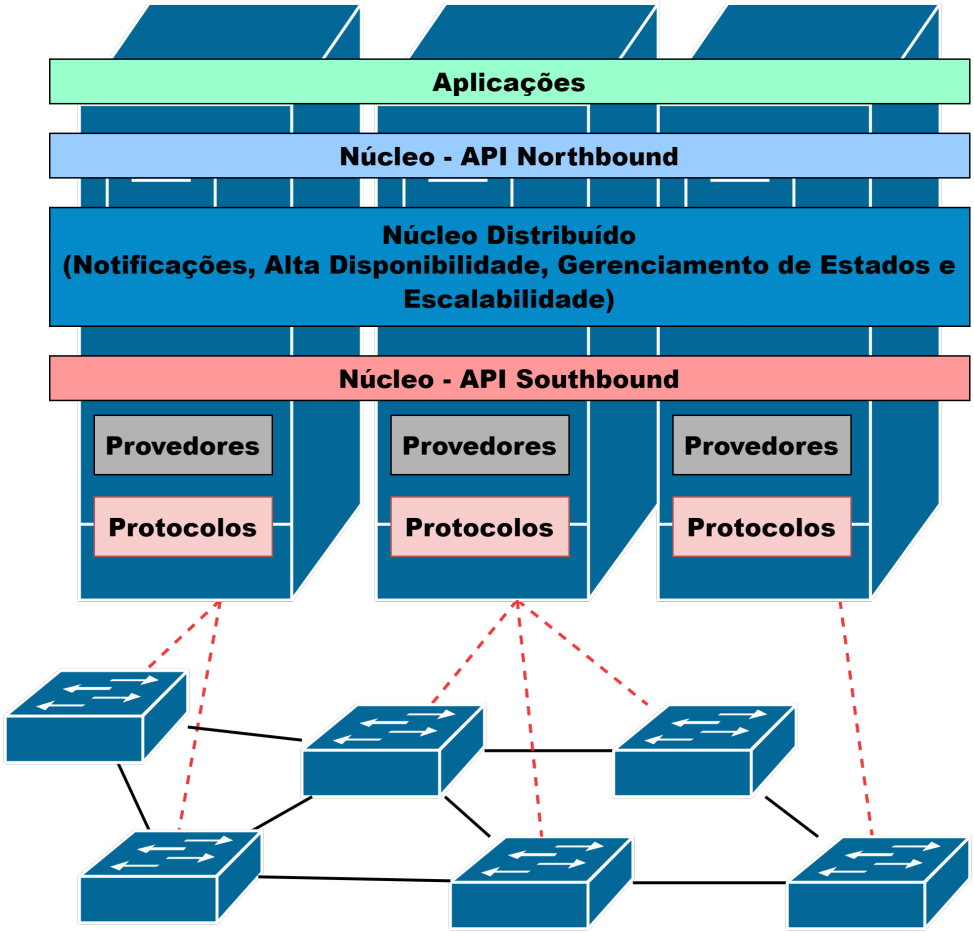

Figura 3: Diagrama da arquitetura do controlador ONOS.

Em termos de aplicações, o ONOS é amplamente utilizado em projetos de pesquisa e redes de telecomunicações, onde é necessário garantir alta performance e estabilidade. Sua modularidade permite a integração de diversas aplicações e funções de rede, como roteamento dinâmico, políticas de segurança e otimização de tráfego. Esse suporte a múltiplas aplicações faz com que ele seja uma escolha popular em ambientes que requerem personalização e flexibilidade na configuração de redes [19]. Como por exemplo em redes OpenRAN 5G, visto que elas utilizam tecnologias open-source, de forma a gerar independência de fabricantes e suporte para múltiplas plataformas. Isso ocorre, principalmente, porque o ONOS possui um ecossistema de desenvolvimento ativo, com várias contribuições da comunidade acadêmica e industrial, o que facilita a integração de novas funcionalidades e o suporte a diferentes tipos de dispositivos e redes.

Nos experimentos desenvolvidos nesse trabalho, o ONOS será utilizado como controlador da rede, onde será alvo da ferramenta, atuando em diferentes cenários de ataques, assim, será avaliada sua capacidade de gerenciamento e mitigação de falhas. A escolha do ONOS se deve à sua robustez e ao suporte para escalabilidade, características essenciais para testar cenários de ataque em redes de grande porte, como os que envolvem a implementação de ataques DoS e DDoS [20].

2.4 Segurança de Redes SDN

Com uma nova arquitetura, novas vulnerabilidades também surgem e no caso das redes SDN, um dos principais pontos de exploração é a centralização da lógica de controle. Visto que, caso o controlador pare de funcionar, toda a lógica de controle da rede é perdida, isso para todas as redes conectadas a ele, gerando o problema do acoplamento [21]. A Figura 4 é um exemplo de situação em que o problema do acoplamento pode afetar toda a rede de uma empresa, a partir de uma rede, separada logicamente, mas conectadas pelo controlador SDN, assim, em caso de falha no controlador, todas as redes conectadas a ele são afetadas. Dentre os diversos tipos de ameaças encontradas na literatura [22], destacam-se os ataques de negação de serviço ou DoS (*Denial of Service*), os quais visam ocasionar o esgotamento dos recursos do alvo, sendo muito eficientes contra a arquitetura e funcionamento desse tipo de rede, devido a sua característica de centralização.

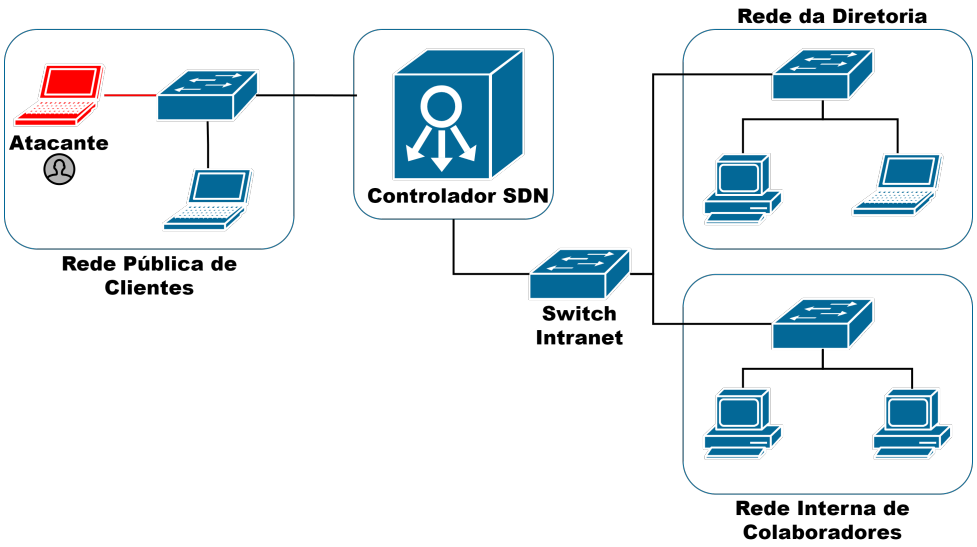

Figura 4: Diagrama representando o problema do acoplamento de switches. Fonte: O autor

Existem diversas variações de ataques DoS, uma das formas mais utilizadas pelos atacantes é feita com o auxílio de uma rede de máquinas, chamados *bots*, elaborando os ataques de negação de serviço distribuídos ou DDoS (*Distributed Denial of Service*). Eles podem ser classificados de acordo com a taxa do tráfego gerado durante a execução do ataque, podendo ser de alta taxa ou baixa taixa [23]. O primeiro é poderoso pois leva a aplicação alvo ao limite de seus recursos, consequentemente também os da rede, visto que o alto tráfego gera uma sobrecarga de dados que acabam levando ao mal funcionamento da rede e estresse das comunicações. Por sua vez, no segundo tipo, a negação de serviço surge com a manutenção de uma grande quantidade de conexões, mantendo um tráfego mínimo para que os dispositivos não sejam excluídos das regras de encaminhamento e consequentemente gera um acúmulo até causar a negação de serviço para usuários legítimos.

