DISSERTAÇÃO DE MESTRADO PROFISSIONAL

PROPOSTA DE INTEGRAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO (IDS) ENTRE UMA REDE SDN E UMA *HONEYNET*

AWATEF ALI YOUSEF R. FARES

Brasília, maio de 2021

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

PROPOSAL FOR THE INTEGRATION OF AN INTRUSION DETECTION SYSTEM (IDS) BETWEEN AN SDN NETWORK AND A HONEYNET

PROPOSTA DE INTEGRAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO (IDS) ENTRE UMA REDE SDN E UMA *HONEYNET*

AWATEF ALI YOUSEF R. FARES

ORIENTADOR : DR. GEORGES DANIEL AMVAME NZE.

DISSERTAÇÃO DE MESTRADO PROFISSIONAL EM
ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPEE.MP.010

BRASÍLIA/DF: MAIO - 2021

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

PROPOSTA DE INTEGRAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO (IDS) ENTRE UMA REDE SDN E UMA *HONEYNET*

AWATEF ALI YOUSEF R. FARES

Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Georges Daniel Amvame Nze, Ph.D, FT/UnB _____
Orientador

Prof. Fábio Lúcio Lopes de Mendonça, Ph.D, FT/UnB _____
Examinador Interno

Prof. Honório Assis Filho Crispim, Ph.D, UniCEUB _____
Examinador Externo

Prof. Vinícius Pereira Gonçalves, Ph.D, FT/UnB _____
Examinador Suplente

FICHA CATALOGRÁFICA

FARES R.Y., AWATEF ALI
PROPOSTA DE INTEGRAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO (IDS) ENTRE UMA REDE SDN E UMA *HONEYNET* [Distrito Federal] 2021.
xvi, 67 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2021).
Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Redes definidas por software	2. Sistema de detecção de intrusão (IDS)
3. Controlador SDN	4. Honeynet
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

FARES R.Y., AWATEF ALI. (2021). *PROPOSTA DE INTEGRAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO (IDS) ENTRE UMA REDE SDN E UMA HONEYNET*.
Dissertação de Mestrado Profissional, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 67 p.

CESSÃO DE DIREITOS

AUTOR: AWATEF ALI YOUSEF R. FARES
TÍTULO: PROPOSTA DE INTEGRAÇÃO DE UM SISTEMA DE DETECÇÃO DE INTRUSÃO (IDS) ENTRE UMA REDE SDN E UMA *HONEYNET*.
GRAU: Mestre em Engenharia Elétrica ANO: 2021

AWATEF ALI YOUSEF R. FARES
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA

Dedico este trabalho à minha família. À minha mãe, Marlice, aos meus filhos, Júlia e Miguel, e ao meu marido, Francisco, que me apoiou nesta jornada e me ajudou a chegar mais longe.

AGRADECIMENTOS

Agradeço ao meu marido, Francisco, que me apoiou em todo este percurso incondicionalmente.

Agradeço à minha mãe, Marlice, que sempre me incentivou aos estudos e me apoiou cuidando dos meus filhos para que pudesse ir às aulas presenciais.

Agradeço ao meu orientador, Professor Georges Daniel Amvame Nze, por ter aceitado me orientar neste caminho, doando o seu tempo e conhecimento para que este trabalho pudesse ser concluído.

Agradeço a todos que, de alguma maneira, fizeram parte desta jornada.

Agradeço ao apoio da Pesquisa Aplicada de Integração Tecnológica e Interoperação Sistêmica na Defensoria Pública da União - DPGU (TED 66/2016) e ao Conselho Administrativo de Defesa Econômica - CADE (TED 08700.000047/2019-14), bem como ao suporte do Laboratório LATITUDE/UnB (Projeto SDN 23106. 099441/2016-43).

RESUMO

Título: Proposta de integração de um sistema de detecção de intrusão (IDS) entre uma rede SDN e uma honeynet.
Autor: Awatef Ali Yousef Rodrigues Fares.
Orientador: Prof. Georges Daniel Amvame Nze, Ph.D, FT/UnB
Programa de Pós-Graduação Profissional em Engenharia Elétrica – Área de Concentração em Segurança Cibernética
Brasília, maio de 2021

A arquitetura das redes definidas por *software* vem sendo cada vez mais utilizada no cenário atual das redes de comunicações, pois traz diversos benefícios em relação às redes convencionais por possuírem em sua estrutura um controle central sobre a rede. Dentre esses benefícios, estão a sua flexibilidade e facilidade de programação. Apesar das vantagens, a rede SDN, assim como as redes tradicionais, possui a necessidade de implementação de recursos de segurança, principalmente devido ao seu gerenciamento centralizado. Neste trabalho de pesquisa, é proposta uma solução de segurança para mitigação de ataques de negação de serviço utilizando a implementação de um sistema de detecção de intrusões (IDS) associado a um *firewall* em uma arquitetura de rede *OpenFlow*. Para realizar o estudo e análise do comportamento do tráfego malicioso, foi colocada em prática a utilização de um ambiente *honeynet* cujo modelo foi elaborado no laboratório da UNB. Sob o comando da controladora SDN, os pacotes classificados pelo IDS como originados de um ataque malicioso são encaminhados para esse ambiente, onde é possível restringir acessos não desejados em caso de comprometimento da rede e realizar análise dos ataques para possíveis medidas de contrapartida. Na solução proposta, é possível reagir a ataques realizando o bloqueio do fluxo de dados mais próximo da origem, impedindo que o funcionamento da rede seja prejudicado. Este trabalho implementou uma solução *open source* para segurança contra ataques DoS, e os resultados obtidos mostram como a arquitetura proposta pode auxiliar na identificação, análise, no estudo dos parâmetros e na tomada de decisão nas ações necessárias para detectar um ataque cibernético.

Palavras-chave - Redes definidas por *software* (SDN); Sistema de Detecção de Intrusão (IDS); controlador SDN; *honeynet*.

ABSTRACT

Title: Proposal for the integration of an intrusion detection system (IDS) between an SDN network and a honeynet.
Author: Awatef Ali Yousef Rodrigues Fares.
Advisor: Prof. Georges Daniel Amvame Nze, Ph.D, FT / UnB
Professional Graduate Program in Electrical Engineering - Cybersecurity Concentration Area
Brasilia, may, 2021

The architecture of networks defined by software has been increasingly used in the current scenario of communications networks, as it brings several benefits in relation to conventional networks because they have a central control over the network in their structure. Among these benefits are its flexibility and ease of programming. Despite the advantages, the SDN network, like traditional networks, has the need to implement security resources, mainly due to its centralized management. In this research work, a security solution is proposed to mitigate denial of service attacks using the implementation of an intrusion detection system (IDS) associated with a firewall in an OpenFlow network architecture. To carry out the study and analysis of the behavior of malicious traffic, the use of a honeynet environment was put into practice, a model that was developed in the UNB laboratory. Under the command of the SDN controller, packets classified by IDS as originating from a malicious attack are forwarded to this environment, where it is possible to restrict unwanted access in the event of network compromise and carry out analysis of attacks for possible counterpart measures. In the proposed solution, it is possible to react to attacks by blocking the data flow closest to the source, preventing the functioning of the network from being impaired. This work implemented an open source solution for security against DoS attacks, and the results obtained show how the proposed architecture can assist in identification, analysis, in the study of parameters and in the decision making in the necessary actions to detect a cyber attack.

Keywords - Software defined networks (SDN); OpenFlow; Intrusion Detection System (IDS); SDN controller; honeynet.

SUMÁRIO

1 INTRODUÇÃO 1
1.1 OBJETIVOS 3
1.1.1 OBJETIVO GERAL 3
1.1.2 OBJETIVOS ESPECÍFICOS 4
1.2 MOTIVAÇÃO E JUSTIFICATIVA 4
1.3 CONTRIBUIÇÕES DO TRABALHO 4
1.4 ESTRUTURA DA DISSERTAÇÃO 5

2 REVISÃO BIBLIOGRÁFICA E TRABALHOS CORRELATOS 6
2.1 REDES DEFINIDAS POR *software* 6
2.2 PROTOCOLO OPENFLOW 7
2.3 PLANO DE CONTROLE 8
2.4 CONTROLADORES SDN 9
2.4.1 NOX 10
2.4.2 POX 11
2.4.3 RYU 11
2.4.4 FLOODLIGHT 12
2.4.5 OPENDAYLIGHT 12
2.5 SEGURANÇA 13
2.5.1 SISTEMA DE DETECÇÃO DE INTRUSÃO 13
2.5.2 SISTEMA DE PREVENÇÃO DE INTRUSÃO 14
2.5.3 *Firewall* 16
2.5.4 FALHAS NA SEGURANÇA DE DADOS 16
2.5.5 ATAQUES DE NEGAÇÃO DE SERVIÇO DDOS E DOS 17
2.6 VIRTUALIZAÇÃO 18
2.6.1 REDES LOCAIS VIRTUAIS 19
2.6.2 REDES PRIVADAS VIRTUAIS 19
2.6.3 REDES *Overlay* 20
2.6.4 EMULADORES E SIMULADORES DE REDE 21
2.6.5 GNS3 21
2.6.6 VMWARE 21
2.6.7 MININET 22
2.6.8 API 22
2.6.9 PLANO DE GERENCIAMENTO 23
2.6.10 *Northbound Interface* 23
2.6.11 *Southbound Interface* 24
2.6.12 CDN 25

2.6.13 *Honeynet* 26
2.6.14 DMZ- Zona desmilitarizada 26
2.7 Trabalhos correlatos 26

3 Proposta de arquitetura 32
3.1 Arquitetura Geral 32
3.2 Escolha dos Dispositivos da Topologia 34
3.2.1 *Switch* SDN 34
3.2.2 Controlador SDN 35
3.2.3 *Honeynet* 44
3.2.4 Integração *Firewall* / Snort 44
3.2.5 Integração Controlador Ryu / Snort 46
3.2.6 Ambiente de simulação 47
3.3 Fluxograma de funcionamento do programa para integração . 48
3.3.1 *Honeynet* utilizada na solução 49
3.3.2 Mininet e Open Virtual Switch 49
3.3.3 Comunicação entre Open Virtual Switch e controladora 49

4 Resultados 52
4.1 Arquitetura utilizada 52
4.1.1 *Switch Virtual OVS* 53
4.1.2 *Firewall* / IPS 54
4.1.3 Identificação de ataque de negação de serviço 54
4.1.4 PC Atacante 54
4.1.5 Servidor Web 55
4.1.6 Resultados 56

5 Conclusão e trabalhos futuros 60
5.1 Conclusão e Trabalhos futuros 60

REFERÊNCIAS BIBLIOGRÁFICAS 61

LISTA DE FIGURAS

2.1 Comparação da arquitetura SDN versus arquitetura tradicional [1] 6
2.2 Protocolo OpenFlow adaptado de [2] 8
2.3 Integração das camadas de aplicação, controle e infraestrutura. 9
2.4 Controlador NOX [3] 10
2.5 Arquitetura de funcionamento do Controlador RYU 12
2.6 Fluxo do funcionamento do IDS [4] 14
2.7 Diferença entre o funcionamento do IDS e do IPS. Adaptado de [4] 15
2.8 Firewall 16
2.9 Fluxo de um ataque DoS 17
2.10 Plataforma de aplicações virtualizadas [4] 18
2.11 Comunicação entre VLANS 19
2.12 Topologia de um túnel VPN 20
2.13 Redes *Overlay* – Sobrepostas 21
2.14 Disposição da API inserida na arquitetura SDN. 25

3.1 Topologia da solução proposta. 34
3.2 Associação do Ryu ao OVS. 35
3.3 Execução do controlador OpenDaylight 36
3.4 Exibição do OpenDaylight no navegador porta 8080 37
3.5 VM Floodlight 37
3.6 VM Floodlight com Mininet em execução 38
3.7 Módulos pré-requisitos de instalação do controlador Ryu 38
3.8 Latência do OpenDaylight 39
3.9 Métrica de Jitter e perda de pacote do OpenDaylight 40
3.10 Latência do Floodlight 41
3.11 Métrica de Jitter e perda de pacote do Floodlight 41
3.12 Latência do Ryu 42
3.13 Métrica de Jitter e perda de pacote do Ryu 43
3.14 Instalação do fwsnort. 45
3.15 Integração entre *firewall* e IDS utilizando fwsnort. 46
3.16 Snort em execução no controlador Ryu 47
3.17 Fluxograma 48
3.18 Comunicação OpenFlow. 50

4.1 Arquitetura desenvolvida no GNS3 52
4.2 Associação do OVS ao controlador 53
4.3 Ataque iniciado a partir da VM *host* atacante 55
4.4 Arquitetura desenvolvida no GNS3 para prova de conceito 55

4.5 Convergência de segurança da rede SDN..... 56
4.6 Resultado de 100 testes realizados, medindo os fatores que compõem e tempo de convergência..... 57
4.7 Gráfico BoxPlot das variáveis X e Y 59

LISTA DE TABELAS

3.1 Resumo dos testes realizados por meio de ferramentas para medir parâmetros de utilização de rede. 43

4.1 Resultado dos testes..... 57
4.2 Resumo dos resultados. 58

LISTA DE SIGLAS E ABREVIAÇÕES

SDN	*Software-Defined Networking*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
IDS	*Intrusion Detection System*
DoS	*Denial of Service*
DDoS	*Distributed Denial of Service*
API	*Application Programming Interface*
GUI	*Graphical User Interface*
HTTP	*Hypertext Transfer Protocol*
IPS	*Intrusion Prevention System*
ABNT	*Associação brasileira de normas técnicas*
NETCONF	*Network Configuration Protocol*
RTT	*Roud Trip Time*
VPN	*Virtual Private Network*
VLAN	*Virtual Local Area Network*
OVSDB	*Open vSwitch Database*
BGP	*Border Gateway Protocol*
SNMP	*Simple Network Management Protocol*
LAN	*Local Area Network*
WAN	*Wide Area Network*
DMZ	*Demilitarized Zone*
REST	*Representational State Transfer*
DNS	*Domain Name System*
UDP	*User Datagram Protocol*
ICMP	*Internet Control Message Protocol*
OSI	*Open System Interconnection*
OVS	*Open Virtual Switch*
VM	*Virtual Machine*
PC	*Personal Computer*
NIST	*National Institute of Standards and Technology*
GNS3	*Graphical Network Simulator-3*
ONOS	*Open Network Operation System*
CPU	*Central Process Unit*
CDN	*Content Delivery Network*
IoT	*Internet of Things*
RTHB	*Remotely triggered black hole*
ISP	*Internet Service Provider*

1 INTRODUÇÃO

Pode-se afirmar que a Internet se tornou indispensável ao meio de vida da população mundial, pois utilizamos as redes de dados para diversos tipos de necessidades, como troca de informações em redes sociais, transações bancárias, atividades médicas, entre tantas outras funcionalidades. De fato, a comunicação através de uma rede de dados tornou-se essencial, proporcionando mecanismos para estabelecer uma comunicação entre diversos segmentos da sociedade e em vários locais do nosso hemisfério. Nesse contexto, é correto afirmar que as redes tradicionais não acompanharam a necessidade desse desenvolvimento, mantendo-se limitadas em hardwares e softwares proprietários de difícil gerenciamento quando utilizados em grande escala [5]. As redes convencionais utilizam protocolos de comunicação que surgiram em conjunto com a primeira rede de dados mundial, a Internet. Um exemplo desses protocolos é o TCP. Embora, de modo geral, este e outros protocolos ainda atendam às necessidades de funcionamento da estrutura das redes de comunicação, cada vez mais surgem casos em que os padrões existentes mostram uma falta de flexibilidade, de controle ou de padronização. Os protocolos de redes, em sua maioria, não acompanharam a evolução das redes de comunicação. Com o crescente volume de tráfego de dados entre as redes de computadores, pontuou-se a grande necessidade de torná-las cada vez mais escaláveis. Com isso, é de extrema importância a introdução de tecnologias que possam suprir a enorme infraestrutura que possuem atualmente e, principalmente, o seu gerenciamento [6].