Por conta da característica mais silenciosa, esse tipo de ataque se torna mais difícil de detectar do que os de alta taixa.

A limitação gerada pela quantidade de memória TCAM em *switches* SDN é uma das vulnerabilidades que podem ser exploradas para gerar a negação de serviço. Um atacante pode realizar um ataque do tipo *Slow-TCAM* [24], onde o usuário malicioso utiliza uma rede de *bots* ou *botnet*, que são controlados por ele, para se conectarem com a rede SDN e manterem suas conexões ativas. São enviados pacotes antes que o *timeout* seja atingido, assim, preenchendo a tabela de fluxos do *switch* e negando serviço para qualquer usuário legítimo que tente se conectar.

Esse ataque pode ainda ser avançado para um *Slow-Saturation* [25], nesse tipo de ataque, a *botnet* é separada em dois subgrupos, ambos gerando tráfego de baixa taxa, porém, um dos subgrupos vai enviar picos de alta taxa. Assim, além de preencher a tabela de fluxos, também gera um mau funcionamento no *switch* que faz com que sejam enviados não somente os cabeçalhos dos pacotes para o controlador, mas sim os pacotes inteiros. Isso gera o estresse da comunicação entre o *switch* e o controlador, gerando a negação de serviço e erros na comunicação entre os dispositivos de rede e o controlador, caracterizando um ataque do tipo *Saturation*, que visa sobrecarregar o alvo.

A Figura 5 representa o processo de ataque do *Slow-Saturation*, é possível observar que, inicialmente, o tráfego parece legítimo e o controlador funciona normalmente, porém, com o disparo de requisições do *Saturation*, o controlador fica sobrecarregado e não consegue responder normalmente.

2.5 Trabalhos Relacionados

Ao pesquisar na literatura por soluções voltadas para automação em segurança para redes SDN, encontram-se materiais que apresentam diferentes métodos de automação para defesa das redes, com foco em proteger contra ataques no ambiente de produção e em tempo real. Essas soluções automatizadas de defesa são classificadas em reativas e proativas [26]. Abaixo, são analisados os principais trabalhos encontrados nas bases de conhecimento.

As soluções em defesa reativa são voltadas para análise de ameaças em tempo real da rede, ou seja, identificar um ataque enquanto é executado, normalmente, são utilizadas tecnologias de inteligência artificial. Em [27], por exemplo, que traz um método de detecção de ataques DDoS com uso de aprendizagem de máquina, utilizando dados de estatísticas de rede como base para o treinamento do modelo. Por sua vez, [28] propõe a criação de um sistema de defesa para as redes SDN que investigam os primeiros pacotes de um novo fluxo, em busca de traços de uso malicioso. As soluções proativas são voltadas para previnir a ocorrência de ataques, ou seja, atrapalhar o atacante antes de iniciar os

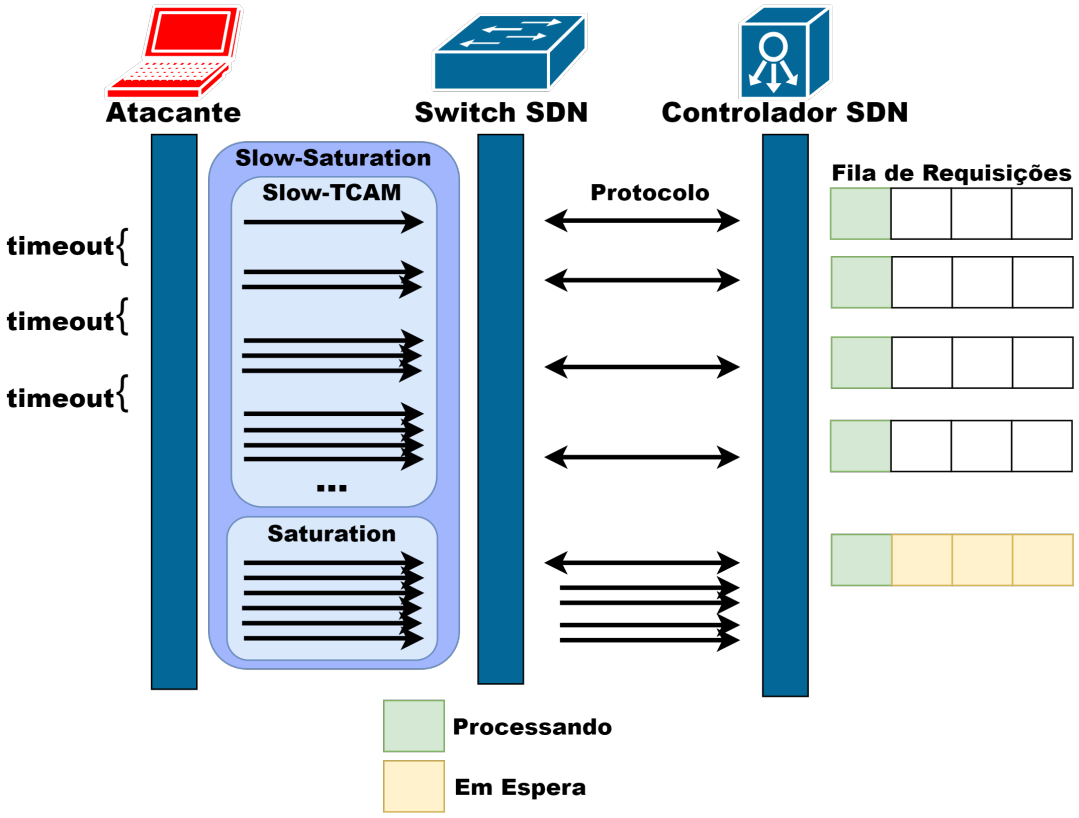

Figura 5: Representação do Ataque *Slow-Saturation*.

ataques. São utilizados conceitos como MTD (MTD - *Moving Target Defense*), onde a rede é modificada para atrapalhar o processo de reconhecimento da rede pelo atacante e *Deception*, onde são utilizadas armadilhas de rede, como *Honeypots*, para enganar o atacante, fazendo-o se expor ao achar que está atacando o alvo. Em [29] é proposto um framework que alia o uso de técnicas reativas com uso de *Shadow Controllers*, ou seja, o uso de controladores de fachada para responder a tráfegos maliciosos e para enganar o tráfego de escaneamento de rede do atacante. Fazendo uso também de MTD como técnica proativa, modificando os endereços e os dispositivos, gerando falhas no reconhecimento da rede feito pelo atacante. Outro exemplo de uso dessa técnica é apresentado em [30], porém, sua ênfase é na aplicação para redes IoT (*Internet of Things*), onde é proposto um framework mais leve para defesa de ataques DDoS e aplicável em redes com dispositivos de menor poder computacional.

Foi possível observar que os trabalhos são voltados para realizar a defesa das redes em tempo real e em ambiente de produção, porém, a ideia da ferramenta apresentada nesse trabalho é gerar um auxílio aos administradores da rede ao testar a segurança da rede SDN a cada nova mudança de configuração. Integrando a ferramenta aos ciclos de CI/CD e assim automatizando o processo de testes no ambiente da rede, tornando a rede mais segura e analisando suas capacidades de defesa contra os principais métodos de ataque antes mesmo que ela seja disponibilizada em produção.

3 Metodologia

Nessa seção, são apresentados os procedimentos realizados para a construção da ferramenta de testes de segurança automatizada para redes SDN, bem como a descrição do processo de testes em cenário emulado, especificando quais ferramentas foram utilizadas e a justificativa para cada escolha.

3.1 Criação da Ferramenta

Para promover a facilidade de uso da ferramenta de testes pelos usuários, a arquitetura da ferramenta foi pensada para promover a modularização. Os *scripts* de ataques podem ser inseridos e retirados da ferramenta como módulos, o que permite que os usuários possam criar seus próprios *scripts*, editando os já existentes. Além disso, por meio do arquivo de configuração da sessão de testes, o usuário tem a possibilidade de indicar os ataques utilizados em cada sessão de testes e fornecer informações do alvo, promovendo o controle da ferramenta pelo usuário. A ideia é que a ferramenta seja simples de integrar às operações de CI/CD e esquemas de orquestração, facilitando o processo de manutenção das redes. Ao executar os ataques, a ferramenta gera arquivos de *log* que servem de base para os administradores da rede identificarem pontos de falha, os quais devem ser tratados antes que a rede entre em ambiente de produção. Além disso, a possibilidade de utilizar os *logs* em algum sistema de *syslog* pode trazer uma visualização muito mais efetiva do processo de testes.

Para a criação do código da ferramenta, buscou-se uma linguagem de programação mais popular e com vasto material de apoio, visto que isso facilitaria para possíveis manutenções no código por parte da equipe que esteja utilizando-a. Tornando, assim, mais acentuada a capacidade de personalização da mesma, para isso, foi utilizada a linguagem Python na versão 3.x, a qual além das características acima, conta com diversas bibliotecas úteis e práticas para criar aplicações de diversos tipos. Para implementação da ferramenta, foram utilizadas as seguintes bibliotecas:

- **Scapy:** Biblioteca Python utilizada para manipular pacotes de rede e enviá-los pela rede, foi muito útil para criação de scripts personalizados;
- **Importlib:** Biblioteca que facilita o processo de gerenciamento e importação dos diversos arquivos e pastas de um projeto;
- **Subprocess:** Utilizada para realizar a execução de comandos do terminal a partir do código Python, permitindo inclusive que ferramentas do mercado sejam executadas, tornando o processo de testes mais produtivo;

- **Logging:** Biblioteca utilizada para gerar os *logs* padronizados da ferramenta, útil também para conexões com sistemas de *syslog* devido a compatibilidade;
- **Time:** Biblioteca que foi utilizada para cronometrar o tempo de execução de cada ataque e da sessão de teste em geral.

Na Figura 6 é apresentada a arquitetura e organização da ferramenta com seus arquivos e diretórios, onde é possível observar que o papel principal vem do arquivo *Main.py*. Ele que vai realizar o processo de leitura das configurações, que atualmente funciona com uso de listas e dicionários de python para armazenar as configurações. Então, com a leitura de quais ataques serão utilizados e os dados dos alvos, o arquivo principal recorre ao módulo de ataques para executar conforme especificado pelo usuário. Por fim, são geradas as saídas por CLI (CLI - *Command Line Interface*) e pelo arquivo de *logs*, que pode então ser utilizado por um sistema de *syslog*, gerando uma visualização dos dados.

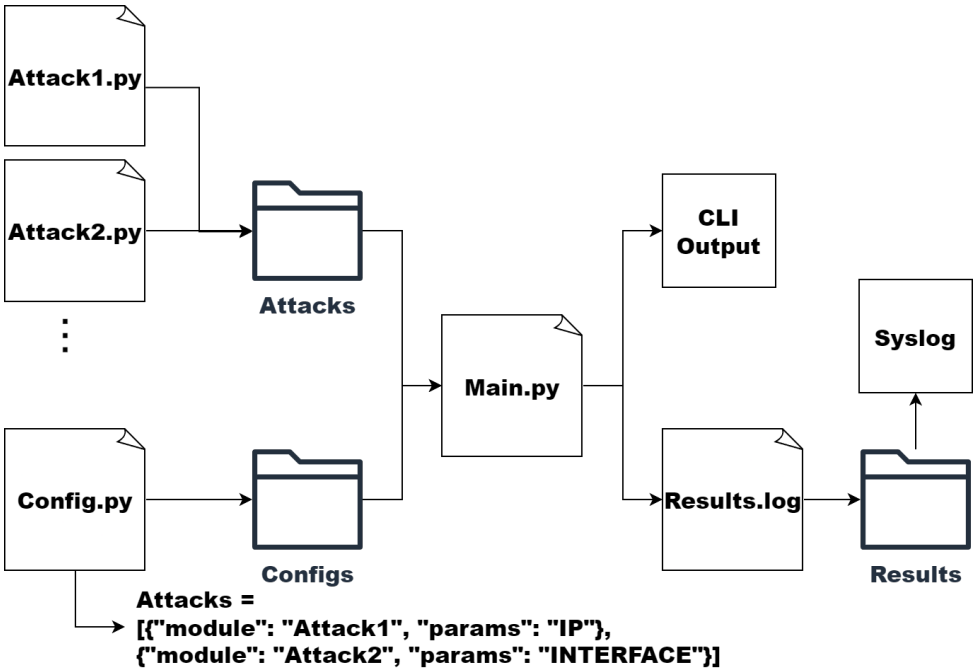

Figura 6: Arquitetura da ferramenta de automação de testes de segurança em redes SDN.

A Figura 7 apresenta o diagrama de sequências da ferramenta, mostrando os passos envolvidos na execução de uma sessão de testes, onde a interação do usuário se resume apenas a realizar as configurações iniciais e utilizar as saídas dos *logs*. Ficando sob responsabilidade da ferramenta a execução dos ataques, coleta de dados e organização e exibição dos resultados em *logs*, observa-se que quanto mais scripts são executados, menor é o trabalho para a equipe de manutenção e maior é a utilidade da ferramenta.