Para que seja possível o atendimento da crescente demanda de tráfego nos diversos segmentos existentes no mercado, como de pesquisa, bancários ou até mesmo em atividades de suma importância à vida, como laboratoriais e médicas, as empresas vêm buscando soluções que possam aumentar a utilização através de um melhor aproveitamento de recursos, dispondo de um gerenciamento centralizado que proporcione a visão do todo da estrutura implementada, maior agilidade nas ações a serem tomadas, controlando facilmente os ativos existentes na rede. A necessidade de melhores formas de aproveitamento dos recursos computacionais traz consigo um conceito de arquitetura de rede de dados chamado redes definidas por *software (Software-Defined Networking-SDN)*. Esse paradigma sugere o desacoplamento do plano de dados, que é responsável pelo encaminhamento de pacotes, e do plano de controle, o qual possui a inteligência e visão global da rede, estando logicamente centralizado em uma entidade externa denominada de controlador de rede [7].

O conceito de SDN insere um modelo de arquitetura que altera a forma de gerenciar, administrar, controlar e programar redes de comunicação, reduzindo as limitações constantes das redes convencionais onde é proposto separar o plano de controle, relacionado ao controle do tráfego de rede, do plano de dados, responsável por encaminhar esse tráfego em acordo com o plano de controle [8].

O protocolo mais utilizado em redes SDN é o *OpenFlow*, um recurso *open source* que possui a

capacidade de configurar tabelas e traz o benefício de gerenciar decisões de fluxo atuando em conjunto com o *software* controlador central que possui o domínio de todos os recursos da rede [9].

Os benefícios que a SDN traz em relação às redes tradicionais fez com que essa tecnologia fosse mais utilizada no decorrer dos últimos anos. Sua facilidade de instalação, flexibilidade, programabilidade e escalabilidade de forma rápida compõem um conjunto de itens que favorece a sua implementação. Ao contrário das redes SDN, as redes tradicionais são baseadas em uma arquitetura construída em camadas utilizando como padrão o modelo TCP/IP em uma comunicação ponto a ponto [10]. A comunicação mundial vem crescendo rapidamente, ampliando a necessidade de tecnologias que possam acompanhar esse crescimento com a utilização de recursos mais flexíveis e adaptáveis à nova realidade de consumo. Com a tecnologia SDN, toda a inteligência é executada em um ponto central da rede, o que facilita o controle e a adequação de novos serviços; o seu gerenciamento é realizado por meio de um *software controller* [11], que gerencia regras de encaminhamento aos seus dispositivos; nesta modalidade de arquitetura, os *switches* apenas realizam as operações de encaminhamento de pacotes, enquanto nas redes tradicionais o *switch* de camada 3 possui papel de roteador, interligando vários segmentos de rede e podendo ser configurado com diferentes redes virtuais (VLANs).

O comportamento das redes tradicionais possui uma estrutura diferente das redes SDN, sendo necessário realizar o gerenciamento e sua administração em praticamente todos os seus dispositivos, mas, assim como a SDN, possuem a necessidade de implementação de ferramentas de segurança. No modelo tradicional de redes de computadores, a administração dos equipamentos que as compõem é realizada de maneira individual, dificultando o trabalho do administrador e tornando mais complicada a sua análise para a resolução de um problema ou para melhoria na configuração de uma política de segurança [12]. A associação de segurança a uma rede SDN já constituída desempenha uma melhor utilização de seus recursos de forma integrada.

No entanto, o fato de a SDN possuir uma administração centralizada pode gerar o entendimento de que haverá um aumento dos riscos à segurança de sua estrutura, pois, caso ocorra um evento malicioso em seu ponto central, toda a rede poderá ficar fragilizada. A criação de uma instância de segurança integrando redes tradicionais às redes SDN possibilita a prevenção de ataques de negação de serviço e uma possível intervenção contra esses ataques caso seja necessário. Essa modalidade de ataque tem por objetivo tornar um servidor ou uma infraestrutura indisponível, gerando fluxos de dados muito além do que a infraestrutura pode tratar. Esses ataques partem de dispositivos previamente infectados e são denominados *bots*. Uma botnet é uma rede de hosts infectados coordenada por um bot master para gerar diferentes tipos de ataque [13].

> Os *bots* compõem as botnets [14], que são redes formadas por máquinas infectadas por alguma forma de *malware*. As botnets apresentaram elevado crescimento no início do século XXI, tornando-se uma das ameaças mais desafiadoras no campo de

defesa cibernética por coordenarem os ataques para o destino escolhido pelo atacante [15].

Com o objetivo de realizar estudos sobre as possíveis vulnerabilidades das redes de comunicações, surgiram as *honeynets*, estruturas que fornecem recursos reais de redes e fazem com que seja possível observar e analisar o comportamento de um ataque de negação de serviço. As *honeynets* reais fornecem sistemas operacionais que são utilizados em equipamentos com que o invasor pode interagir, não conseguindo distinguir que aquela estrutura foi elaborada apenas para o estudo de seu comportamento e sem perceber que não se trata da rede de fato.

Este trabalho de pesquisa objetiva demonstrar uma entre as várias possibilidades existentes de identificar e tratar ataques DoS com o intuito de minimizar os impactos causados em uma estrutura de rede SDN, utilizando uma ferramenta *(Network Intrusion Detection Systems – IDS)*, recurso de segurança tradicionalmente implementado em redes convencionais, associada ao ambiente *honeynet* para estudos dos ataques a fim de aprender o comportamento do invasor, qual a finalidade do ataque, entre inúmeras outras oportunidades de aprendizado [16]. A rede SDN tornou-se tendência no segmento tecnológico de comunicação, promovendo muitas possibilidades de exploração em termos de pesquisa e desenvolvimento. Este trabalho de dissertação aborda o desafio da segurança por meio do desenvolvimento e da avaliação de uma proposta que engloba a implementação do IDS que pode atuar de forma eficiente na realização da segurança de redes, identificando e analisando os pacotes suspeitos e encaminhando-os para a controladora, que decidirá como deve ser realizada a tratativa para esse tráfego indesejado. O IDS é associado ao *firewall*, ao controlador e à ferramenta *honeynet* para atuar na segurança e análise de dados contra uma possível inviabilidade de todo o sistema por sobrecarga de pacotes. Os testes realizados num ambiente simulado têm por objetivo detectar e analisar o comportamento do ataque DoS após ser identificado e direcionado por meio do Sistema de Detecção de Intrusão associado e da controladora.

1.1 OBJETIVOS

1.1.1 Objetivo geral

No contexto exposto, este trabalho de pesquisa possui por objetivo implementar uma solução de segurança para detecção e mitigação de ataques de negação de serviço em uma rede SDN associada a uma rede convencional, por meio da implementação de ferramenta IDS integrada a outros recursos de segurança como o *firewall* e um ambiente *honeynet* de modo a reduzir os impactos causados pelo ataque DoS do tipo volumétrico e realizar o estudo de seu comportamento.

1.1.2 Objetivos específicos

- Implementar arquitetura de uma SDN;
- Identificar padrão de assinatura de tráfego;
- Analisar a solução de segurança de mitigação de ataque de negação de serviço pela controladora;
- Avaliar o comportamento do ataque no ambiente *honeynet*.

1.2 MOTIVAÇÃO E JUSTIFICATIVA

A segurança contra ataques de negação de serviço (DoS) em redes definidas por *software* é abordada neste trabalho de pesquisa mediante a utilização de recursos implementados nas redes tradicionais e que comprovam por meio de testes e pesquisas que também podem ser utilizados com eficiência nas redes SDN, evitando que uma máquina ou um recurso de rede se torne indisponível a seus usuários, procurando comprovar que é possível gerar segurança nessa modalidade de arquitetura.

Esta proposta implementa um método de proteção contra o ataque de negação de serviço no plano de dados em que o controlador define qual a forma de tratar o fluxo anômalo detectado pelo IDS em um ambiente simulado, comprovando que há segurança em redes virtuais. O fato de o controlador possuir uma visão geral da arquitetura pode ser utilizado para uma melhor prevenção, identificação de tratativa dessa modalidade de ataque, associando medidas apropriadas para solução de forma automática sem a necessidade da intervenção do administrador. Mediante o desenvolvimento deste trabalho, podemos assegurar a integridade, escalabilidade e total segurança contra ataques DoS, evitando uma carga de dados não suportada pela rede SDN, tornando-a segura e estável.

1.3 CONTRIBUIÇÕES DO TRABALHO

O presente trabalho trouxe por contribuição a publicação de um artigo completo, *DoS Attack Prevention on IPS SDN Networks* [17], para o WCNPS 2019 (*Workshop on Communication Networks and Power Systems*), que gerou o desenvolvimento deste projeto de pesquisa e os itens seguintes:

- Apresentação de uma proposta de implementação de segurança contra ataques de negação de serviço associada a um IDS em um ambiente simulado;
- Desenvolvimento da solução de segurança e estudo do comportamento do ataque DoS após sua mitigação e desvio para a *honeynet*;

- Estudo e análise do comportamento do tráfego malicioso dentro do ambiente *honeynet*;
- Implementação funcional e análise dos resultados associados à controladora SDN.

1.4 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está organizado em cinco capítulos, sendo o primeiro o de introdução.

Os demais capítulos estão dispostos conforme descrito a seguir:

O Capítulo 2 oferece uma revisão atual das principais tecnologias utilizadas, contém uma revisão bibliográfica sobre segurança na arquitetura de redes SDN e esboça trabalhos relacionados ao assunto.

O Capítulo 3 apresenta a arquitetura proposta, a configuração do dispositivo IDS, o controlador, a *honeynet* e a integração dessas tecnologias no ambiente de simulação, os ataques gerados para avaliação.

O capítulo 4 apresenta os resultados obtidos por meio da técnica de segurança aplicada na mitigação do tráfego anômalo dos resultados práticos.

Por fim, o Capítulo 5 conclui o trabalho, apresentando os resultados alcançados após a conclusão deste estudo, assim como sugestões de estudos futuros.

2 REVISÃO BIBLIOGRÁFICA E TRABALHOS CORRELATOS

Este capítulo contém a revisão dos principais conceitos abordados nesta dissertação, como redes definidas por *software* – SDN, OpenFlow, sistemas de detecção de intrusão, ataques de negação de serviço (DoS – *Denial of Service*), controladores SDN, APIs de integração e virtualização e *honeynet*.

2.1 REDES DEFINIDAS POR *SOFTWARE*

As redes definidas por *software* possuem uma abordagem que utiliza um controlador centralizado atuando na separação do plano de controle e do plano de encaminhamento de pacotes, sendo uma rede programada de uma maneira inteligente, na qual, por meio desse *software*, é possível controlar suas variadas funcionalidades, tais como a forma de as solicitações externas interagirem com a rede SDN por intermédio de APIs OpenFlow. A SDN é uma arquitetura dinâmica, gerenciável, adaptável que acarreta redução nos custos operacionais e reduz o tempo de programação quando há necessidade de alterações em sua estrutura lógica. Suas principais vantagens são fornecer flexibilidade e ter uma alta programabilidade ao controle da rede por meio da separação do plano de controle e de seu *hardware*. O operador pode definir de maneira simples os fluxos e as ações sobre os fluxos por meio de uma interface de programação de aplicação [18].

A figura 2.1 ilustra a diferença entre a estrutura de uma rede convencional e a de uma rede SDN, na qual o plano de dados é separado do plano de controle; já nas redes tradicionais os dois planos funcionam em conjunto, não havendo uma diferenciação evidente de suas funcionalidades.

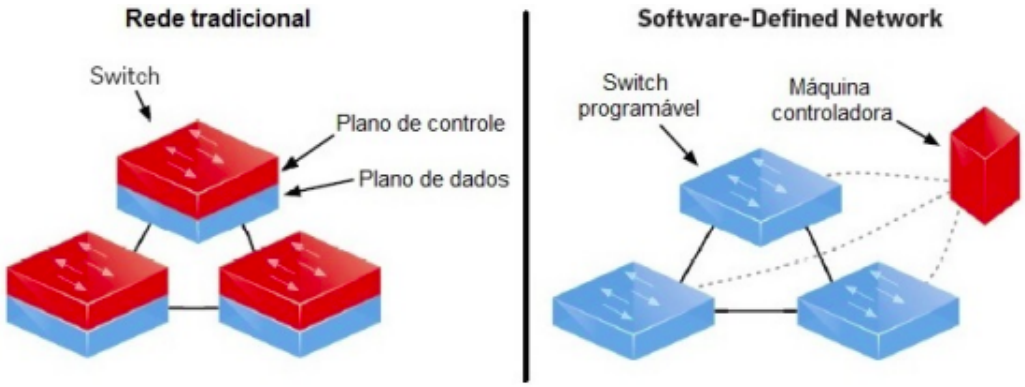

Figura 2.1: Comparação da arquitetura SDN versus arquitetura tradicional [1]

.

A rede SDN [19] possui o conceito semelhante ao das redes tradicionais, que regeneram o sinal

transmitido ao próximo nó da rede. Para a SDN, esse conceito é executado por meio do *switch*. *O que difere essas redes é que a SDN também possui uma visão para redes programáveis mediante software. A programabilidade do plano de controle centralizado com a visão global da rede dos elementos encaminhadores se constitui em uma poderosa facilidade que proporciona um processamento ativo e dinâmico de todos os fluxos da rede [20].*

2.2 PROTOCOLO OPENFLOW

Pesquisas realizadas na Universidade de Stanford deram origem ao protocolo de código aberto OpenFlow, tecnologia de rede que funcionou como ponto de partida para o surgimento do conceito de Software-Defined Networking-SDN [21]. O OpenFlow foi proposto para atender à demanda de validação de novas propostas de arquiteturas e protocolos de rede, possibilitando a comunicação entre controladores e equipamentos de comutação, como switches, roteadores, pontos de acesso sem fio e o isolamento de recursos entre usuários [22].

O OpenFlow vem viabilizar a implementação de uma tecnologia que possui como proposta ser uma arquitetura dinâmica, gerenciável, adaptável e com redução de custos, necessitando de um protocolo de comunicação para realizar a interação entre o controlador SDN e os dispositivos da rede, sendo o OpenFlow o protocolo mais utilizado para essa finalidade [23], pois é o que permite a implementação dos conceitos das redes definidas por software.

Com a utilização do OpenFlow, é possível programar os switches de forma que possam ser gerenciados por um software controlador externo, quando o comutador recebe comandos para os fluxos de pacotes de acordo com regras predeterminadas e realiza o armazenamento nas tabelas existentes. Um dos resultados da utilização desse protocolo se refere ao conceito de fluxo, constituído pela combinação de campos do cabeçalho do pacote que são processados pelo switch [24].

As regras e ações instaladas no hardware de rede é responsabilidade do controlador SDN. Pode-se dizer que o OpenFlow se trata de um protocolo padrão que possibilita uma interface para comunicação entre a camada de controle e de encaminhamento dentro da arquitetura SDN, permitindo que aplicações programem as tabelas de fluxo dos dispositivos de encaminhamento [25]. A arquitetura do OpenFlow consiste em três elementos principais:

- *Dispositivos de encaminhamento compatíveis com OpenFlow;*
- *Uma ou mais instâncias de controlador;*
- *O canal seguro para a comunicação entre os dispositivos e as instâncias de controlador.*

Alguns dos principais benefícios da implementação de uma rede SDN estão na possibilidade das diversas formas de dividir seus recursos e de associar todo um processamento complexo, definido por software a pacotes que se encaixem em determinado padrão, podendo então unir

comportamentos de rede distintos em uma única estrutura de comunicação [26].

As tabelas de fluxo contêm regras de encaminhamento dos pacotes da rede e são implementadas na controladora e repassadas ao switch utilizando o protocolo OpenFlow. Após a regra associada ser identificada, são realizadas as ações associadas e o incremento dos contadores. A figura 2.2 demonstra as camadas do protocolo OpenFlow e a comunicação com o controlador.

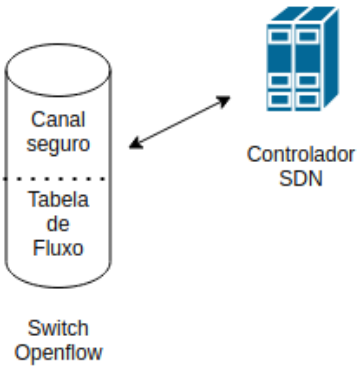

Figura 2.2: Protocolo OpenFlow adaptado de [2]

.

2.3 PLANO DE CONTROLE

O plano de controle da rede se encontra instalado nas controladoras – as quais podem estar no mesmo segmento de rede ou operando em um ambiente remoto – e tem como principal diferencial permitir a integração de aplicações e funcionalidades ao plano de dados. Essa agregação de novos recursos em redes tradicionais depende da atualização do firmware do fabricante; essa limitação não ocorre em redes SDN, nas quais a mesma aplicação pode ser executada em diferentes equipamentos físicos ao mesmo tempo.