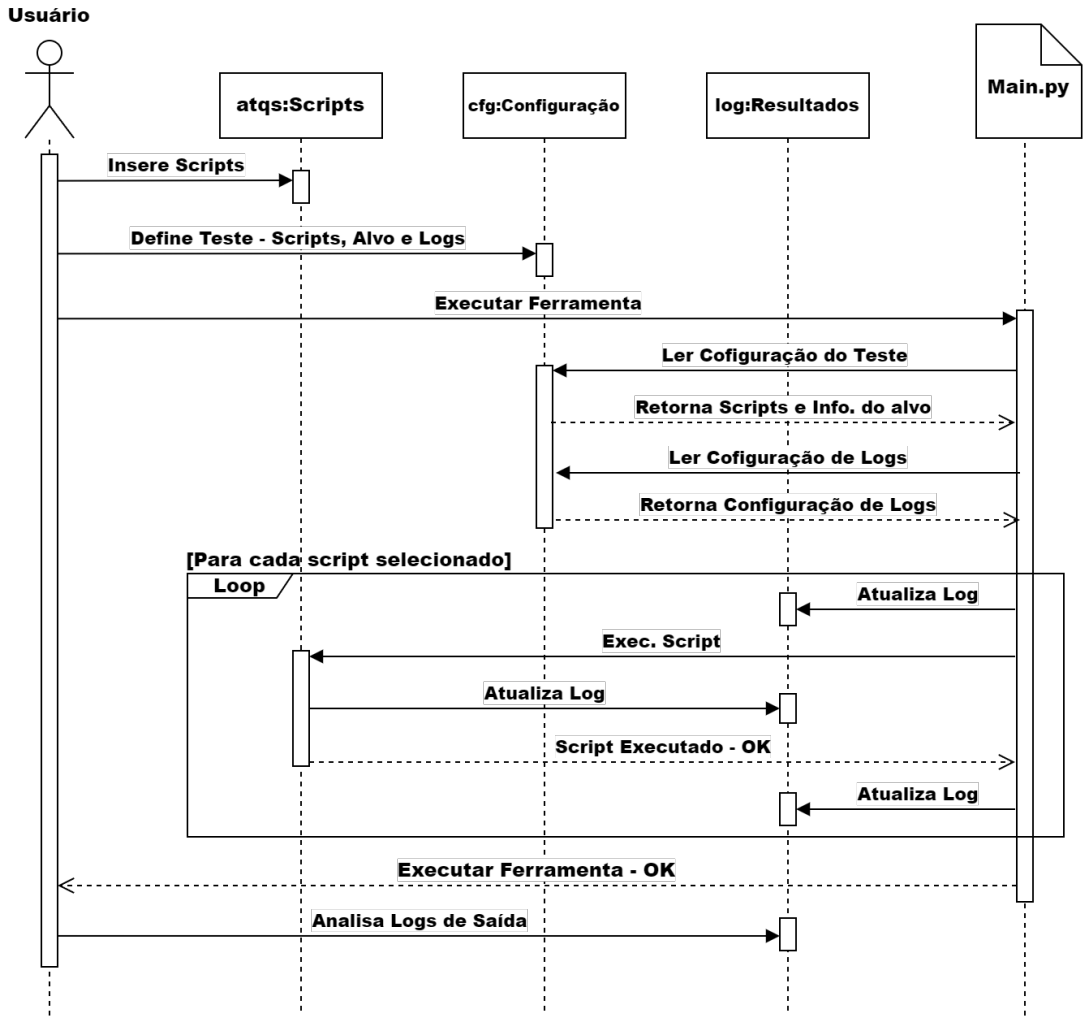

Figura 7: Diagrama de sequência da ferramenta. Fonte: O autor

O pseudocódigo que expressa o funcionamento da ferramenta é apresentado pelo Algoritmo 1. Tendo o usuário selecionado as configurações desejadas, observa-se a execução sequencial de cada script selecionado, posteriormente, são obtidos os resultados de execução. Os resultados trazem dados como o status de sucesso ou não na execução do script, o tempo em que ele foi executado e os impactos sendo representados pelos dados de saída do ataque. Sendo a base da ferramenta, seu funcionamento é simples, facilitando processo de integração com as soluções de orquestração e CI/CD.

3.2 Cenário de Testes

Conforme eram desenvolvidas as funcionalidades da ferramenta, foram realizados experimentos, para testar o funcionamento e medir o ganho de tempo com relação ao processo de execução manual do cenário de testes de segurança. Assim, nesta seção é feito o detalhamento desse processo.

Para emular o ambiente de redes SDN foi utilizado o Mininet, ferramenta am-

Algoritmo 1 Funcionamento Geral da Ferramenta de Testes

Require: Usuário deve ter preenchido o arquivo de configuração

```
1: ler_configuracao()                          ▷ Lê o arquivo de configuração
2: for cada ataque no arquivo de configuração do
3:     param ←coleta_parametros()              ▷ lê e verifica parâmetros do ataque
4:     resultado ←executar_ataque(parametros)  ▷ Executa o código do ataque
5:     status,tempoExecucao, impacto ←resultado
6:     salvar(status, tempoExecucao, Impacto)  ▷ Salva o log com os resultados
7: end for
8: exibe_Resultado()                           ▷ Finaliza com a exibição dos resultados
```

plamente utilizada para gerar topologias de redes desse tipo e utilizando o OpenFlow como protocolo de comunicação em camadas. Para o elemento controlador, inicialmente, optou-se pelo uso do Ryu, controlador básico e utilizado amplamente para estudos em SDN. Porém, conforme os testes foram progredindo, optou-se pelo uso de um controlador de rede mais robusto, então, foi utilizado o ONOS, sendo uma das soluções mais utilizadas no mercado e em ambiente acadêmico [20].

Inicialmente, foram utilizadas máquinas virtuais para hospedar o ambiente, onde o Mininet e a ferramenta eram executados em uma máquina virtual Kali Linux. Enquanto isso, os controladores eram disponibilizados em outra máquina virtual com o sistema Xubuntu, o que foi muito útil para os testes iniciais. Contudo, conforme os testes foram progredindo, o controlador passou a ser executado a partir dos servidores do Laboratório de Redes (LaR) da UFPB. Com o andamento da pesquisa, ocorreram limitações para acessar remotamente, então, foi utilizado um ambiente de nuvem para executar o ambiente de testes, o que facilitou bastante o processo de testes. A Figura 8 mostra essas mudanças no ambiente, separadas em 3 fases, desde as execuções em ambiente local, até a execução em ambiente de nuvem, onde foram executados os experimentos deste trabalho.

Para testar a ferramenta e sua utilidade na execução de ataques ao controlador, foram utilizados ataques voltados para gerar negação de serviço a partir de ataques de inundação. Utilizaram-se tanto *scripts* criados com uso do Scapy quanto por *scripts* que automatizam a execução do ataque a partir de ferramentas disponíveis no Kali Linux, abaixo estão listados todos os ataques utilizados [22]:

- ***MAC Address Flooding*:** Utilizando a ferramenta macof, gera inundação de pacotes ARP (ARP - *Address Resolution Protocol*) com endereços MAC (MAC - *Media Access Control*) aleatórios, congestionando a rede e podendo levar a tabela de endereços MAC do switch a ser completamente preenchida e gerar comportamentos inesperados;
- ***TCP SYN Flood*:** Ataque de inundação que utiliza o pacote TCP (TCP - *Transmission Control Protocol*) de início de conexão do tipo SYN para ocupar as portas

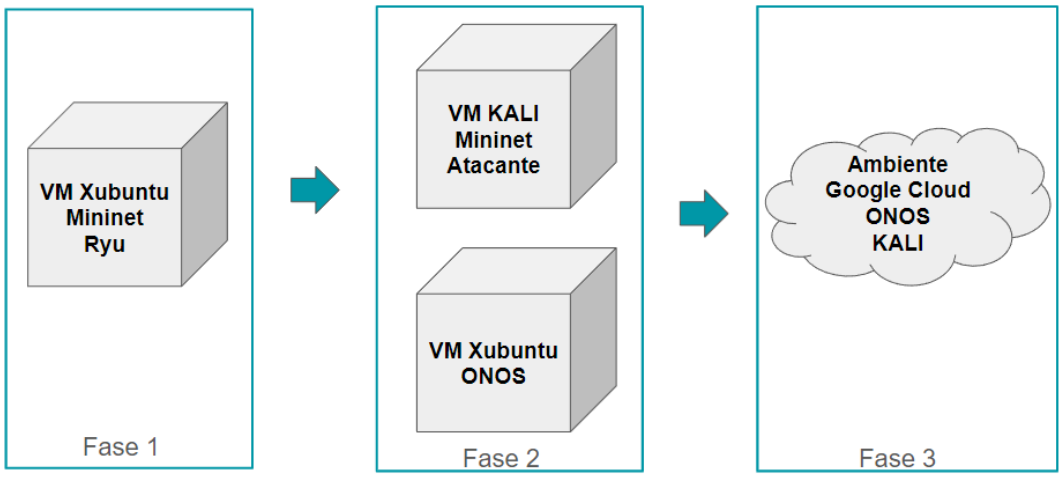

Figura 8: Representação das mudanças no ambiente utilizado nos experimentos. Fonte: O autor

do switch e gerar travamentos e negação de serviço. Nos testes foram utilizadas tanto uma versão implementada utilizando Scapy quanto a versão do Hping3, uma ferramenta que gera pacotes TCP/UDP/ICMP (UDP - *User Datagram Protocol*, ICMP - *Internet Control Message Protocol*) personalizados e permite também realizar inundações;

- ***UDP Flood*:** Ataque de inundação que utiliza pacotes do protocolo UDP para sobrecarregar o switch, os pacote UDP recebidos, normalmente são respondidos com um ping. Caso sobrecarregado, diversos pacotes de respostas são enviados, porém, maioria dos ataques modifica o IP de origem para enganar o alvo;

- ***ICMP Flood*:** Esse ataque também visa sobrecarregar o switch e gerar a negação de serviço, porém, nesse caso são enviadas múltiplas requisições ICMP do tipo *echo request*, as quais são respondidas com *echo response*. Porém, devido a alta quantidade de requisições e respostas, o alvo não consegue responder e é gerada a negação de serviço;

Para testar a ferramenta no ambiente, foram emuladas diferentes topologias de rede SDN com uso do Mininet em uma máquina virtual Kali Linux, a Figura 9 é uma representação da topologia completa utilizada, criada a partir do Cisco Packet Tracer. A Figura 10 é a topologia emulada vista na interface web do controlador ONOS, onde são representados somente os *switches* da rede. A ferramenta foi executada a partir de um *host* escolhido arbitrariamente, um exemplo de arquivo de configuração utilizado é exibido na Figura 11.

Com todo o ambiente em execução e os ataques selecionados no arquivo de configuração, bem como seus parâmetros, foram executados ataques contra o *switch*, o con-

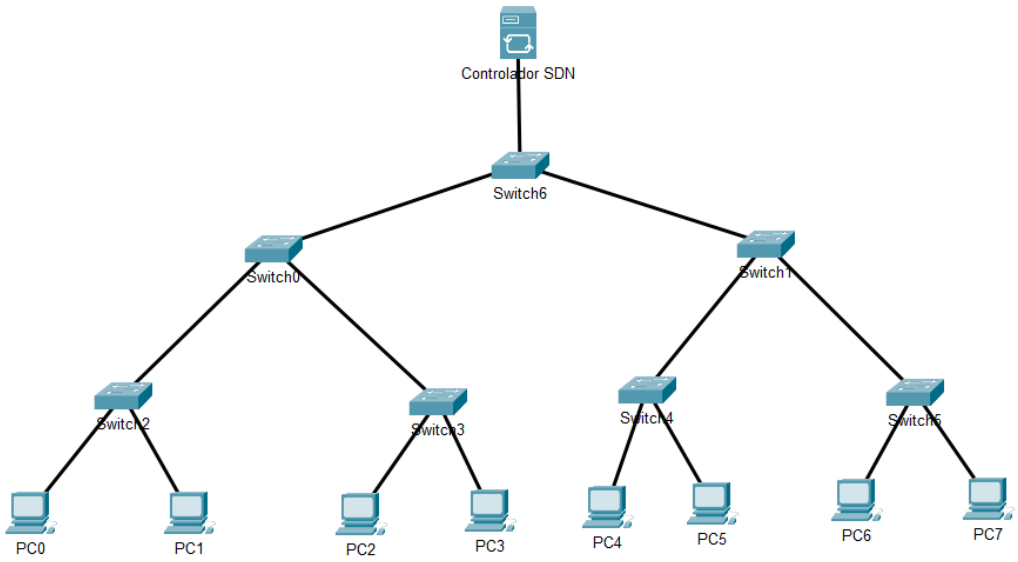

Figura 9: Representação da Topologia utilizada no Cisco Packet Tracer. Fonte: O autor

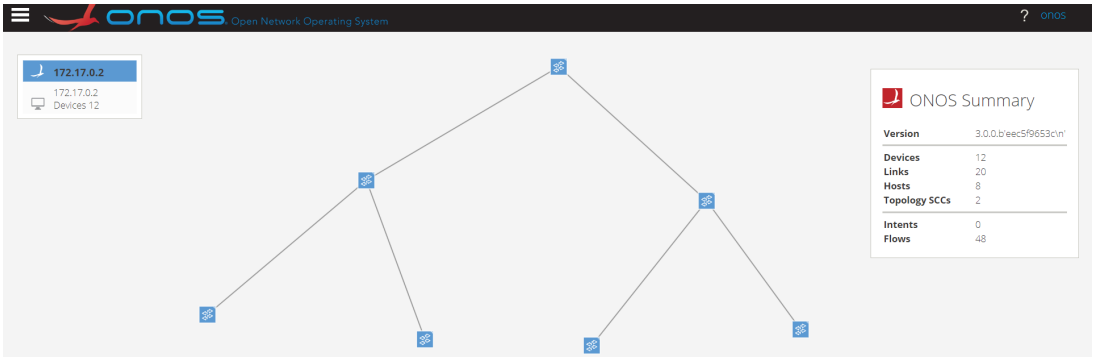

Figura 10: Topologia de rede emulada com Mininet, vista na interface gráfica do ONOS. Fonte: O autor.

trolador e as portas de seus serviços web e de controle da rede SDN. Para visualizar os resultados dos ataques, foram executados testes de disponibilidade com a ferramenta *pingall* do próprio Mininet, obtendo dados de disponibilidade entre os dispositivos da rede. Na Figura 12, é apresentada a disponibilidade da rede antes da execução da ferramenta, onde é possível visualizar a disponibilidade de cada um dos *hosts*, identificados pelos nomes de h1 até h8. Para esse caso, todos os *hosts* estão disponíveis e conseguem se comunicar entre si. Em caso de indisponibilidade, a comunicação será representada por um 'X' na linha correspondente ao *host* afetado. Além disso, a disponibilidade dos *switches* da rede também pode ser acompanhada visualmente através da interface web do ONOS. A Figura 13 exibe um exemplo das saídas de *logs* da ferramenta, que utiliza a biblioteca *logging* para padronizar as mensagens..

```python
ATTACK_SCRIPTS = [
    {'module': 'macof_attack', 'params': {'interface': 'INTERFACE'}},
    {'module': 'hping_attack', 'params': {'target': 'IP', 'interface': 'INTERFACE'}},
    {'module': 'port_SYNflood', 'params': {'target': 'IP'}}
    # Adicione mais ataques conforme necessário
    # Padrao:
    # {'module': 'nome_Ataque', 'params': {'param1': 'REFValor', 'param2': 'REFValor2'}}
]

# Informações sobre o alvo
# Deve ter o mesmo nome do REFValor acima e segue o padrao:
# 'REFValor1': 'valorReal', ...
TARGET_INFO = {
    'IP': '10.0.0.1',
    'INTERFACE': 'h1-eth0'
}
```

Figura 11: Exemplo de arquivo de configuração utilizado nos testes. Fonte: O autor.

```
mininet> pingall
*** Ping: testing ping reachability
h1 → h2 h3 h4 h5 h6 h7 h8
h2 → h1 h3 h4 h5 h6 h7 h8
h3 → h1 h2 h4 h5 h6 h7 h8
h4 → h1 h2 h3 h5 h6 h7 h8
h5 → h1 h2 h3 h4 h6 h7 h8
h6 → h1 h2 h3 h4 h5 h7 h8
h7 → h1 h2 h3 h4 h5 h6 h8
h8 → h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
```

Figura 12: Execução do *pingall* antes da execução da ferramenta. Fonte: O autor.

```
524 2024-10-13 19:15:56,527 - attack_scripts.macof_attack - WARNING -
    Timeout de 30 segundos atingido para macof na interface h1-eth0
525 2024-10-13 19:15:56,630 - attack_scripts.macof_attack - INFO -
    Monitoramento tcpdump finalizado
526 2024-10-13 19:15:56,640 - __main__ - INFO - Ataque macof_attack
    concluído com sucesso em 30.57 segundos
527 2024-10-13 19:15:56,640 - config.result_logger - INFO - Resultado do
    macof_attack: MacofAttack executado na interface h1-eth0 - Status:
    Timeout (30s reached)
```

Figura 13: Exemplo de arquivo de *log* da ferramenta. Fonte: O autor.

4 Apresentação e Análise dos Resultados

Como apresentado na seção anterior, os testes foram executados com uso de ataques de inundação: *MAC Flooding, TCP Flooding, UDP Flooding e ICMP Flooding.* Atacando os *switches* e o controlador, diretamente e por meio de suas portas, visando desabilitar serviços e comunicação, abaixo é realizada uma análise dos resultados obtidos com a execução da ferramenta no ambiente de testes.

Durante o processo de testes, foram realizadas verificações contínuas da disponibilidade da rede, o ataque de *MAC flooding* conseguiu gerar instabilidade na rede durante seu processo de execução na ferramenta, vista a partir do *pingall*. A Figura 14 mostra a indisponibilidade gerada durante a execução da ferramenta. Quando um *host* não consegue se comunicar com outro, um 'X' é exibido na linha correspondente. Além disso, a interface web do ONOS, mostrada na Figura 15, permite observar a representação gráfica da indisponibilidade gerada no switch.