Na figura 2.3, é demonstrada a associação das camadas de aplicação, controle e infraestrutura conectadas por meio de APIs dentro da arquitetura SDN, onde a API de interface do plano de controle e do plano de dados integra a controladora aos dispositivos da rede na camada de infraestrutura.

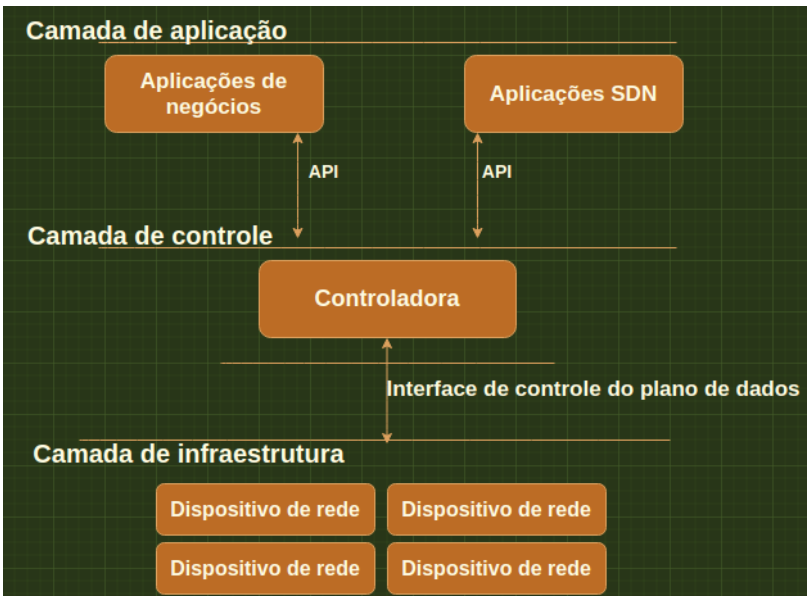

Figura 2.3: Integração das camadas de aplicação, controle e infraestrutura.
Adaptado de [23]

2.4 CONTROLADORES SDN

O ponto principal de uma rede SDN é o controlador, também chamado de sistema operacional de rede. O controlador realiza a interconexão de todos os elementos programáveis que compõem a rede de uma maneira centralizada, sendo o ativo responsável pela tomada de decisões, exerce o papel de absorção da função da infraestrutura física, exercendo a funcionalidade de gerenciador dos serviços antes executados pelo hardware nas redes tradicionais, fazendo a gestão dos fluxos de entrada na rede. Trata-se de um software que implementa o protocolo OpenFlow, tornando viável o gerenciamento centralizado da rede [27]; é o responsável por monitorar, gerenciar e tomar decisões referentes à rede de dados. Quando um pacote não é identificado na tabela de fluxo do switch, ele é encaminhado ao controlador para que avalie qual ação será adota de acordo com a lógica pré-programada.

O controlador fornece uma interface para criar, modificar e controlar o fluxo de tabelas do comutador. Essa interface se chama API [28]. O controlador é executado normalmente em um servidor conectado à rede, sendo parte integrante de uma arquitetura de rede SDN e, para que sua comunicação com switches OpenFlow ocorra, o controlador deve ter suporte ao OpenFlow. Há diversas possibilidades de controladores para serem utilizados em estrutura SDN e neste documento são mencionados alguns dos mais utilizados.

2.4.1 NOX

NOX foi o primeiro controlador SDN desenvolvido inicialmente por Nicira Networks, junto com o OpenFlow [29]. Foi doado para a comunidade SDN Open source, tornando-se a base para as posteriores soluções de controladores. O controlador NOX tinha por objetivo prover uma interface de programação de alto nível que pudesse desenvolver o gerenciamento da rede. O NOX clássico possui versão disponível sob licença pública geral, suporta as linguagens C++ e Python, há uma versão deste controlador chamada de novo NOX que possui apenas a versão em C++ o que, conforme pesquisas, a torna mais rápida e de melhor qualidade [30]. De acordo com os estudos de pesquisa, o controlador NOX está obsoleto, sendo seu sucessor o POX.

O NOX define um sistema operacional de rede, tal que a rede é composta por switches e um ou mais servidores que executam o software NOX e as aplicações de gerenciamento nele implementadas [31].

A figura 2.4 demonstra a topologia da estrutura SDN utilizando o controlador NOX.

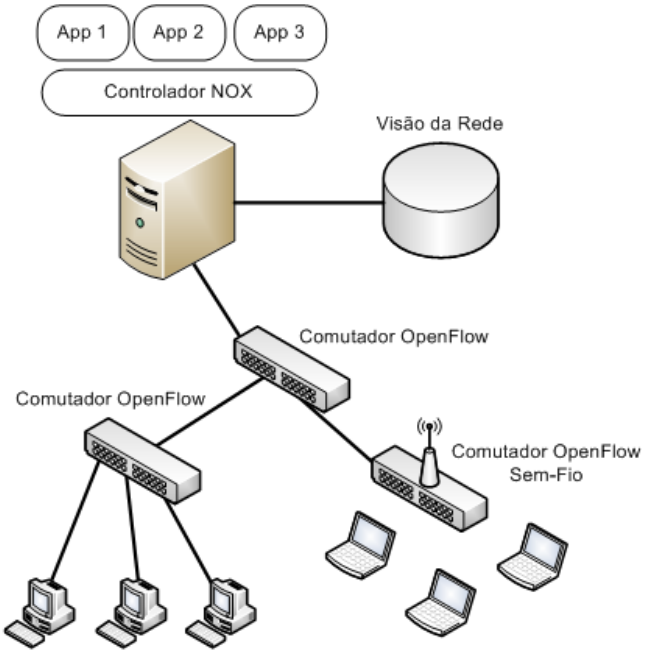

Figura 2.4: Controlador NOX [3]

2.4.2 POX

O POX é um controlador SDN desenvolvido para pesquisas de ensino [32], possui interface mais amigável e elegante do que a do NOX [33], e se trata de um *software* de rede que utiliza Python. O POX tem uma biblioteca exclusiva para a análise e construção de pacotes, a biblioteca pox.lib.packet.

Algumas das principais funcionalidades do controlador POX são [34]:

- Possuir componentes de simples reutilização como, por exemplo, para descoberta de caminho e de topologia;
- Rodar em qualquer ambiente de sistema operacional;
- Suportar a Interface Gráfica de Usuário e arquitetura virtual do NOX;
- Fornecer um melhor desempenho em comparação ao NOX.

2.4.3 RYU

Ryu é um controlador de rede SDN que pode ajudar a gerenciar o tráfego, aumentar a largura da banda, alocar recursos, criar fluxos, entre outras funções de configurações. Utilizar o Ryu como controlador torna mais fácil o gerenciamento e a adaptação do tráfego, dessa forma o conceito de SDN se torna mais consolidado nos tempos reais, podendo todos os dispositivos serem gerenciados por um único controlador [35]. Desenvolvido pelo Centro de Inovação em software da empresa japonesa NTT sob a licença apache 2.0 [36], o projeto Ryu é um arcabouço baseado em componentes para programação de redes definidas por *software*, utiliza Python como linguagem de programação e permite programar aplicações utilizando vários protocolos, entre os quais se destacam no contexto deste trabalho os protocolos OpenFlow 1.0, 1.2, 1.3 e 1.4. Suporta tanto o protocolo OF-CONFIG, do inglês *Openflow Management and Configuration Protocol*, como o protocolo NETCONF, do inglês *Network Configuration Protocol*, na função de gerenciamento de configurações de *switch*, possibilitando a criação de ferramentas para a unificação de alteração de configurações, tanto dos *switches* habilitados com o OpenFlow como para os dispositivos de rede convencionais [37].

O Ryu é o controlador utilizado neste projeto de pesquisa, foi escolhido devido a suas funcionalidades serem de simples utilização e fácil implementação. Sua estrutura é simples e de fácil entendimento, como demostrado na figura 2.5, que mostra as camadas de funcionamento da arquitetura desse controlador.

Figura 2.5: Arquitetura de funcionamento do Controlador RYU
adaptado de [35]

2.4.4 Floodlight

O controlador Floodlight é escrito na linguagem Java, licenciado pela Apache Foundation e suportado pela comunidade de desenvolvedores, incluindo um grupo de engenheiros da Big Switch Networks [38], possui suporte ao OpenFlow e continua com seu desenvolvimento ativo de acordo com seu repositório de código. Verificando a possibilidade de sua utilização, ele foi descartado por possuir uma curva de aprendizado grande [39]. O Floodlight foi projetado para funcionar com o crescente número de switches, roteadores, comutadores virtuais e pontos de acesso que suportem o padrão OpenFlow [40].

2.4.5 OpenDaylight

A primeira versão foi a Hydrogen, que foi lançada em fevereiro de 2014 e tinha um controlador de código aberto, juntamente com alguns plugins e ferramentas de virtualização [41]. O OpenDaylight implementa o protocolo OpenFlow por meio de uma API HTTP, permitindo que outras aplicações possam interagir com os equipamentos/switches controlados através dele.

O núcleo da plataforma OpenDaylight é a camada de abstração de serviço orientada a modelos (MD-SAL), seus dispositivos de rede e aplicativos são tratados como objetos ou modelos. Possui uma comunidade ativa e crescente de centenas de desenvolvedores, que estão em constante evolução e expandindo a plataforma [42].

2.5 SEGURANÇA

Pode-se afirmar que, com o crescimento do mercado corporativo mundial, a informação se tornou um dos ativos mais preciosos de uma empresa, passando a ser indispensável a estratégia empresarial [43]. A preocupação em garantir a segurança da informação não é uma questão apenas dos dias atuais. Há décadas, o segmento político e militar, em principal, considera primordial manter o sigilo de suas informações e, com a inevitável introdução da tecnologia para a comunicação de dados, a ameaça tornou-se mais evidente. Nesse contexto, a informação vem a ser um ativo importante aos negócios, possuindo um valor importante para uma organização, necessitando de segurança adequada por meio de políticas e regras capazes de proteger esse ativo que, no decorrer do tempo, resultou em um recurso crítico para a execução de negócios.

A segurança da informação é o conjunto de orientações, normas, procedimentos, políticas e demais ações que tem por objetivo proteger o recurso da informação, possibilitando que o negócio da organização seja realizado e sua missão seja alcançada [44]. Os principais pilares da segurança da informação são a confidencialidade, integridade e disponibilidade conforme descritos na IEC 27.002:2005 normas da ABNT [45]:

- *Confidencialidade: propriedade de que a informação não esteja disponível ou revelada a indivíduos, entidades ou processos não autorizados;*
- *Integridade: propriedade de salvaguarda da exatidão e completeza de ativos;*
- *Confidencialidade: propriedade de estar acessível e utilizável sob demanda por uma entidade autorizada.*

Para tratar a falta de segurança nas redes tradicionais, utilizamos recursos como IDS, IPS, Firewall entre outros. Esses recursos também podem ser utilizados nas redes SDN com a mesma eficiência de quando utilizados em redes tradicionais.

2.5.1 Sistema de Detecção de Intrusão

O Sistema de Detecção de Intrusão (IDS) é uma solução que pode identificar uma invasão da rede com intenções maliciosas ou pode apenas detectar um usuário não autorizado conectado à rede de dados. O IDS não realiza uma varredura na rede, faz a captura de uma cópia do pacote e uma análise de seu conteúdo. Ele é capaz de identificar e analisar os dados que trafegarem pela rede mediante a observação de pacotes que possuem padrões predefinidos e podem atuar em tratativas para soluções de possıveis invasões na rede [27]. Não há uma ação de prevenção ao ataque, o IDS apenas identifica pacotes fora do padrão determinado e realiza uma notificação informando possíveis problemas.

O IDS utilizado em conjunto com a tecnologia das redes SDN proporciona uma maior possibilidade de segurança efetiva a essa modalidade de arquitetura de rede. A Associação do

IDS a uma rede SDN permite introduzir maior segurança para esse ambiente [46]. Um dos objetivos desse sistema é acionar alarmes para atividades maliciosas e tentativas não autorizadas. Basicamente, o sistema identifica atividades e ataques maliciosos e se enquadra em duas categorias: o Host Intrusion Detection System – HIDS, baseado em host, e o Network Intrusion Detection System – NIDS baseado em rede. O HIDS é instalado nas estações de trabalho, monitorando o tráfego de rede individualmente.

Figura 2.6: Fluxo do funcionamento do IDS [4]

A figura 2.6 demonstra o fluxo de funcionamento do sistema IDS, onde os dados são capturados e analisados e, caso necessário, são gerados alertas de identificação do tráfego que não se enquadre no padrão predefinido. É composto por três componentes:

- Primeiro componente do IDS é o data source: onde são inseridas as regras para que possa identificar as vulnerabilidades;
- Segundo componente é o motor de análise que realiza a detecção de atividades maliciosas;
- Terceiro componente do IDS é o gerenciador de respostas, item responsável por armazenar e informar ao usuário quando um possível ataque for identificado.

2.5.2 Sistema de Prevenção de Intrusão

O Sistema de Prevenção de Intrusão (IPS) atua na prevenção de invasões nas redes de comunicação. Trata-se de uma ferramenta que possibilita tomar decisões de intervenções na rede, podendo realizar bloqueios de tráfego malicioso. Esta ferramenta se baseia em conteúdo de aplicações e não apenas nos endereços IPs ou em portas, como é feito pelo *firewall*. Diferentemente do IDS, que trabalha apenas na identificação e análise dos pacotes, o IPS também realiza o bloqueio caso seja identificada uma ameaça ao funcionamento adequado do ambiente da rede. Este *software* de prevenção de intrusão pode ser associado ao ambiente de uma rede SDN, pois, devido a prover uma prevenção de ataques maliciosos, proporciona uma segurança maior associando seu funcionamento a um controlador.

Uma solução de segurança constituída pela associação de um conjunto IDS/IPS é chamado de NIDS/NIPS Network Intrusion Detection System/Network Intrusion Prevention System. Esse conjunto realiza o monitoramento e a atuação de intervenções como bloqueio na rede em que estejam acoplados, instalados em *swtiches* ou roteadores. Um sistema de IPS pode tomar várias ações, dentre elas: bloqueios de portas no *switch*, interação com políticas de *firewall* externos, regras dos roteadores, ou, ainda, geração de tráfego na camada de transporte [47].

O IPS possui entre suas funcionalidades impedir uma possível tentativa de invasão em uma rede de dados. A figura 2.7 demonstra a diferença das ações realizadas pelo IDS e o pelo IPS. No primeiro quadrante da figura, o IDS identifica o tráfego malicioso e emite apenas um alerta; no segundo quadrante da figura, o IPS, ao identificar o tráfego não desejado, realiza o seu bloqueio.

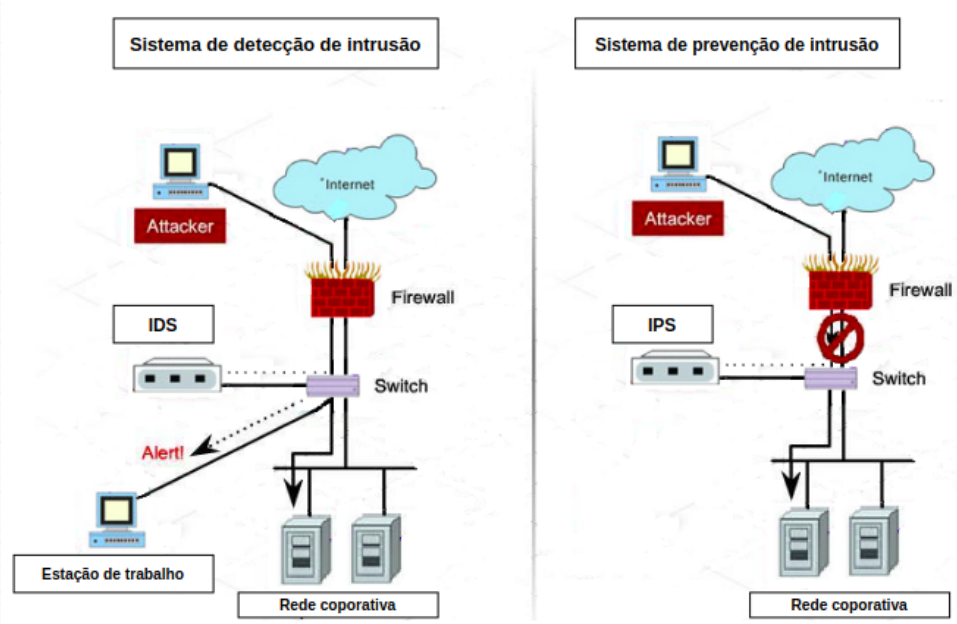

Figura 2.7: Diferença entre o funcionamento do IDS e do IPS. Adaptado de [4]

2.5.3 Firewall

Um firewall é um conjunto de hardware e software que possui a finalidade de proteger uma rede local de possíveis ameaças provenientes da Internet por meio do controle de tráfego de dados entre redes interconectadas, utilizando regras pré-configuradas. Existem alguns tipos de firewall, sendo os mais comuns os filtros de pacotes, filtro de estado de sessão, Gateway de aplicação (Proxy) e firewall de nova geração [48].

- *Filtro de pacotes utiliza regras preestabelecidas para analisar o cabeçalho do pacote de forma a liberar ou bloquear as informações;*
- *Filtro de estado de sessão é um aprimoramento do filtro de pacotes, examina o status das conexões ativas. Permite ou bloqueia o pacote de acordo com o estado, a porta ou o protocolo;*
- *Firewall de aplicação ou proxy de rede atua na camada de aplicação do modelo OSI, sendo possível estabelecer regras específicas para cada aplicação;*
- *Firewall de nova geração atua em todas as camadas do modelo OSI, principalmente na camada de aplicação. Entre seus diferenciais de funcionamento estão sua atuação na visão de usuários, filtro URL (Proxy), IPS/IDS. A topologia do firewall é demonstrada na figura 2.8, onde ele está posicionado estrategicamente entre a rede interna e a saída para a Internet.*

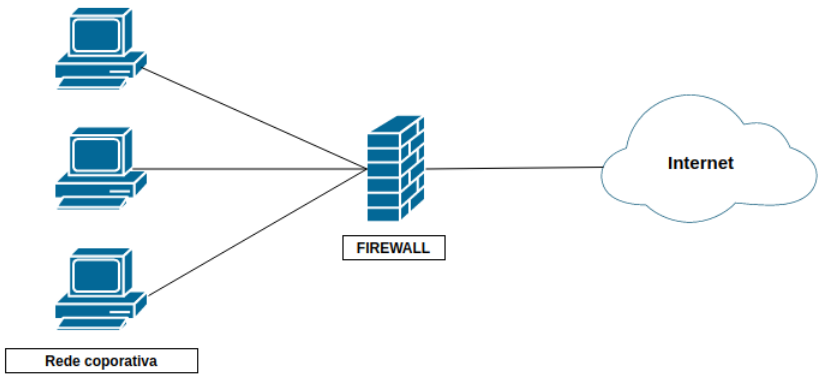

Figura 2.8: Firewall
[Autor]

2.5.4 Falhas na segurança de dados

A segurança da informação é uma ferramenta fundamental para viabilizar, manter e homologar a cadeia de serviços das organizações que trabalham diretamente com informações sigilosas [49].

Em segurança da informação, as vulnerabilidades existentes em uma estrutura de redes fazem com que os dados sigilosos de uma instituição sejam ameaçados. Entre os principais pontos de falha que geram a vulnerabilidade estão a falha do sistema, o acesso do atacante à falha, a capacidade do atacante de explorar a falha. Um dos pontos mais vulneráveis para o sigilo dos dados transmitidos de uma empresa é o ser humano: o usuário da rede acessa e compartilha os dados, podendo compartilhar informações que deveriam possuir acesso restrito [49].

2.5.5 Ataques de negação de serviço DDoS e DoS

Um ataque de negação de serviço distribuído (DDoS) possui por finalidade que um servidor ou website fique inoperante. Isso ocorre porque o alvo recebe uma quantidade muito elevada de requisições oriundas de diversas fontes diferentes e, não sendo possível atendê-las da forma adequada, o servidor ou website deixa de funcionar. Ataques DDoS podem utilizar até milhares de computadores para atacar determinada máquina, distribuindo a ação entre elas [50]. Ataques de negação de serviço (DoS) tem por objetivo principal interromper o acesso dos usuários pelo esgotamento dos recursos computacionais, tais como processador, rede, memória. Um ataque DoS, assim como o ataque DDoS, possui o intuito de tornar um site ou uma rede indisponível para os usuários, objetivando assegurar que toda a infraestrutura de usuários seja inutilizada, porém é proveniente de um único host, ou seja, um único endereço IP. São enviados ao destino diversas solicitações de pacotes, e essa grande quantidade de requisições de uma única vez sobrecarrega o servidor e os recursos da rede, gerando lentidão e até mesmo a queda do serviço [51]. Em um ataque DoS, um perpetrador pode usar uma única conexão à Internet para explorar uma vulnerabilidade de software ou inundar o alvo com solicitações falsas e, finalmente, fazer com que o site fique indisponível, impedindo-o de responder aos pedidos dos usuários legítimos.

A figura 2.9 demonstra um ataque distribuído de negação de serviço.

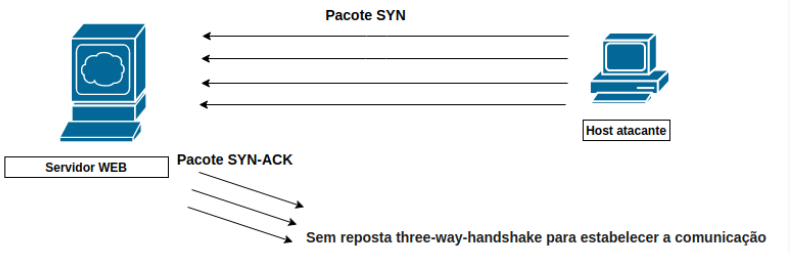

Figura 2.9: Fluxo de um ataque DoS
[Autor]

2.6 VIRTUALIZAÇÃO

A virtualização é a capacidade de criar uma abstração de alto nível de uma plataforma de hardware, servidor, serviço ou até mesmo dispositivos de armazenamento de uma forma fracamente acoplada ao hardware subjacente por meio de camadas adicionais de software [52]. As tecnologias de virtualização são utilizadas para permitir que mais de um recurso computacional seja implementado utilizando um mesmo meio físico, ou seja, utilizando um mesmo hardware é possível gerar várias máquinas virtuais, sejam elas hosts, servidores ou switches. Esse recurso pode ser utilizado para criar redes lógicas que se integram facilmente e permitem um maior poder de programabilidade.

Virtualização é qualquer forma de particionar ou combinar um conjunto de recursos de redes e abstraí-lo para o usuário de tal forma que cada usuário, por meio de seu conjunto de recursos particionados ou combinados, tenha uma visão única e separada da rede [53]. Sua utilização reduz os custos da montagem de uma rede de dados, pois compartilha os hardwares, armazenamentos, também podendo virtualizar aplicações. Com a criação de redes virtuais, sobrepostas logicamente na rede física, a otimização de recursos de uma rede de comunicação tornou-se mais prática e flexível, podendo ser facilmente ajustada conforme surge a necessidade de sua ampliação. Virtualização de redes simplifica as operações e reduz o tempo para provisionamento, melhorando e proporcionando uma maior eficiência do uso de recurso e das operações [54].

A figura 2.10 mostra algumas aplicações virtualizadas, entre elas o SaaS, implementação de software como um serviço, permitindo o acesso a aplicações na nuvem, e outras aplicações como aPaaS, PaaS e iaaS.

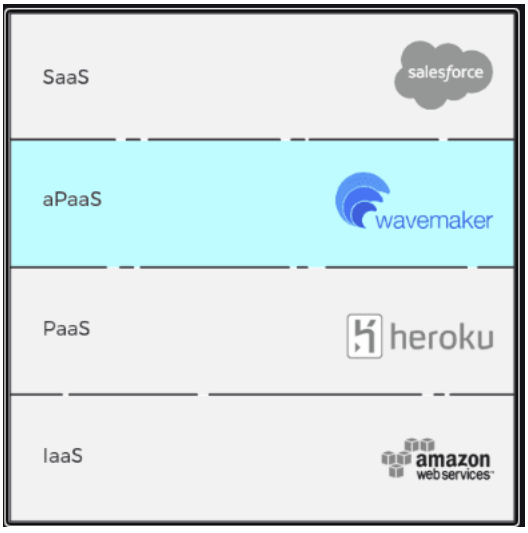

Figura 2.10: Plataforma de aplicações virtualizadas [4]

2.6.1 Redes Locais Virtuais

Uma Rede Local Virtual (VLAN) é um agrupamento lógico de dispositivos ou usuários que podem ser unidos por função, departamento ou aplicativo, independentemente da localização de seus segmentos fisicos. A configuração de VLANs é feita no switch, podendo ser estendida tambem ao roteador, utilizando o protocolo IEEE 802.1Q software ou protocolos proprietários como Cisco ISL [55].

A ideia central da tecnologia VLAN é dividir uma rede local em segmentos lógicos em vez de físicos, permitindo que ocorra uma segmentação de acordo com regras de negócios e departamentos, como financeiro, administrativo, vendas e qualquer outra separação que atenda aos requisitos corporativos.

Para que ocorra comunicação entre elementos de diferentes VLANs é necessário a utilização de um roteador ou um switch de camada 3, que pode realizar o encaminhamento de pacotes entre vlans ou na mesma vlan. [56]. A figura 2.11 demonstra um modelo de comunicação entre vlans, utilizando uma arquitetura amplamente utilizada em ambientes de Data Center conhecida como Spine and Leaf. Nesta arquitetura há diferentes tipos de switches como os utilizados para receber a conexão de usuários finais (Rack1) e os switches utilizados para intercomunicar as diferentes VLANs (Spine e Core).

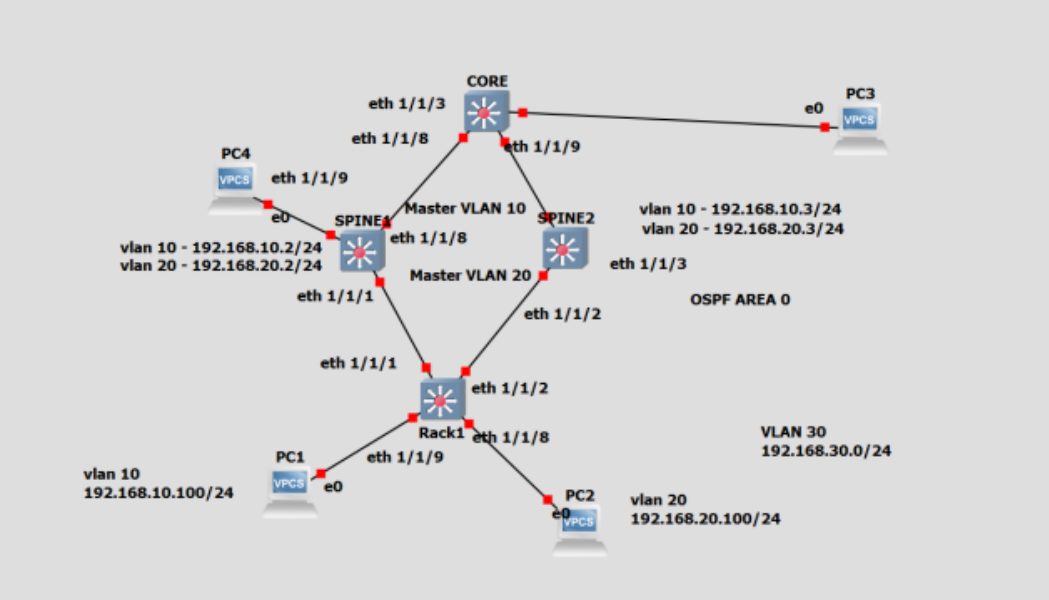

Figura 2.11: Comunicação entre VLANS
[Autor]

2.6.2 Redes Privadas Virtuais

A Rede Privada Virtual (VPN) é uma rede de acesso privada que trafega pela Internet através de um ambiente controlado, normalmente um túnel, que possui segurança, ou seja, pode ser criptografado. Uma VPN é dada por meio do estabelecimento de uma conexão virtual ponto a

ponto, utilizando-se de conexões dedicadas, protocolos de encapsulamento virtuais, ou criptografia [57]. A VPN normalmente é acessada a partir de um túnel de conexão direta, sendo possível solicitar autenticação para o usuário, podendo também os dados serem criptografados, onde se navega de forma segura. A VPN pode conectar as filiais de uma empresa às suas matrizes, utilizar aplicações e sistemas internos dessas empresas de forma segura através de ferramentas para permitir o acesso de clientes remotos autorizados a esses recursos de uma rede corporativa e viabilizar a interconexão de redes geograficamente distantes. Em geral, uma VPN deve estar sempre possibilitando o compartilhamento de recursos e informações, além de assegurar privacidade e integridade dos dados que trafegam pela Internet [58].

Formas de conexão VPN:

- *Tunelamento: dá-se pela forma como os dados trafegam pela conexão VPN. A ideia de túnel surge quando, ao enviar os dados de uma das extremidades da conexão, primeiro se criptografa e depois se encapsula o pacote original dentro de um novo pacote;*
- *Criptografia: permite garantir a autenticidade e a privacidade da informação transmitida mediante chaves públicas e privadas por meio de algoritmos;*
- *Autenticação: quando o usuário necessita utilizar a inclusão de seus dados para realizar o seu acesso à VPN. A figura 2.12 mostra um modelo de VPN utilizado pela empresa Cisco.*

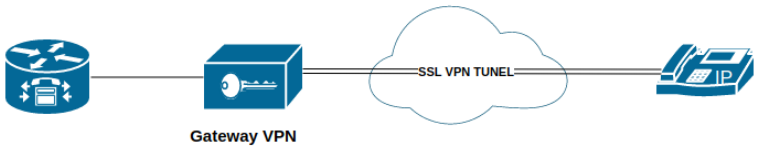

Figura 2.12: Topologia de um túnel VPN
Adaptado de [59]

2.6.3 Redes Overlay

As redes overlay, ou redes sobrepostas, são redes de comunicação construídas sobre outras redes preexistentes, a exemplo das redes P2P e das redes virtuais. Os hosts de uma rede overlay são conectados por links virtuais/lógicos, por meio da estrutura física de hardware. A maioria das redes overlay rodam sobre a internet pública, que começou como uma rede de pesquisa overlay rodando sobre a infraestrutura da rede telefônica pública comutada [60]. Os nós desse tipo de redes são conectados através de links lógicos, que podem abranger várias ligações físicas. A rede overlay permite introduzir funcionalidades mais complexas por cima das oferecidas pelas camadas subjacentes, possibilitando assim superar algumas das limitações subjacentes e, ao mesmo tempo, oferecer novas características de routing e forwarding sem ter que mudar os routers [61].

A figura 2.13 demonstra o serviço funcionando sobre uma rede overlay instalada em uma estrutura underlay ou seja, rede física existente.

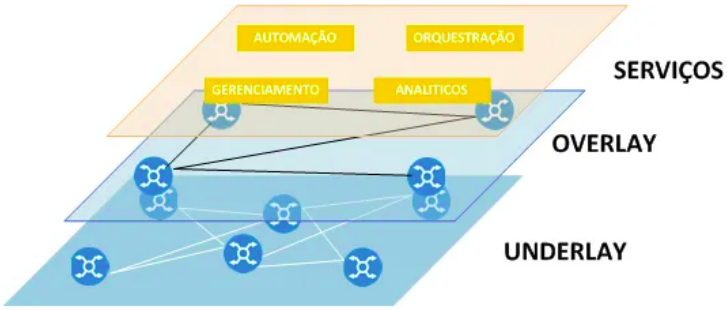

Figura 2.13: Redes *Overlay* – Sobrepostas
[62]

2.6.4 Emuladores e simuladores de rede

Os emuladores de redes são soluções para simular uma rede de computadores com o intuito de auxiliar em estudos e pesquisas com a finalidade de decifrar problemas ou implementar soluções em um ambiente simulado. Este modelo de solução reduz custos na aquisição de equipamentos, traz uma solução rápida, podendo ser instalado em um equipamento já existente emulando equipamentos físicos com funcionalidades reais de uma rede. Há vários emuladores e simuladores no mercado dentre os quais alguns são exemplificados logo a seguir.