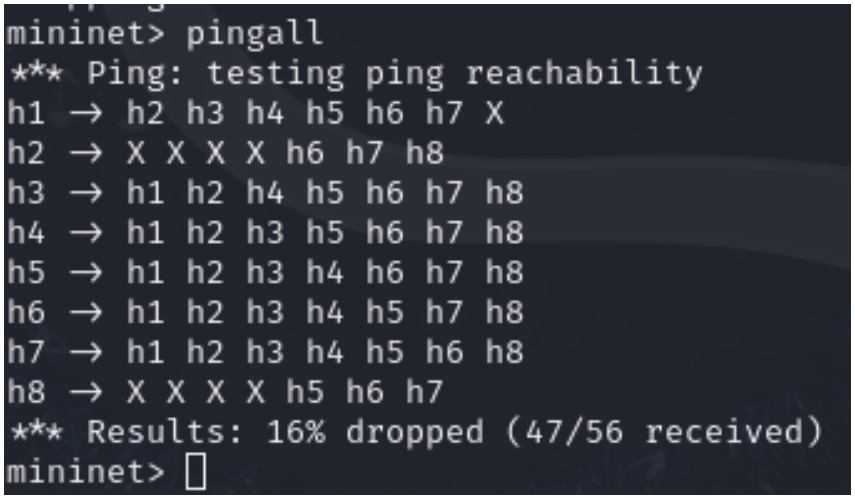

Figura 14: Indisponibilidade gerada na conexão entre *hosts* no Mininet. Fonte: O autor.

Além disso, os ataques *TCP/UDP/ICMP Flooding* foram aplicados aos *switches* e diretamente nas portas de serviço do controlador ONOS. No primeiro caso, o ataque foi executado diversas vezes enquanto o teste de conectividade entre *hosts* era executada no Mininet, constatando a indisponibilidade durante o processo de ataque. A Figura 16 mostra a indisponibilidade da rede durante a execução desse cenário, onde é possível observar erros de conexão entre os host.

Para o segundo cenário, foi testada também a utilização da inundação de *TCP SYN* diretamente no controlador nas portas 6653 e 8181, responsáveis, respectivamente, pelos serviços de comunicação do Openflow e pela interface gráfica do ONOS. Nos testes da porta 6653, notou-se que gradativamente foram acontecendo desconexões de *hosts* e travamentos na interface, como apresentado na Figura 17. Posteriormente, ocorreu a in-

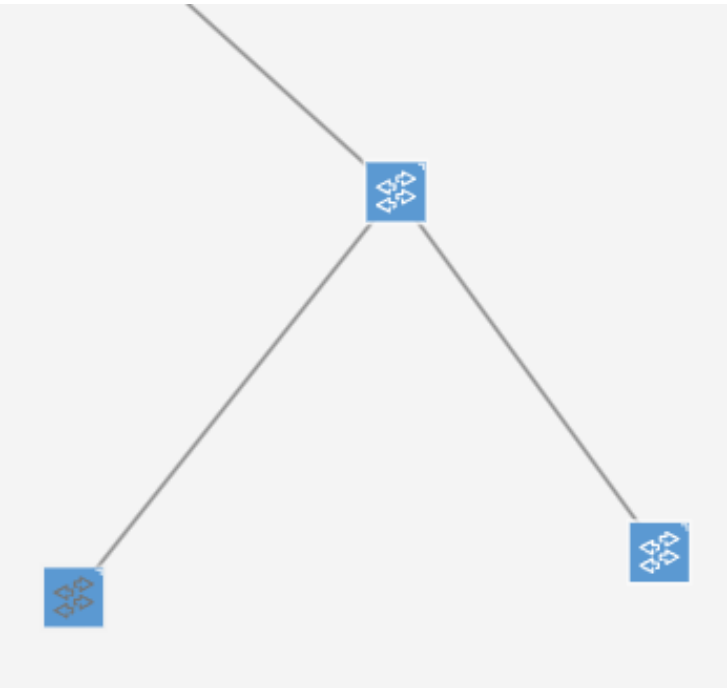

Figura 15: Indisponibilidade de switch representada na interface web do ONOS. Fonte: O autor.

```
mininet> pingall
*** Ping: testing ping reachability
h1 → X h3 h4 h5 h6 h7 h8
h2 → h1 h3 h4 X X X X
h3 → h1 h2 h4 h5 h6 h7 X
h4 → X X h3 h5 h6 h7 h8
h5 → h1 h2 h3 h4 h6 h7 h8
h6 → h1 h2 h3 h4 h5 h7 h8
h7 → h1 X X X h5 h6 h8
h8 → h1 h2 h3 h4 h5 h6 h7
*** Results: 19% dropped (45/56 received)
mininet>
```

Figura 16: Indisponibilidade gerada pelas inundações TCP/UDP/ICMP durante teste de conexão no Mininet. Fonte: O autor.

disponibilidade, apresentando uma mensagem de erro no *web socket* e encerramento da conexão com a interface, como na Figura 18. Além disso, foi observado que durante os testes na porta 8181, houve a indisponibilidade de múltiplos *hosts*, juntamente com o mal funcionamento das funções do menu lateral, fazendo as opções dele desaparecerem, tornando o menu inutilizável, como apresentado na Figura 19. Após isso, houve a indisponibilidade da interface, com mensagem idêntica a apresentada na Figura 18. Após algum tempo com a conexão perdida, a interface web foi reconectada, porém, ela apresentou mal funcionamento, com erros de carregamento de componentes e lentidão, como observado na Figura 20. Vale salientar que a indisponibilidade também afetou a rede emulada no

Mininet, que apresentou perda total de pacotes durante os testes.

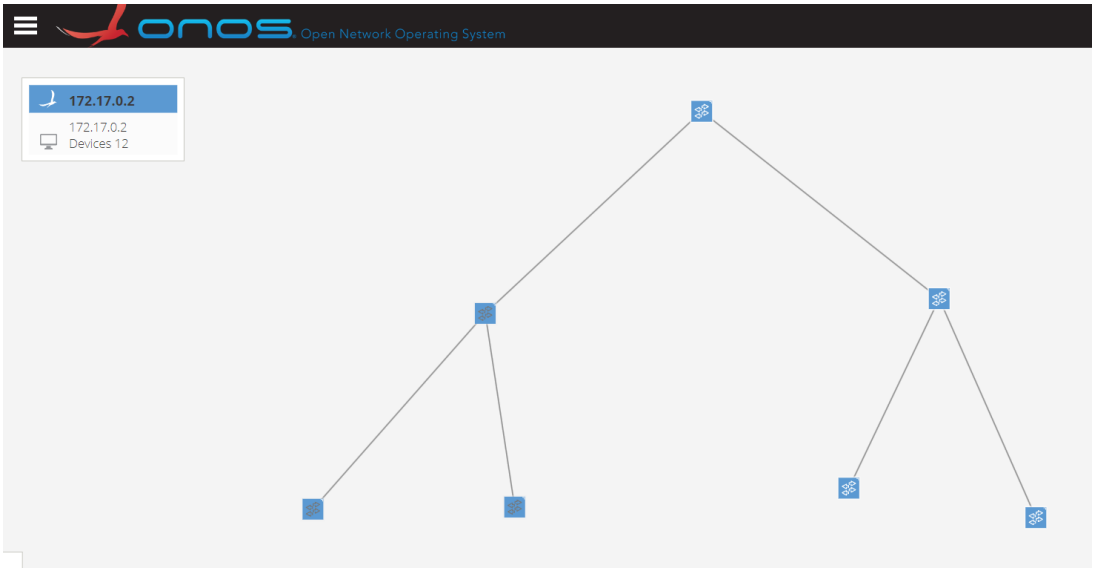

Figura 17: Múltiplos *hosts* desconectados com a inicialização do ataque. Fonte: O autor.

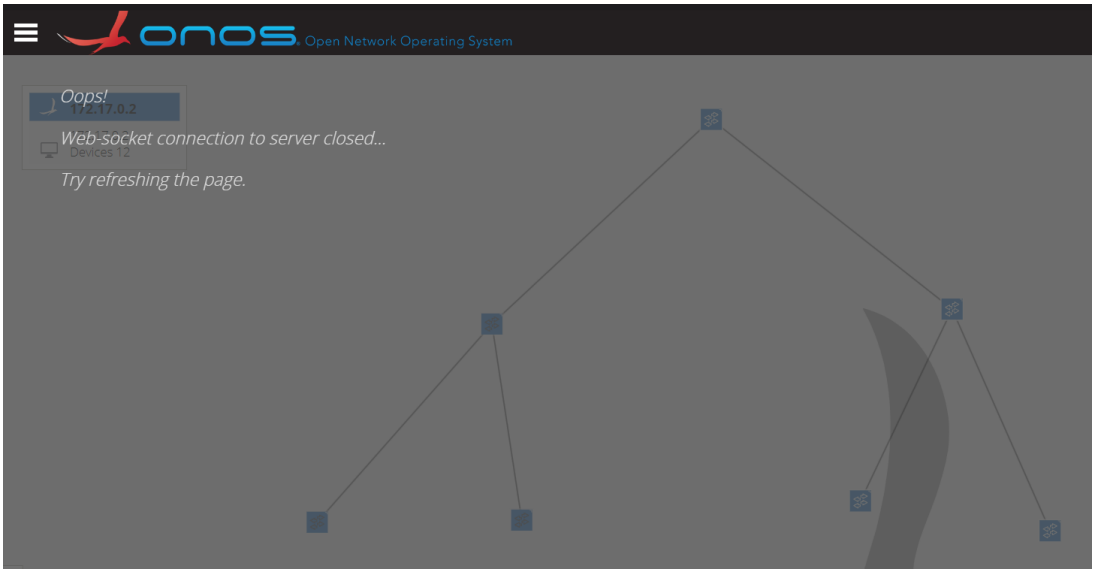

Figura 18: Mensagem de erro gerada com a perda da conexão com a interface web do ONOS. Fonte: O autor.

Com base no exposto, é possível observar como os ataques de negação de serviço possuem um impacto significativo nas redes SDN, mesmo que em um cenário emulado. Em uma análise mais aprofundada, os ataques de inundação foram efetivos para gerar a indisponibilidade da rede, em especial aqueles executados na porta de serviços do controlador, visto que a indisponibilidade chegava a ser de toda a topologia e inclusive do próprio controlador. Diferente por exemplo, de quando houve a execução da inundação de MAC contra o *switch*, que a indisponibilidade era local, em um cenário real, possivelmente haveria a presença de regras para proibir tráfegos de alta carga, porém, vale salientar que

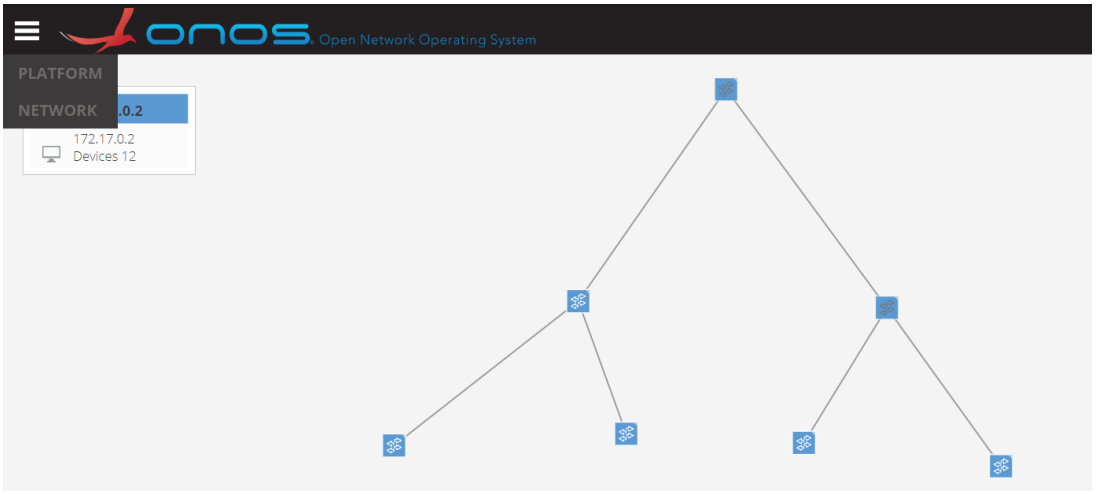

Figura 19: Quedas de múltiplos *hosts* e mal funcionamento da interface web do ONOS. Fonte: O autor.

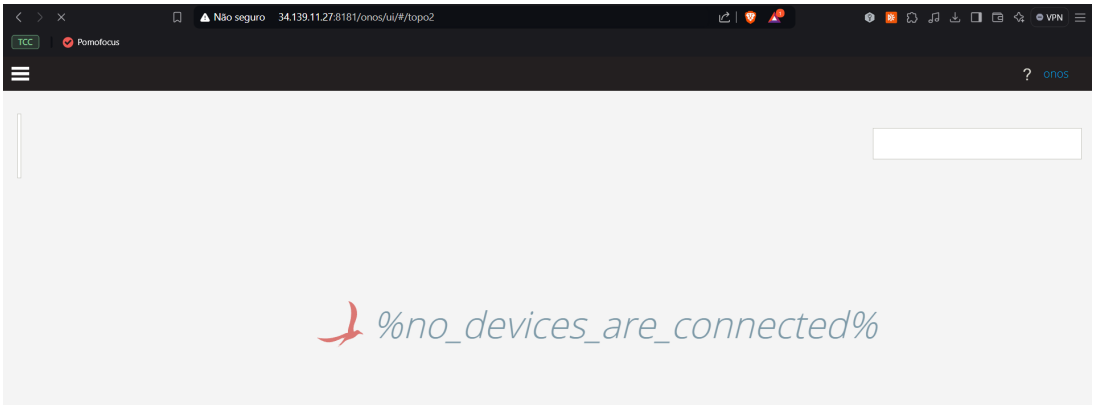

Figura 20: Interface web do ONOS apresentando erros e lentidões depois da indisponibilidade. Fonte: O autor.

os impactos poderiam ser ainda maiores com uso de múltiplos *hosts* maliciosos. Pode-se inclusive executar ataques como o *Slow-TCAM* ou *Slow-Saturation*, os quais poderiam não ser detectados pela defesa. Além disso, o ganho de tempo com o uso da ferramenta e a facilidade de execução dos testes foi um fator importante, visto que foram executados diversos cenários diferentes e em vários momentos, os quais foram configurados rapidamente pelo arquivo de configuração e com o repositório de ataques. Ao imaginar o cenário almejado de inclusão em ciclos de CI/CD, será de grande utilidade, visto que foi muito fácil de manipular e utilizar a ferramenta criada.

Com relação ao uso da ferramenta em si, foi feita uma comparação com execução manual e individual de cada script via terminal no host atacante e também a execução automatizada com a ferramenta. Constatou-se que conforme mais ataques são inseridos, maior o ganho de tempo obtido com o uso da ferramenta. Em testes realizados com 2 ou 3 scripts, o ganho de tempo foi em média de 50%, enquanto para testes com 5 scripts ou mais, foi obtido um ganho médio de 110%. Esses números são demonstrativos

da facilidade gerada pela ferramenta, ao imaginar cenários ainda maiores, esses ganhos podem aumentar consideravelmente. Sem contar que erros de digitação e de execução eram recorrentes no processo de execução manual.

5 Conclusões e Trabalhos Futuros

A ferramenta desenvolvida trouxe ganhos significativos em eficiência e economia de tempo nos testes de segurança em redes SDN. Sua automação permite a execução de múltiplos ataques de forma sequencial e consistente, sem a necessidade de intervenção manual para cada script, o que, em cenários de testes maiores, oferece ainda mais economia de tempo e minimiza erros humanos. Essa praticidade também facilita a integração da ferramenta em ciclos de CI/CD, permitindo que os testes de segurança sejam executados automaticamente a cada modificação na rede.

Nos experimentos realizados, os testes mostraram que redes SDN são vulneráveis a ataques de negação de serviço. Mesmo em um ambiente de testes, com um único host atacante, a ferramenta conseguiu demonstrar a viabilidade de gerar indisponibilidade no controlador e, consequentemente, na rede. Em um cenário real, um ataque distribuído poderia provocar danos ainda maiores. Portanto, a ferramenta não só simplifica e otimiza o processo de experimentação, mas também contribui para uma melhor compreensão das vulnerabilidades e para o fortalecimento das redes SDN frente a ameaças reais.