2.6.5 GNS3

O GNS3 é um software de código aberto e gratuito que emula o hardware de um dispositivo e possibilita a execução de imagens reais no dispositivo virtual. Por exemplo, pode-se copiar o IOS de um roteador cisco físico real e executá-lo em um roteador cisco virtual emulado no GNS3. Este software também simula os recursos e funcionalidades dos dispositivos de redes, executando sistemas operacionais reais. O GNS3 consiste em dois componentes de software, interface gráfica do usuário (GUI) e a máquina virtual (VM GNS3) [63].

2.6.6 VMWARE

As máquinas virtuais são computadores de software com a mesma funcionalidade que os computadores físicos [64]. Estes softwares são executados em computadores existentes utilizando o seu hardware. Operam sistemas operacionais e aplicativos existentes de maneira real com um comportamento de um computador existente. A máquina virtual é separada por sandbox do restante do sistema. O sandbox é um mecanismo de segurança que separa programas em execução. Dessa forma, o que é executado na VM não pode interferir nas configurações do

computador hospedeiro o que a torna segura em relação a testes de ataques maliciosos.

2.6.7 Mininet

O Mininet é uma ferramenta de emulação de rede. Possui a capacidade de reproduzir uma arquitetura de rede em um ambiente virtualizado de forma que seja possível executar aplicações e sistemas pertencentes a outras plataformas sem que haja a necessidade de adquirir esses *hardwares*, reduzindo os custos. Estes programas criam uma camada de *software* entre as plataformas hóspedes e hospedeiras [65]. O Mininet é um emulador de rede que cria uma rede de *hosts*, comutadores, controladores e links virtuais. Os *hosts* Mininet executam o *software* de rede Linux padrão, e seus *switches* suportam o OpenFlow para roteamento personalizado altamente flexível e rede SDN [66]. O ambiente escolhido para simulação da solução proposta é o emulador Mininet devido às diversas vantagens obtidas com a sua utilização. De acordo com mininet.org, são elas:

- Fornece uma plataforma de teste de rede simples e barata para o desenvolvimento de aplicativos OpenFlow;
- Permite que vários desenvolvedores simultâneos trabalhem independentemente na mesma topologia;
- Suporta testes de regressão no nível do sistema, que são repetíveis e facilmente empacotados;
- Permite testes complexos de topologia sem a necessidade de conectar uma rede física;
- Inclui uma CLI com reconhecimento de topologia e OpenFlow para depuração ou execução de testes em toda a rede;
- Oferece suporte a topologias personalizadas arbitrárias e inclui um conjunto básico de topologias parametrizadas;
- É utilizável imediatamente, sem programação.

Este emulador proporciona uma forma simples de simular o comportamento de um ambiente de rede, ocasionando o desempenho correto e esperado do sistema e podendo ser implementado em diversas topologias e estruturas de rede.

2.6.8 API

Uma Interface de Programação de Aplicações (API) Application Programming Interface) torna possível a integração e comunicação entre tecnologias diferentes utilizando um conjunto de regras, padrões e rotinas que fazem o intermédio da troca de informações entre aplicações. A API oferece caminhos para que se possa acessar as informações; isso está relacionado à

ferramenta ou à biblioteca que pode ajudar os desenvolvedores a acessarem os dados dentro de seu código fabricante para fazer a tratativa das informações necessária para o desenvolvimento de sua ferramenta [67].

Existem três tipos de APIs [68]:

- *privadas: são utilizadas internamente entre as aplicações de uma empresa, ou seja, localmente;*
- *parceiros: são utilizadas entre parceiros de negócios ou para permitir a integração entre diferentes softwares;*
- *públicas: podem ser utilizadas livremente. Muitas vezes são disponibilizadas pela empresa para que os desenvolvedores possam fazer a integração com outras aplicações.*

Uma REST API – Representational State Transfer é quando, dentro de uma API, são definidas funções em que um desenvolvedor possa executar solicitações e receber o retorno das respostas [67].

Um controlador SDN tem duas funções principais, comunicar com os dispositivos de encaminhamento através da API southbound e oferecer serviços às aplicações através da API northbound. A maioria dos controladores suporta apenas o OpenFlow na API southbound e existem inúmeras APIs northbound, como APIs ad-hoc, APIs Restful, interfaces de programação multinível e sistemas de arquivos [52].

2.6.9 Plano de Gerenciamento

O plano de gerenciamento permite que o administrador de uma rede SDN possa programar os equipamentos da rede para executar diversas funcionalidades, obter informações sobre seu funcionamento e até mesmo realizar notificações sobre eventos ocorridos no decorrer de seu funcionamento. O controlador da rede em conjunto com plano de gerenciamento define, através de linguagens de programação, como será realizada a aplicação desses comandos. A comunicação entre esses dois elementos é realizada através das APIs. Essas funções são realizadas com o auxílio da integração das APIs, interfaces que facilitam a programação de aplicações utilizadas na programabilidade desta modalidade de rede. As APIs comumente utilizadas são a Northbound Interface e a Southbound Interface.

2.6.10 Northbound Interface

A comunicação entre o plano de gerenciamento e o plano de controle é realizada através da API Northbound Interface. Esta API permite a monitoração da rede, é usada principalmente para integração com aplicativos de gerenciamento de terceiros para gerenciamento de falhas e fluxo de provisionamento [69]. A interface Northbound fornece topologia de rede e recuperação de

configuração e serviços de provisionamento. A API Southbound Interface também participa desta funcionalidade e será descrita logo adiante.

Dentre as variadas funções da Northbound Interface, a principal é decifrar as informações das aplicações de gerenciamento em instruções de baixo nível para os dispositivos da rede e transmitir estatísticas sobre a rede, que foram geradas nos dispositivos da rede e processadas pelo controlador [70].

2.6.11 Southbound Interface

A API Southbound Interface permite que o controlador se comunique com as informações dos requisitos de aplicações utilizados na rede SDN, proporcionando a programação de equipamentos com finalidades diversas dentre as quais estão contidas o controle de fluxo, firewall, sistemas de detecção de intrusos (IDS), além de roteamento e comutação.

A Southbound Interface é usada para os equipamentos de rede se comunicarem com o controlador. Isso ocorre em três casos [70]:

- Envio de avisos de eventos caso ocorra uma mudança de porta ou enlace;
- Envio de estatísticas de fluxo gerado com o tempo e enviadas para o controlador, de forma a fornecer informações mais detalhadas sobre as características da rede para administradores da rede;
- Envio de pacotes para o controlador em dois casos: se os equipamentos do plano de dados não souberem o que fazer com um pacote, ou seja, quando não há uma regra definida para pacotes com alguma característica; ou quando alguma das regras instaladas no equipamento tiver como comando "enviar para o controlador".

A southbound interface pode se basear em vários protocolos, tais como: OpenFlow, OVSDB, ForCES [10] e ainda outros, como SNMP, NETCONF, BGP [8]. O protocolo mais utilizado nesta interface é o OpenFlow, suportado pela grande maioria dos equipamentos SDN do mercado.

A figura 2.14 mostra a disposição das interfaces northbound API e southbound API inseridas em uma arquitetura SDN em camadas.

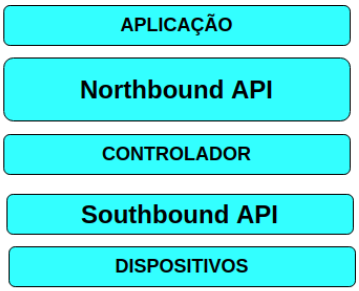

Figura 2.14: Disposição da API inserida na arquitetura SDN.
Adaptada de [67]

2.6.12 CDN

Uma CDN – Content Delivery Network é um conjunto de servidores de borda ou datacenters distribuídos por várias partes do país ou do mundo. Este serviço é utilizado pelos provedores de streams, como Netflix, para que se carregue o conteúdo solicitado mais rapidamente, além de evitar a queda do servidor de origem, pois reduz a quantidade de acessos solicitados a ele. Pode-se dizer que uma CDN corresponde a uma rede sobreposta de servidores atuando de forma colaborativa, com o objetivo de entregar conteúdos digitais aos clientes, atendendo a requisitos de qualidade de serviço [71].

Uma cópia do arquivo solicitado é armazenada em um ambiente mais próximo ao usuário que solicita o arquivo. Esse fato promove menor latência e mais rapidez ao acesso do conteúdo solicitado, visto que não se faz necessário ir à origem dos dados sempre que algum arquivo for requisitado, melhorando consideravelmente a ocupação do núcleo da rede e consequentemente a experiência do usuário.

> *A Rede de Entrega de Conteúdo – CDN é composta por uma plataforma de servidores altamente distribuída que ajuda a minimizar atrasos no carregamento de conteúdo de páginas da Web, reduzindo a distância física entre o servidor e o usuário, permitindo um acesso mais rápido para os usuários, reduzindo o delay de carregamento. Sem uma CDN, os servidores de origem de conteúdo devem responder a cada solicitação do usuário final. Isso resulta em tráfego significativo para a origem e carga subsequente, aumentando assim as chances de falha na origem se os picos de tráfego forem excessivamente altos ou se a carga for persistente [72].*

A utilização de CDNs pode ser uma solução para auxiliar no combate a ataque DDoS, entretanto esta solução se aplica para sites, provedores de conteúdo e Web APPs. Para aplicações que precisam ser acessadas em redes privadas, é necessária uma solução de segurança capaz de mitigar os ataques no perímetro da rede; uma solução que pode atuar com essa finalidade é a Black Holes. Segundo [73], a técnica Remotely triggered black hole (RTHB) permite descartar tráfego indesejável antes que ele entre em uma rede protegida.

2.6.13 Honeynet

Honeynet é um ambiente de rede criado para estudo e pesquisa, elaborado para ser comprometido a vulnerabilidades com o intuito de observar o comportamento de ataques maliciosos. Foi projetado por um grupo de pesquisadores para ser comprometida, com o objetivo de revelar as ferramentas, táticas e motivações dos invasores [74].

As honeynets podem ser reais, fornecem sistemas operacionais reais com que os intrusos possam interagir, e honeynets virtuais permitem executar todos os sistemas operacionais, aplicações e serviços no mesmo hardware através de um software de virtualização. As honeynets virtuais estão divididas ainda em duas categorias: autocontenção, na qual todos os dispositivos geram alertas e realizam coletas de dados, e híbridas, realizando uma combinação de honeynets reais e virtuais [16].

2.6.14 DMZ- Zona desmilitarizada

A DMZ – Demilitarized Zone é uma rede que funciona como uma área intermediária entre a rede local e a Internet. Trata-se de uma rede de perímetro, normalmente implantada para alocar serviços de acesso público. Sua principal função é separar os serviços que possuem acesso externo da rede local [75]. Esse recurso provê o isolamento físico entre as redes aumentado a segurança de seus dados.

2.7 TRABALHOS CORRELATOS

Neste capítulo, são apresentados trabalhos que tratam de temas semelhantes ao desta pesquisa. Nos parágrafos subsequentes, serão expostos trabalhos que abordam redes SDN, IDS, ataques de negação de serviços e redes honeynet que são comparados ao presente trabalho.

Dhawan et al., em seu artigo de pesquisa, propõem detectar um ataque de negação de serviço através do monitoramento da taxa de novas regras criadas pelo controlador da rede SDN [76]. Se a taxa de regras criadas for superior ao limiar previamente estabelecido pela defesa, serão iniciadas ações de mitigação de tráfego. O intuito da pesquisa é evitar que o buffer do switch, e consequentemente a comunicação com o controlador, seja prejudicado. Assim como no artigo de Dhawan et al., o trabalho proposto realiza a detecção de ataque de negação de serviço, porém utiliza como meio a integração de uma ferramenta de segurança de redes convencionais, o IDS, associada ao firewall para realizar a identificação do ataque de negação de serviço e o tratamento de mitigação desse tráfego mal-intencionado, o trabalho de Dhawan et al. apenas implementa regras no controlador sem adicionais de segurança, enquanto que o nosso projeto de pesquisa identifica e atua no tratamento e estudo desse tráfego indevido.

Em uma pesquisa realizada por AVANT-GUARD [77], é tratado o gargalo da comunicação entre o plano de dados e o plano de controle, além de ser proposta uma análise e a tratativa da queda

no desempenho do plano de controle frente a uma sobrecarga de mudanças dos fluxos dentro do plano de dados. O gargalo de comunicação entre os planos de controle e dados da rede SDN facilita a possibilidade de um ataque de negação de serviço, sendo uma vulnerabilidade comumente explorada. A proposta oferecida por este trabalho de pesquisa para solucionar a questão é criar uma extensão do plano de dados que lide com mensagens que possam ser ameaças de ataques DoS. De acordo com a proposta, as mensagens de extensão são analisadas antes da conexão caso esta tenha sido analisada e considerada legítima. Após a legitimidade da mensagem de extensão, a conexão é realizada e então reportada ao plano de controle. Caso as mensagens analisadas sejam consideradas impróprias, ou seja, ilegítimas, a conexão não é realizada, sendo rejeitada no plano de dados, evitando uma alta utilização e a sobrecarga de comunicação entre o plano de dados e o plano de controle. O trabalho de pesquisa proposto detecta um possível ataque de inundação, ou seja, ataque de negação de serviços feito apenas através das regras predeterminadas ao dispositivo IDS, que, após identificação, encaminha o tráfego ao controlador SDN a fim de definir qual será a ação para o tratamento dos dados maliciosos, ao contrário da pesquisa de AVANT-GUARD, em que a conexão é realizada apenas se considerada segura, o nosso trabalho recebe o ataque e utiliza-o para estudos de seu comportamento.

Uma comparação qualitativa entre dois controladores SDN de código aberto, o OpenDaylight e o Open Network Operation System 2(ONOS), é proposta na pesquisa de Bondkovskii [78], na qual a interface Northbound dos dispositivos da rede é configurada para realizar espelhamento de porta utilizando ferramentas de benchmarking usadas para a análise comparativa dos controladores SDN, dentre os quais está o Ryu, controlador utilizado no projeto de pesquisa que propomos. O presente trabalho realiza a integração de um IDS associado ao controlador SDN através de API de integração REST API de modo a identificar e tratar o tráfego malicioso pela solução proposta, bem como descreve alguns dos principais controladores SDN e suas diferenças para conhecimento de mercado. O trabalho [78] mencionado não atua em tratativas de segurança, ao contrário da pesquisa que propomos, mas também realiza o espelhamento de porta no OVS para encaminhar o tráfego para a análise do IDS.

A elaboração de ambiente para coleta e análise de dados com o objetivo de estudar o comportamento de ataques cibernéticos foi proposto no trabalho de pesquisa de Gildásio Júnior [16]. Foi criada uma arquitetura física denominada honeyselk, onde é empregado um host hospedeiro ligado diretamente à Internet por meio de um roteador em um ambiente possuindo três redes distintas, a Internet, a honeynet e a rede de gerência, sendo esta o ponto de monitoração da solução. O fluxo de dados é controlado por regras previamente determinadas, o iptables foi utilizado, assim como também utilizamos em nosso trabalho, porém com a finalidade de prevenir ataques partindo de dentro da honeynet, enquanto no trabalho proposto o iptables é associado ao IDS para identificar e tratar o ataque DoS antes que seja ou não enviado à honeynet, identificando tráfego malicioso a partir da rede interna do cliente, e após tratativa encaminhado à honeynet ao comando da controladora SDN. A arquitetura comprovou ser capaz de coletar, analisar, visualizar e estudar ataques cibernéticos e vulnerabilidades exploradas em sistemas de

rede e ainda realizar coletas e explorar graficamente as relações entre as entidades (países, IPs, portas de origem e destino, protocolos, honeypots). Este trabalho de pesquisa associa o ambiente honeynet desenvolvido [16] para prover a análise, estudo e detalhamento dos ataques identificados.

O trabalho de pesquisa [79] propõe uma ferramenta de identificação para detecção de ataques Syn flooding em uma rede SDN, utiliza uma medida da variação da quantidade de fluxos em determinado intervalo de tempo predeterminado, também monitora as portas TCP por meio de ferramenta desenvolvida pelo pesquisador do trabalho cujo nome é FindFlows, que exibe uma lista de todos os hosts ativos em uma rede SDN. Essa análise também informa a quantidade de fluxos dos dispositivos em intervalos de tempo diferentes, a variação dos fluxos nesses intervalos e sua classificação como atacante, vítima ou usuário. A solução detecta também ataque Syn flooding em grande parte dos casos testados. A ferramenta desenvolvida atua em associação ao controlador. Assim como para este trabalho, o controlador escolhido foi o Ryu, devido a ele fornecer atualizações frequentes, suportando até a última versão do OpenFlow, o Findflows; utilizando Python como linguagem de programação, foi elaborado um script para identificação em conjunto com o Ryu. Este trabalho, da mesma forma que o trabalho citado, também utiliza o controlador SDN Ryu para tomada de decisões sobre o direcionamento do tráfego identificado pelo IDS e encaminhado ao controlador SDN através da proposta de integração de uma rede convencional existente a uma arquitetura SDN e uma solução para identificação de possíveis tráfegos maliciosos oriundos de ataques de negação de serviço, porém realizando o encaminhamento do tráfego malicioso para uma honeynet antes de realizar o bloqueio, para que possa ser estudado o comportamento do invasor e informações como IP de origem e destino.