Vale salientar que no momento da escrita desse trabalho, estão sendo feitos testes no *testbed* do projeto OpenRAN@Brasil da RNP (RNP - Rede Nacional de Ensino e Pesquisa), o qual visa implantar um padrão aberto de acesso a rádio, tema importante para telecomunicações, em especial para as redes 5G. Além disso, foram realizadas duas publicações em *Workshops* de iniciação científica, uma no SBSeg [31] (SBSeg - Simpósio Brasileiro de Segurança da Informação e Sistemas Computacionais) e outro no SBrT [32] (SBrT - Simpósio Brasileiro de Telecomunicações e Processamento de Sinais), realizados nos meses de Setembro e Outubro de 2024, respectivamente.

Como trabalhos futuros, pretende-se adicionar novos testes de clientes, analisando dados que vão além da disponibilidade, obtendo assim novas métricas. A ideia é que também possam ser feitas melhorias de interface e funcionamento da ferramenta, adicionando novos ataques de intrusão e outros focados em SDN, também gerando melhorias nas funcionalidades atuais. Além disso, serão explorados novos cenários de testes, com outras topologias de rede, outros ataques e novas métricas de resultado, de forma a variar exemplos de ambientes de execução.

Referências

[1] OWASP Foundation, "OWASP Top 10," 2021. Acessado em: 08 de novembro de 2024.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[3] H. Pan, Z. Li, P. Zhang, K. Salamatian, and G. Xie, "Misconfiguration checking for sdn: Data structure, theory and algorithms," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pp. 1–11, 2020.

[4] D. Gopi, S. Cheng, and R. Huck, "Comparative analysis of SDN and conventional networks using routing protocols," in *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pp. 108–112, 2017.

[5] B. Sokappadu, A. Hardin, A. Mungur, and S. Armoogum, "Software defined networks: Issues and challenges," in *2019 Conference on Next Generation Computing Applications (NextComp)*, pp. 1–5, 2019.

[6] A. M. Alberti, D. G. Silva, F. A. P. de Figueiredo, F. de Assis Silva do Carmo, G. P. Aquino, and J. M. C. Brito, "Openran: a conexão do futuro," 2023. Instituto Nacional de Telecomunicações.

[7] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing SDN from openflow to P4: A survey," *ACM Comput. Surv.*, vol. 55, jan 2023.

[8] X. Wen, B. Yang, Y. Chen, L. E. Li, K. Bu, P. Zheng, Y. Yang, and C. Hu, "Ruletris: Minimizing rule update latency for TCAM-based SDN switches," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 179–188, 2016.

[9] S. Georgiev and K. Nikolova, "Implementation of an agile SDLC CI/CD pipeline for managing a SDN VXLAN-EVPN fabric," in *2023 31st National Conference with International Participation (TELECOM)*, pp. 1–4, 2023.

[10] K. Nsafoa-Yeboah, E. T. Tchao, B. Yeboah-Akowuah, B. Kommey, A. S. Agbemenu, E. Keelson, M. Monirujjaman Khan, and G. Ferrari, "Software-defined networks for optical networks using flexible orchestration: Advances, challenges, and opportunities," *J. Comput. Netw. Commun.*, vol. 2022, jan 2022.

[11] M. N. A. Sheikh, I.-S. Hwang, M. S. Raza, and M. S. Ab-Rahman, "A qualitative and comparative performance assessment of logically centralized SDN controllers via mininet emulator," *Computers*, vol. 13, no. 4, 2024.

[12] C. Prabha, A. Goel, and J. Singh, "A survey on SDN controller evolution: A brief review," in *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, pp. 569–575, 2022.

[13] L. Zhu, M. M. Karim, K. Sharif, C. Xu, F. Li, X. Du, and M. Guizani, "SDN controllers: A comprehensive analysis and performance evaluation study," *ACM Comput. Surv.*, vol. 53, Dec. 2020.

[14] S. Bhardwaj and S. N. Panda, "Performance evaluation using RYU SDN controller in software-defined networking environment," *Wireless Personal Communications*, vol. 122, pp. 701–723, Jan 2022.

[15] S. Yang, L. Cui, Z. Chen, and W. Xiao, "An efficient approach to robust SDN controller placement for security," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1669–1682, 2020.

[16] D. Espinel Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 256–281, 2021.

[17] L. V. Ruchel, R. C. Turchetti, and E. T. de Camargo, "Evaluation of the robustness of SDN controllers ONOS and ODL," *Computer Networks*, vol. 219, p. 109403, 2022.

[18] L. Mamushiane and T. Shozi, "A QoS-based evaluation of SDN controllers: ONOS and opendaylight," in *2021 IST-Africa Conference (IST-Africa)*, pp. 1–10, 2021.

[19] O. Romanov, N. Korniienko, I. Obod, and I. Svyd, "Construction of the SDN control level based on ONOS," in *2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo)*, pp. 127–132, 2021.

[20] S. Ahmad and A. H. Mir, "Scalability, consistency, reliability and security in SDN controllers: A survey of diverse SDN controllers," *Journal of Network and Systems Management*, vol. 29, no. 9, 2021.

[21] F. Albuquerque, T. Pascoal, V. Nigam, and I. E. Fonseca, "Usando o particionamento de switch para mitigar o problema de acoplamento," in *XL Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2022)*, 2022.

[22] X. Qiu and Z. Tang, "Research advanced in the security defence of software defined network," in *2022 International Conference on Electronics and Devices, Computational Science (ICEDCS)*, pp. 380–384, 2022.

[23] V. D. M. Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, "Detection and mitigation of low-rate denial-of-service attacks: A survey," *IEEE Access*, vol. 10, pp. 76648–76668, 2022.

[24] T. A. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow TCAM exhaustion DDoS attack," in *IFIP International Conference on ICT Systems Security and Privacy Protection*, pp. 17–31, Springer, 2017.

[25] T. A. Pascoal, I. E. Fonseca, and V. Nigam, "Slow denial-of-service attacks on software defined networks," *Comput. Networks*, vol. 173, p. 107223, 2020.

[26] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Pérez-Díaz, and M. Zareei, "Towards security automation in software defined networks," *Computer Communications*, vol. 183, pp. 64–82, 2022.

[27] A. Alshamrani, A. Chowdhary, S. Pisharody, D. Lu, and D. Huang, "A defense system for defeating DDoS attacks in SDN based networks," in *Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '17, (New York, NY, USA), p. 83–92, Association for Computing Machinery, 2017.

[28] C. Liu, A. Raghuramu, C.-N. Chuah, and B. Krishnamurthy, "Piggybacking network functions on SDN reactive routing: A feasibility study," in *Proceedings of the Symposium on SDN Research*, SOSR '17, (New York, NY, USA), p. 34–40, Association for Computing Machinery, 2017.

[29] M. F. Hyder and M. A. Ismail, "Securing control and data planes from reconnaissance attacks using distributed shadow controllers, reactive and proactive approaches," *IEEE Access*, vol. 9, pp. 21881–21894, 2021.

[30] Y. Zhou, G. Cheng, and S. Yu, "An SDN-enabled proactive defense framework for DDoS mitigation in IoT networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5366–5380, 2021.

[31] R. M. S. Leal, J. K. E. Freitas, F. A. C. A. Júnior, W. T. A. Lopes, F. B. S. Carvalho, and I. E. Fonseca, "Automação e cibersegurança: Criando uma ferramenta de testes para redes SDN," in *Anais Estendidos do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2024)*, (São José dos Campos, SP, Brasil), pp. 209–218, SBC, September 2024. Publicado em 16 de setembro de 2024.

[32] R. M. S. Leal, J. K. E. Freitas, F. A. C. A. Júnior, W. T. A. Lopes, F. B. S. Carvalho, and I. E. Fonseca, "Uma ferramenta para automação de testes de cibersegurança em

redes SDN," in *Anais do XLII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2024)*, (Belém, PA, Brasil), October 2024. Publicado em 01 de outubro de 2024.