Reza et al. [80] propõe uma solução que realiza a monitoração das conexões entre cliente e servidor cujo nome dado pelo autor da pesquisa é SLICOTS. O SLICOTS inclui regras de curto prazo, temporárias, e é utilizado como uma contramedida eficaz e eficiente para mitigar o ataque de inundação de TCP SYN em SDN. Esta solução utiliza a programação dinâmica da rede SDN para prevenir e detectar os ataques. SLICOTS é implementado no controlador, ele supervisiona as solicitações de conexão TCP em andamento e bloqueia hosts maliciosos. Na pesquisa, os SLICOTS são módulos de extensão do controlador, funcionando como um complemento ao controlador utilizado que foi o OpenDaylight. Os resultados em ambiente experimental retornam a redução do tempo de resposta na metade, garantindo a proteção proposta. Em comparação ao trabalho proposto, este projeto de pesquisa também detecta, analisa e trata ataques em redes SDN, porém trata de ataque de negação de serviço, que é identificado por meio do IDS, normalmente utilizado em redes convencionais.

A proposta [81] desenvolve uma solução em que o controlador atua como um proxy para verificar a autenticidade de um host de origem. A solução foi nomeada de OPERETTA, possui a funcionalidade de prevenir ataques do tipo TCP SYNFLOOD. O OPERETTA é um protocolo implementado no controlador que gerencia os pacotes TCP SYN recebidos. De acordo com a proposta, também é possível rejeitar solicitações de conexão não verídicas. O protocolo OPERETTA funciona em redes heterogêneas, pois pode ser implementado não apenas em um

controlador centralizado, mas também em controladores dispersos, ou seja, descentralizados disponíveis nos roteadores de acesso nas instalações dos usuários. Assim como a solução proposta neste trabalho de pesquisa, o OPERETTA foi testado utilizando o Mininet e, para este propósito, protótipos das funções relevantes do Plano de Controle foram implementados a partir do controlador POX. Conforme resultado ao longo de sua execução, o trabalho proposto [81] mostra que o OPERETTA atinge bom desempenho em termos de resiliência a ataques TCP SYNFLOOD e baixo nível de consumo de CPU e memória. A solução proposta neste trabalho implementa a solução com a utilização do controlador da rede SDN, o IDS identifica o tráfego malicioso e encaminha ao controlador centralizado, que define qual a solução mais adequada para a tratativa de mitigação desse tráfego que pode ser encaminhado ao ambiente honeynet para análise de parâmetros de ataque. O tráfego malicioso não chega a alcançar o controlador, como na proposta do OPERETTA; em nossa proposta, o IDS identifica o pacote indesejado e notifica a controladora e o administrador da rede. Henrique Fernandes [82] propõe integração do IDS a uma arquitetura SDN, utilizando o OpenFlow, e propõe uma nova forma de implementação deste segmento de solução de segurança, que foi nomeada de IDSFlow. Sua proposta apresenta uma solução para integrar um controlador SDN, com um ou mais IPS/NIDS em uma rede fisicamente distribuída. A ferramenta proposta minimiza os problemas de integração entre um IDS e o controlador SDN. O IDSFlow entrega o tráfego encaminhado até os pontos de análise. Foram utilizados algoritmos para realizar a validação da solução que comprovam a integração dos sistemas sem prejuízos ao funcionamento da rede. A proposta deste trabalho utiliza a integração das redes convencionais e definidas por software ao recurso de segurança IDS, porém não utiliza o IPS. O IDS faz apenas a identificação do tráfego não desejado e a tratativa é definida pelo controlador SDN, que pode decidir encaminhá-lo a um ambiente de estudo chamado honeynet.

Thiago Oliveira [12] propôs a utilização do IDS em uma arquitetura de redes SDN, onde são identificados ataques de detecção de intrusão originados de um host dentro da rede SDN. Esses ataques são gerados pela ferramenta hydra e identificados por meio de regras inseridas no IDS, sendo posteriormente analisados. O trabalho [12] demonstrou que o IDS não detectou mais da metade dos ataques que foram lançados pela ferramenta, gerando um alto número de falsos negativos. Assim como neste trabalho de pesquisa, Thiago utilizou o emulador Mininet para simular o ambiente SDN juntamente com o controlador Ryu, utilizou a ferramenta THC Hydra para gerar tráfego dentro do ambiente de teste, onde também aplicou o Dataset Darpa 99 para sua base de ataques com as principais características de cada intrusão para a realização dos testes, enquanto esta proposta utilizou o IPERF para gerar tráfego utilizado em nosso ambiente de testes. Os resultados de Thiago não foram totalmente satisfatórios na detecção e tratativa do ataque de negação de serviço, pois gerou alguns falsos positivos, sendo que no trabalho proposto utilizamos uma honeynet para estudos do comportamento desse tráfego identificado como malicioso, obtendo o resultado desejado: os ataques foram identificados e tratados.

O artigo de pesquisa de Wanderson Paim et al. [83] realiza uma análise de propostas de segurança recentes sobre a utilização da tecnologia SDN, tratando da sua contribuição para segurança e melhora no uso de SDN, e explica a ajuda e privacidade em Cloud Computing. O

artigo informa que uma das vantagens em utilizar a SDN em nuvem de redes de computação se refere à possibilidade de implementar serviços de segurança, tais como IDS, IPS, firewall, balanceadores de carga, proporcionando que ameaças possam ser tratadas rapidamente por meio da programação de módulos de software adicionais, ou então desenvolvendo software com técnicas autônomas que forneçam roteamento automático e adaptável com múltiplos caminhos, controle de congestionamento, ou qualquer função necessária enquanto esse trabalho de pesquisa implementa a associação de um recurso de segurança existente em redes convencionais a uma rede SDN, associando um IDS ao firewall iptables e o controlador OpenFlow para obter um maior nível de segurança. Após a identificação do tráfego malicioso, as decisões são tomadas pelo controlador, que é o ponto central de uma estrutura SDN. A análise realizada pelo artigo de Wanderson Paim et al. menciona as possíveis consequências indesejadas quando a SDN é implementada em ambientes de computação em nuvem, pois quando há uma alteração em um equipamento de uma rede tradicional, apenas o hardware alterado é afetado em SDN. A inteligência da rede está centrada no plano de controle e permite que as alterações sejam realizadas por meio de módulos de software no controlador, independentemente das modificações no plano de dados relativos aos módulos de equipamento. Isso permite atualizações e inovação mais rápidas, mas, de acordo com o artigo, também introduz preocupações com a segurança acesso não autorizado, vazamento de dados, modificação de dados, aplicativos maliciosos, negação de serviço e problemas de configuração. Este trabalho de pesquisa propõe a tratativa de um desses itens de segurança, o ataque DoS.

Em [84], foram realizados experimentos em que a configuração de um ambiente SDN para teste com a associação de recursos de segurança IDS e firewall foi criada por meio de máquinas virtuais e, assim como no trabalho que propomos, é utilizado o controlador Ryu. Trata-se de uma proposta de dinâmica de elaboração da integração desses recursos que comprovou ser capaz de identificar os ataques gerados por hosts configurados nessa estrutura. O trabalho de pesquisa que propomos também realiza a integração desses recursos, mas, além da identificação, realiza o tratamento desses ataques, seja por desvio do tráfego a um ambiente honeynet, seja por seu bloqueio.

Em seu trabalho de pesquisa, Helton Ribeiro Lustosa [85] aborda testes realizados em um ambiente honeypot montado com o intuito de avaliar a segurança em três dos controladores SDN que são mais utilizados: o OpenDaylight, o POX e o Floodlight. O ambiente para os testes foi simulado no software Mininet. A honeypot sofre ataques externos à rede, e esses pacotes são capturados por TCP Dump e nmap, havendo também ataques de SSH. Para realizar os ataques de negação de serviço, foi utilizado um programa chamado T50, e os resultados demonstraram que o controlador SDN que obteve o melhor desempenho foi o Floodlight, sendo que a identificação do tráfego malicioso não é automática. O trabalho apresentado em nossa pesquisa identifica o ataque DoS por meio de regras inseridas no IDS. Esses ataques são direcionados a uma honeynet para análise. No trabalho correlato, o ataque é realizado dentro do ambiente honeypot e não apenas direcionado para estudos, caso o pacote seja considerado malicioso, como forma de tratativa.

O artigo DoS Attack Prevention on IPS SDN Networks [17] propõe um sistema de detecção e mitigação de ataques de negação de serviço através da utilização de recursos de segurança implementados em redes convencionais e deu origem a este trabalho de pesquisa. O tráfego é identificado por regras predefinidas inseridas no IDS, notificando a controladora, a qual pode solicitar que os pacotes sejam encaminhados à rede honeynet criada para essa finalidade, ou que sejam bloqueados para sanar as preocupações de segurança introduzidas pela arquitetura SDN. O referido artigo evoluiu para este projeto de pesquisa, no qual podemos comprovar a eficiência da integração de itens de segurança de redes tradicionais a uma arquitetura SDN.

3 PROPOSTA DE ARQUITETURA

3.1 ARQUITETURA GERAL

A arquitetura proposta neste projeto de pesquisa realiza a integração dos elementos de uma rede SDN com os diferentes dispositivos que garantem a segurança em redes LAN, tais como *firewall*, IDS, controlador SDN, *switches* associados a uma *honeynet*. Essa agregação faz com que toda a rede possa reagir a ataques originados da rede Local ou até mesmo da própria Internet caso os dispositivos da rede sejam infectados e façam parte de uma botnet com o intuito de causar um ataque de negação de serviço. Esses ataques também podem ser oriundos do percurso contrário, da Internet para o perímetro de proteção da rede local, onde o tráfego malicioso será desviado para servidores *honeynet* e devidamente analisado para a identificação detalhada de seus dados, podendo ser identificada a origem do ataque, quais protocolos foram utilizados e as informações pelas quais o atacante executou esse ataque. Dessa forma, é possível a inclusão de políticas de proteção mais eficazes contra esses ataques. O tráfego malicioso também poderá ser bloqueado ou encaminhado à quarentena para observação, sendo essas possibilidades de intervenções decididas pelo administrador, que realizará a inclusão das regras na controladora SDN.

As tecnologias SDN vêm se tornando um grande aliado no combate a cyber ataques, agilizando o provisionamento de elementos de segurança para restringir o tráfego de endereços suspeitos, facilitando a operacionalização da aplicação de listas de acesso.

As grandes operadoras de telecomunicações possuem a capacidade de realizar a mitigação dos ataques de negação de serviço, bloqueando o pacote em seus roteadores de borda mais próximo da origem do ataque, impedindo que eles atravessem toda a sua rede [86].

Em termos gerais, grandes empresas de Telecomunicações possuem equipes dedicadas à segurança de seus dados, operando em regime de operações 24x7. Essas equipes são responsáveis por gerenciar o tráfego que percorre seu *backbone*, e esse monitoramento é realizado com a utilização de ferramentas que possuem o objetivo de encontrar soluções para um problema através da análise do comportamento do volume do tráfego. Ao identificar o tráfego que não possui seus parâmetros dentro dos padrões preestabelecidos destinado a determinado domínio, os operadores de segurança criam listas de acesso com o IP de origem dos ataques para mitigá-lo. Essa estratégia foi adotada para evitar ataques de negação de serviço contra grandes instituições financeiras em todo o mundo [87], entretanto, apresenta alguns problemas:

- Necessidade de equipe dedicada com especialização em segurança, aumento do custo final do produto;
- Dificuldade de distinguir, de forma proativa, um tráfego anômalo do tráfego real;
- Aplicação de listas de acesso em roteadores e *firewalls* pode ser lenta.

A identificação de ataques utilizando ferramentas automáticas de detecção leva em consideração o histórico do tráfego destinado para determinado domínio. Essa análise exige grandes recursos computacionais, tornando uma solução dispendiosa tanto em recursos computacionais quanto em ativos humanos. Manter uma equipe de segurança a postos para reagir a possíveis ataques possui um custo muito elevado, e essa solução pode inviabilizar a concorrência da operadora.

Com objetivo de separar os custos de conectividade com a Internet com os custos de segurança, operadoras de telecomunicações e provedores de distribuição de conteúdo costumam oferecer o serviço de proteção contra ataques DDoS como um serviço adicional [88].

A solução proposta neste trabalho tem como objetivo realizar a mitigação de um ataque de negação de serviço o mais próximo possível da origem, seja ela externa, seja interna à rede local, utilizando recursos open source, fazendo assim com que o tráfego malicioso não percorra toda a extensão da rede local.

3.2 ESCOLHA DOS DISPOSITIVOS DA TOPOLOGIA

A topologia da figura 3.1 apresenta a disposição dos elementos que foram utilizados na solução de rede proposta e, em sequência, o descritivo desses equipamentos.

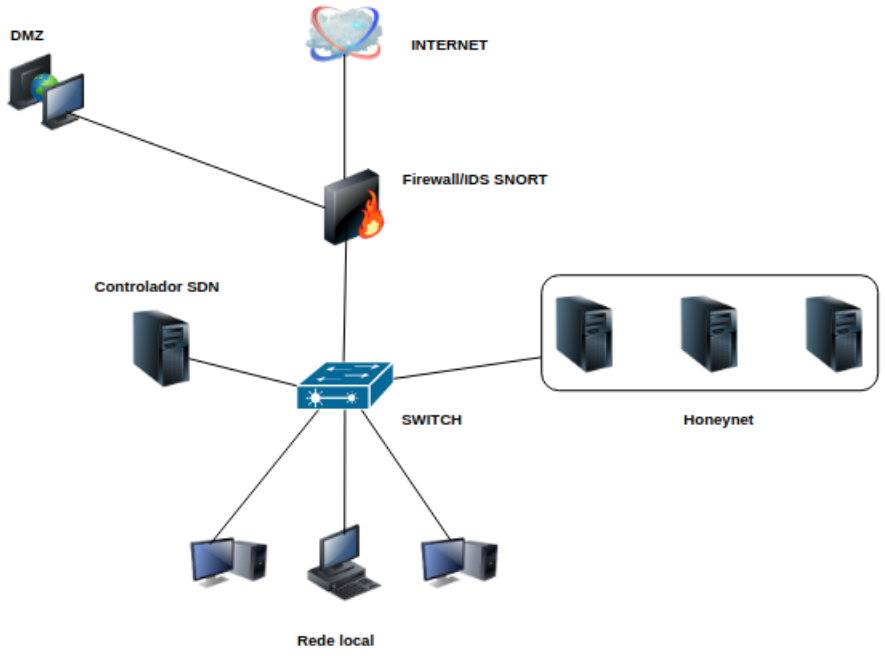

Figura 3.1: Topologia da solução proposta.
Autor

3.2.1 Switch SDN

Switch SDN, também conhecido como switch com suporte ao protocolo OpenFlow, é o dispositivo que interage com o controlador da rede. A gerência desta modalidade de rede é realizada pelo controlador SDN, que centraliza todas as funções permitindo ao seu administrador controlar os dispositivos e o tráfego em um único momento sem que seja necessário programar equipamento por equipamento. Devido à visão centralizada da rede SDN, é possível que os administradores ditem aos equipamentos que compõem a rede (switches e roteadores) como lidar com o tráfego, ou seja, configurando essa estrutura por meio de um comando centralizado para todos os dispositivos que se deseje alcançar de uma única vez. O controlador gerencia o switch, insere e exclui fluxos que suportam um subconjunto de critérios de correspondência e ação do OpenFlow 1.3 e 1.0, e o switch se conecta ao controlador usando a porta de gerenciamento [89]. Existem diversos softwares switch OpenFlow, entre os mais utilizados está o Open vSwitch (OVS). Esse software possui não apenas o modo usuário, mas tamém um modo kernel, sendo propício até para ambientes de produção [27].

O Open vSwitch foi utilizado na arquitetura proposta para realizar a interconexão dos elementos

da rede, onde o controlador SDN alimenta a tabela de fluxo do OVS por meio do protocolo OpenFlow. Essas tabelas são elaboradas e distribuídas pelo controlador, que conhece todas as VMs, os dispositivos da rede e também as conexões realizadas pelo switch. O controlador informa o que deve ser direcionado ou bloqueado. O OVS possui fácil programação, realizando as conexões com os dispositivos da rede. Conforme a figura 3.2, o controlador está associado ao OVS utilizando o IP 192.168.92.130 na porta de gerência 6633, onde foi utilizado o comando:

"sudo ovs-vsctl set-controller br0 tcp:192.168.92.130:6633"

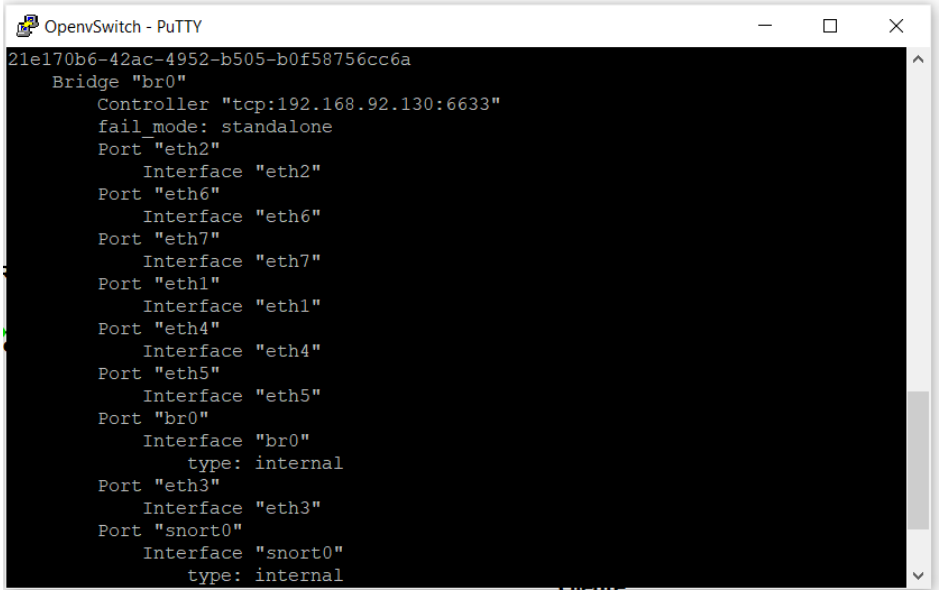

Figura 3.2: Associação do Ryu ao OVS.
Autor

3.2.2 Controlador SDN

Para escolha do Controlador SDN, foi configurado um ambiente de testes para avaliar quais controladores seriam os mais adequados para o ambiente proposto.

Para realizar a escolha do controlador foram executados testes de desempenho utilizando Ryu, OpenDaylight e Floodlight dentro do ambiente proposto, medindo métricas chave que influenciam diretamente o desempenho de aplicações como latência, perda de pacotes, variação do atraso (Jitter) e trafego. Foram considerados para a escolha da controladora, a pesquisa acadêmica de trabalhos semelhantes.O trabalho correlato de Helton Ribeiro Lustosa [85] também realiza a comparação entre três controladores com o intuito de avaliar suas funcionalidades e possíveis vulnerabilidades, sendo eles o POX, o OpenDaylight e o Floodlight.

O ambiente de testes para a avaliação dos controladores foi elaborado no simulador GNS3, onde foi possível montar uma estrutura utilizando o GNS3 VM como servidor. Dentro desta VM é possível executar outras máquinas virtuais, facilitando assim o processo de elaboração de redes complexas.

Foi criada uma VM com sistema operacional Debian 10 minimal para instalação do controlador Ryu e um QEMU utilizando Ubuntu server para a instalação do OpenDaylight. Os dois sistemas operacionais foram escolhidos devido a consumirem poucos recursos computacionais. O controlador Floodlight possui, em seu site oficial, uma VM Floodlight Linux pré-configurada, não sendo necessário realizar sua instalação ou integração com o Mininet e o OVS, pois a VM inclui esses itens. Para o OpenDaylight, a interface é realizada no navegador da máquina cliente que compõe a topologia de testes. o Mininet foi utilizado para emular a rede SDN para o controlador Ryu, uma nuvem NAT para que possa ser realizada entrada e saída de pacotes pela internet, o Open vSwitch em uma máquina Debian 10 minimal contendo o firewall e o IDS.

A versão utilizada na instalação para testes do controlador OpenDaylight foi a distribution-karaf-0.5.2-Boron-SR2, escolhida por se tratar de uma versão mais estável. Para utilizar a interface gráfica, foi necessário instalar o módulo de "odl-dlux-all"[42], que se trata de uma UI WEB para usuário. Também foram adicionados os módulos para configuração da rede, como pode ser observado nas figuras 3.3 e 3.4.

- *odl-restconf: Permite acesso à API RESTCONF;*
- *odl-l2switch-switch: Oferece funcionalidade de rede semelhante a um switch Ethernet;*
- *odl-mdsal-apidocs: Permite acesso à API Yang;*
- *odl-dlux-all: interface gráfica do usuário OpenDaylight.*

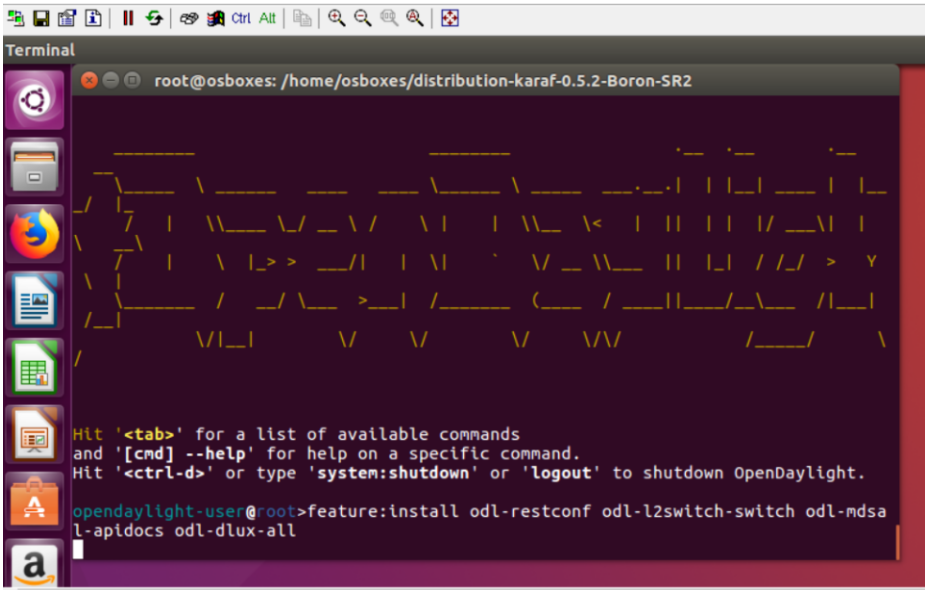

Figura 3.3: Execução do controlador OpenDaylight
Autor

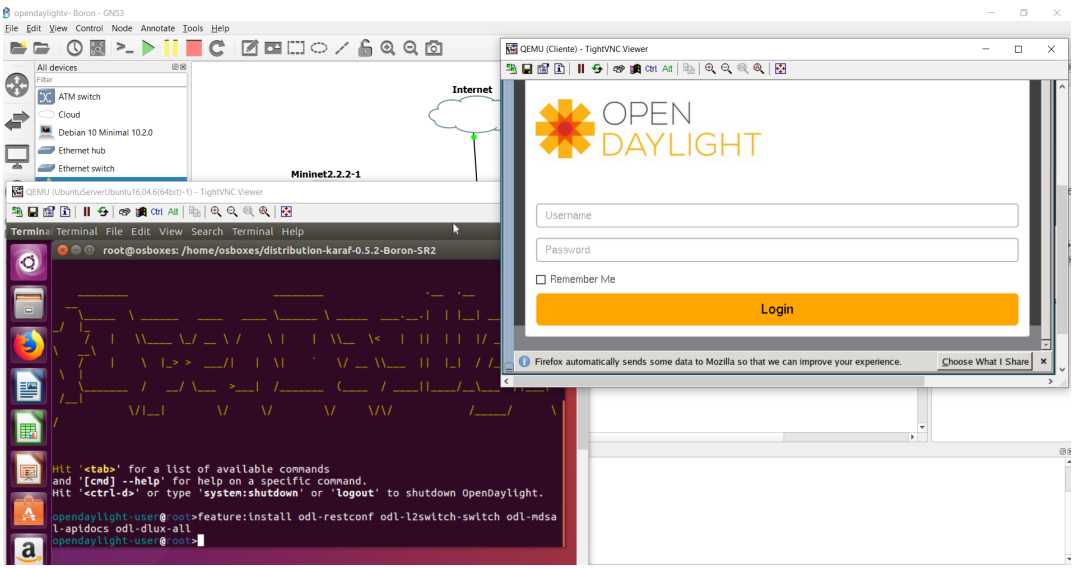

Figura 3.4: Exibição do OpenDaylight no navegador porta 8080
Autor

O controlador Floodlight foi instalado a partir da VM disponibilizada no site oficial, pré-configurada com as informações básicas do OVS e do emulador Mininet. A VM traz as informações básicas e exemplos de configuração demonstrados nas figuras 3.5 e 3.6, o que tornou mais prática a pesquisa realizada para execução dos testes com este controlador.

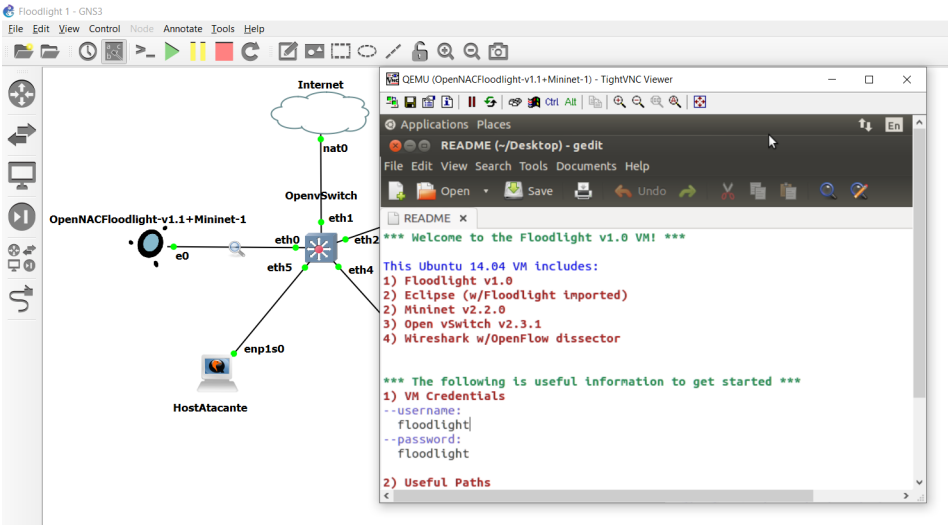

Figura 3.5: VM Floodlight
Autor

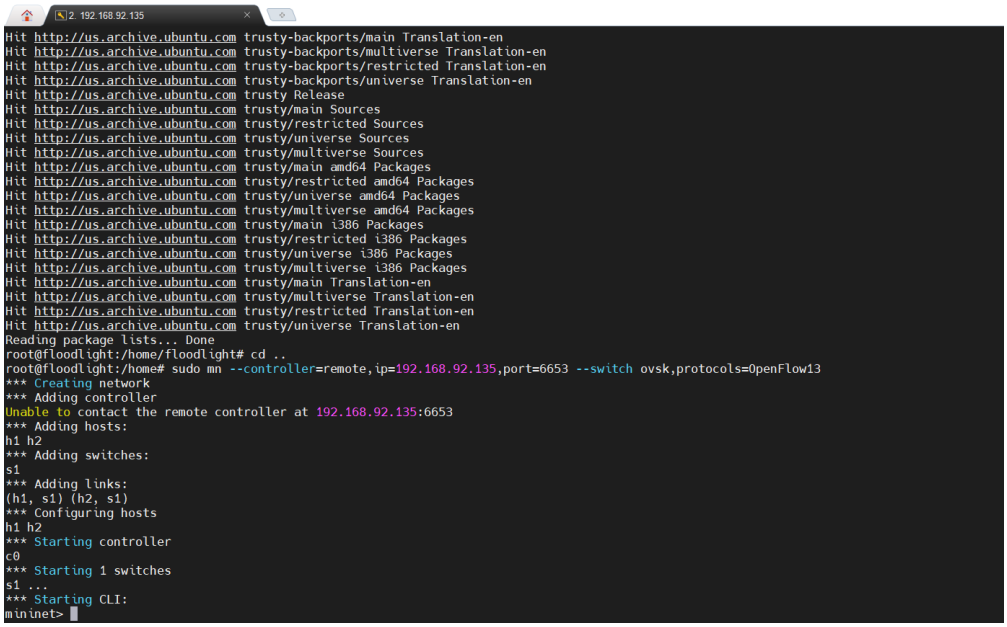

Figura 3.6: VM Floodlight com Mininet em execução
Autor

Um dos pré-requisitos para instalar o controlador Ryu é a execução do Python e a instalação de alguns módulos solicitados em seu tutorial de instalação e outros mais que foram requisitados no decorrer do processo, conforme é demonstrado nos comandos da figura 3.7. Esses módulos foram adquiridos no repositório do Python [90].

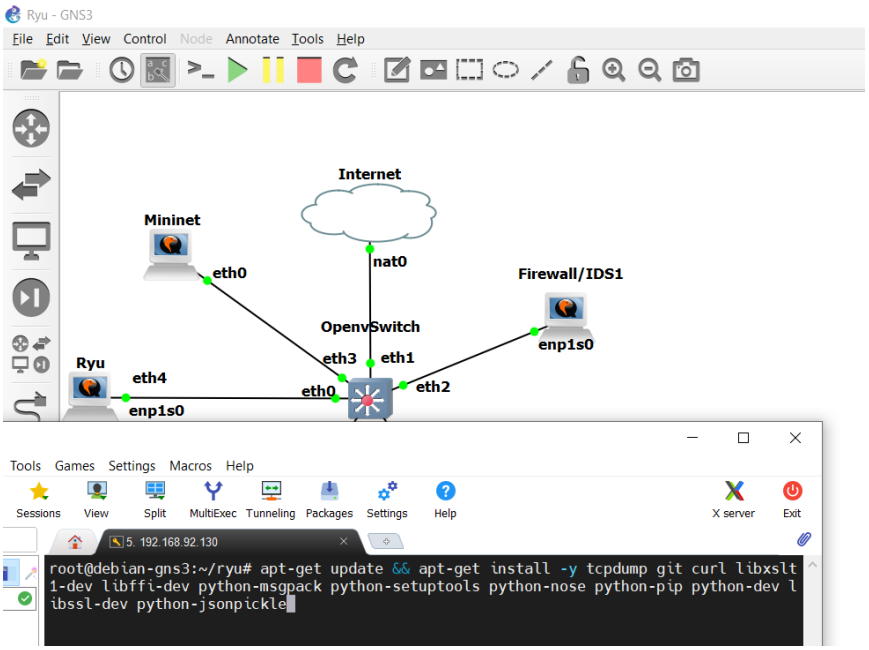

Figura 3.7: Módulos pré-requisitos de instalação do controlador Ryu
Autor

- *Testes OpenDaylight:*

O tempo médio de resposta foi medido por meio de comando padrão "ping", com pacotes ICMP de tamanho fixo de 64 bytes. A figura 3.8 demonstra que o RTT médio obtido durante o intervalo de testes foi de 3.25 ms, chegando ao máximo de 8 ms, sem apresentar perdas de pacotes porem longe de medições constantes e a figura 3.9 demonstra os testes realizados para aferir as métricas de variação de atraso (Jitter) e perda de pacotes. A variação do atraso durante os testes ficou em 0.379 ms quando foi gerado com o software iperf3 um fluxo de dados de 1.05 megabits por segundo.

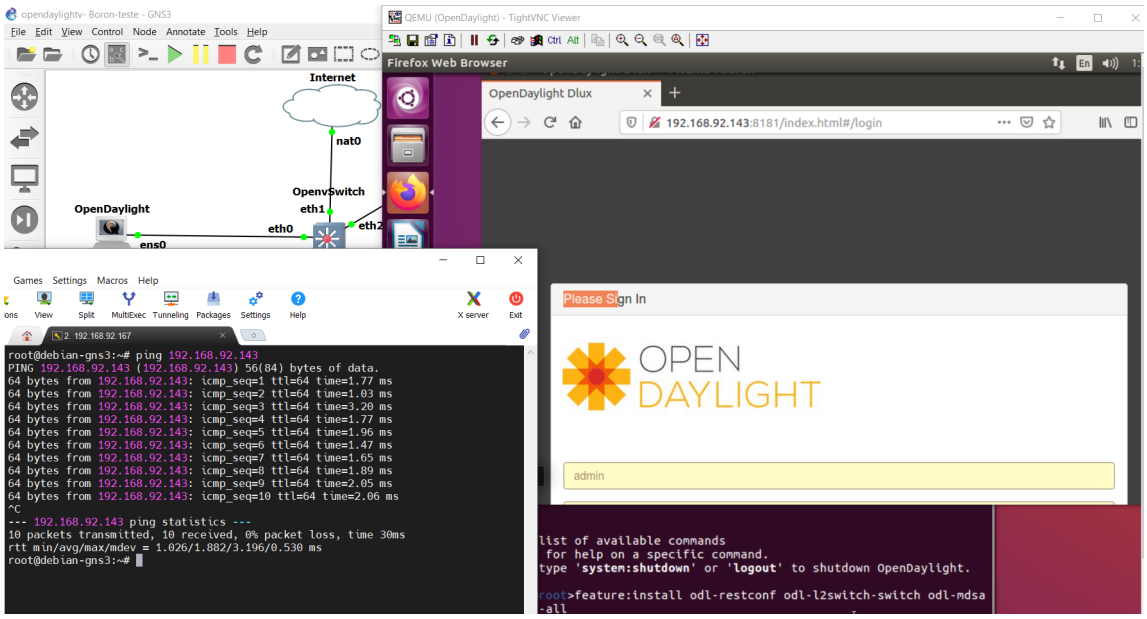

Figura 3.8: Latência do OpenDaylight
Autor

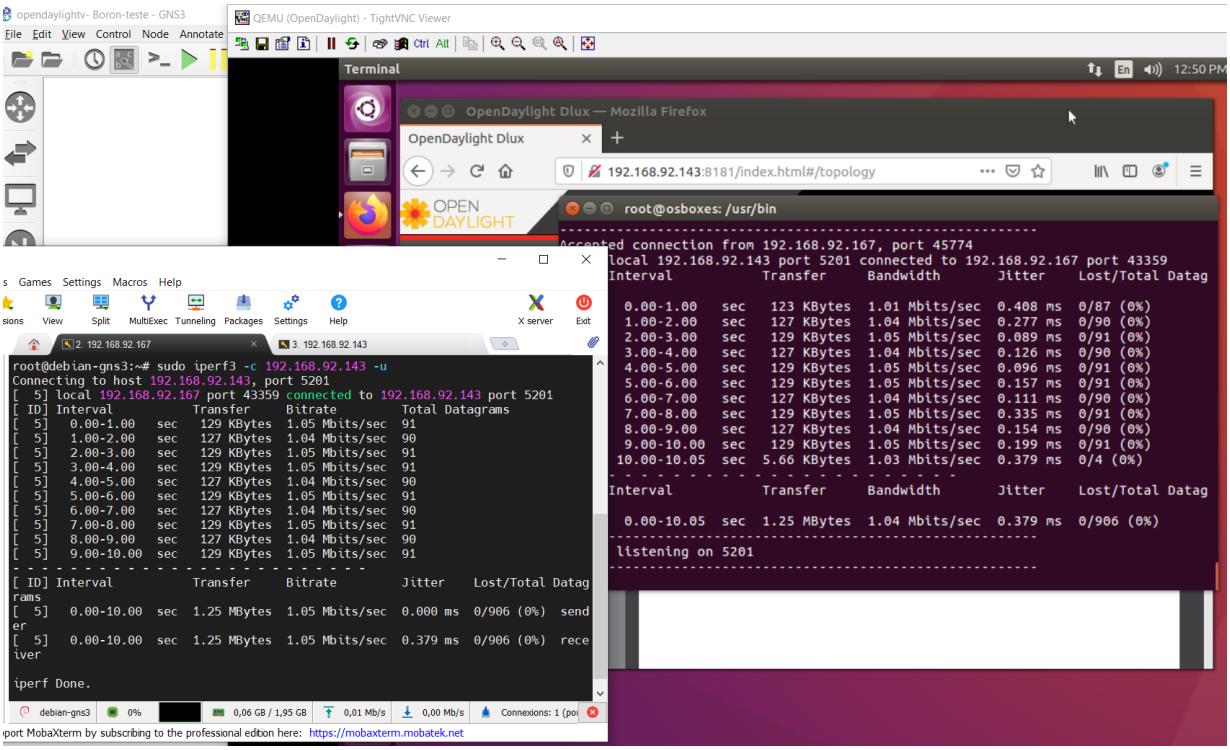

Figura 3.9: Métrica de Jitter e perda de pacote do OpenDaylight
Autor

- *Testes Floodlight:*

Os testes realizados no controlador Floodlight representados nas figuras 3.10 e 3.11 demonstraram não haver variações significativas nos resultados das métricas mensuradas em relação ao controlador OpenDaylight. Após realizar a elaboração do cenário foram realizados testes de "ping"utilizando pacotes fixos de 64 bytes. O RTT mínimo e médio obtido durante os testes foi de 1.026 e 1.882 ms, com picos de 3.196 ms. Não ocorreram perdas de pacotes durante a execução.

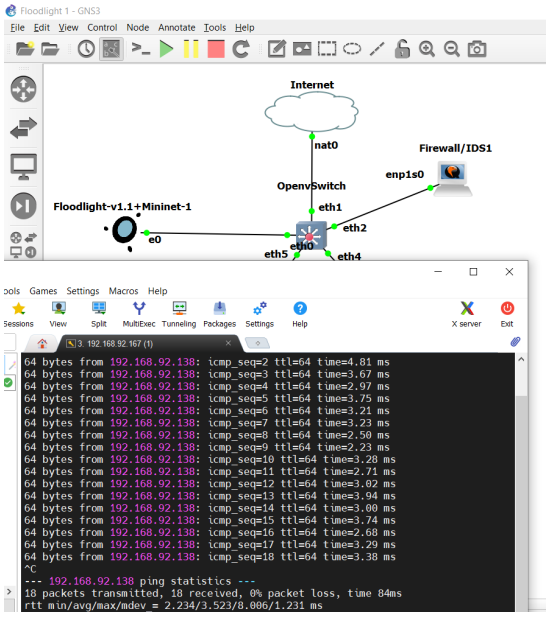

Figura 3.10: Latência do Floodlight
Autor

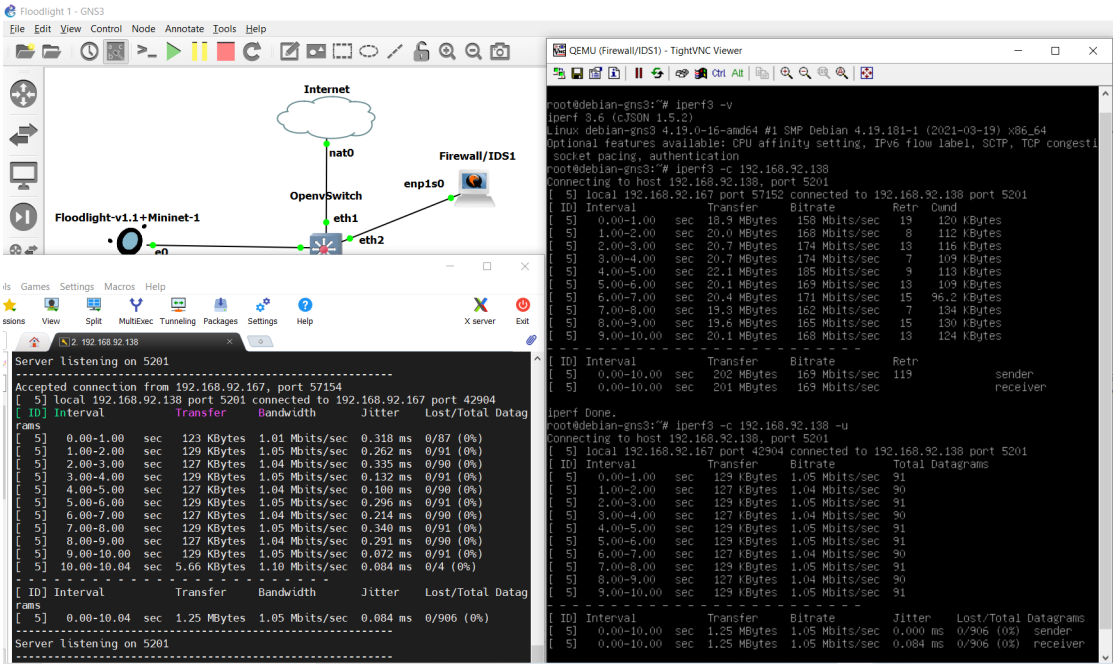

Figura 3.11: Métrica de Jitter e perda de pacote do Floodlight
Autor

- *Testes Ryu:*

Um dos principais benefícios das redes SDN é permitir a separação do plano de dados e do plano de controle nos equipamentos de rede. Dessa forma, melhorias podem ser incluídas constantemente no plano de controle e podem ser repassadas para todos os equipamentos com suporte a OpenFlow, independentemente do fabricante. A controladora escolhida para essa solução foi o Ryu [35], pela facilidade de configurar novas funcionalidades, documentação

ampla e possibilidade de implementar a solução em ambientes utilizando Mininet. O Ryu também permite integração utilizando chamadas RESTFull, permitindo a integração com Snort de forma simples.

Os testes realizados no controlador Ryu, demonstrados nas figuras 3.12 e 3.13, evidenciam que, em relação aos demais controladores, o Ryu possui menor ocupação de largura de banda, menor latência e variação do atraso, ou seja, jitter. Não houve perda de pacotes nos testes realizados em nenhum dos controladores.

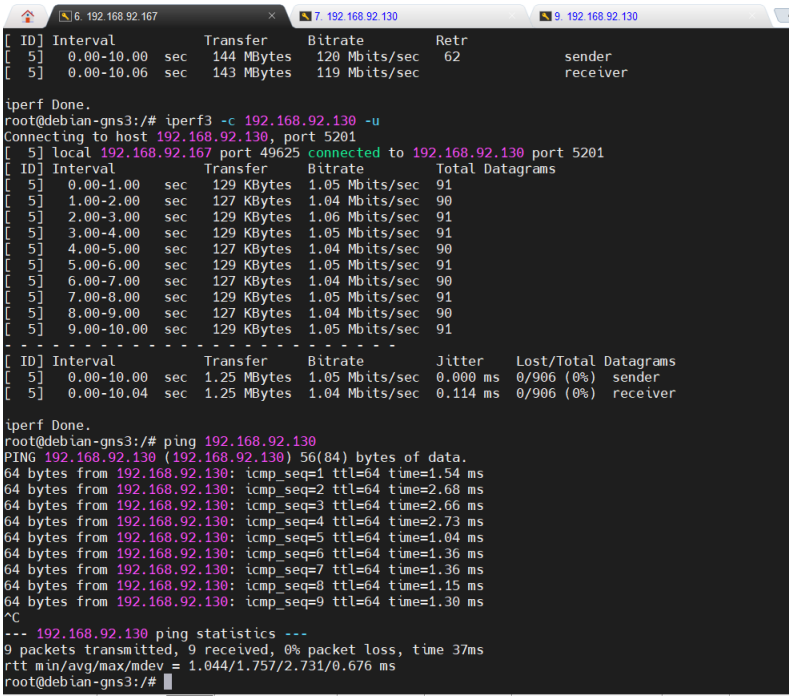

Figura 3.12: Latência do Ryu
Autor

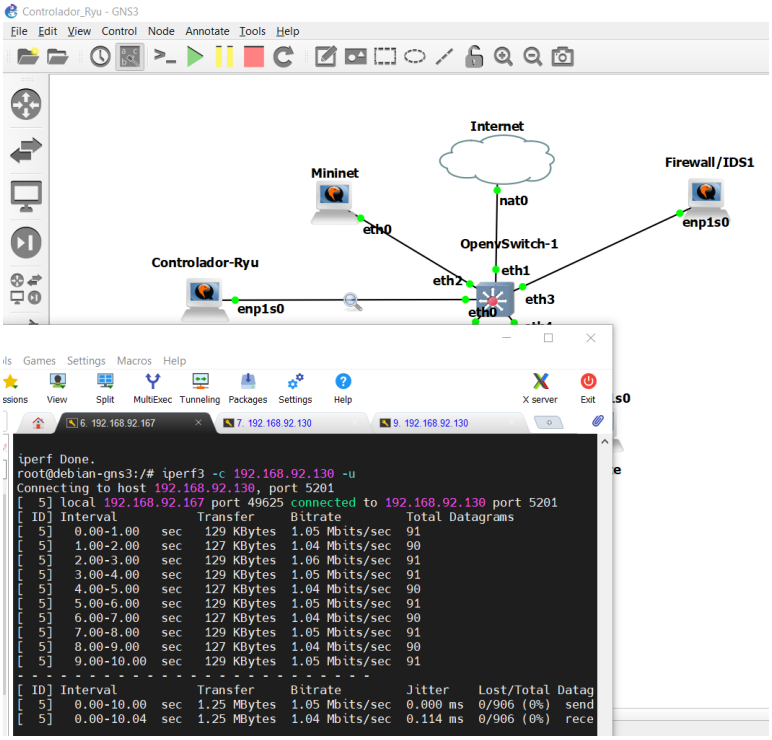

Figura 3.13: Métrica de Jitter e perda de pacote do Ryu
Autor

O comando "iperf3 -c 192.168.92.143 -f k"foi utilizado para medir a largura de banda nos três controladores.

Foram realizados testes em porta específica utilizada por cada um dos controladores como descrito no trabalho correlato [85], com o objetivo de realizar métricas de comparação entre os resultados obtidos expostos na tabela 3.1.

Tabela 3.1: Resumo dos testes realizados por meio de ferramentas para medir parâmetros de utilização de rede.

Controlador	Largura de banda	RTT	Jitter	Perda de pacotes
OpenDaylight	49.0 KBytes/sec	30ms	0.379 ms	0
Floodlight	56.7 Kbits/sec	84ms	0.693 ms	0
Ryu	60.2 Kbits/sec	37ms	0.114 ms	0

Autor

O trabalho correlato [85] utiliza Bash script, de autoria de Greg Bledsoe, para monitorar as atividades da rede e realizar possíveis intervenções. Optamos por utilizar em nossa proposta o Snort, pois se trata de ferramenta de rede robusta baseada em assinaturas que atua de forma mais completa comparada aos testes realizados e exemplificados no trabalho mencionado e a outros demais scripts prontos encontrados gratuitamente na Internet. O Snort pode ser utilizado como sniffer e log de pacotes, detecção e prevenção de intrusão, sendo a integração do controlador Ryu ao Snort nativa e exposta em sua documentação oficial [91].

O controlador utilizado nesta proposta é o Ryu, uma estrutura de rede definida por software

baseada em componentes. O Ryu fornece componentes de software com APIs bem definidas que tornam mais fácil aos desenvolvedores criarem novos aplicativos de gerenciamento e controle de rede, além de suportar vários protocolos para gerenciar dispositivos de rede, como OpenFlow, Netconf, OF-config etc. Sobre o OpenFlow, o Ryu suporta totalmente 1.0, 1.2, 1.3, 1.4, 1.5 e extensões Nicira [35]. O Ryu utiliza a linguagem Python, fornece a funcionalidade de coleta de informações de dispositivos de rede convencionais utilizados em redes tradicionais e para dispositivos que possuem suporte ao protocolo OpenFlow habilitados, possibilitando o gerenciamento dos dispositivos nas duas tecnologias [37], sendo este um ponto crucial ao desenvolvimento da proposta estudada neste projeto.

O controlador Ryu está na camada de controle, trata-se de um software que proporciona funcionalidades simples e avançadas. A última camada contém os equipamentos de rede, a exemplo dos switches [40]. A integração dessas camadas é realizada por meio de APIs.

3.2.3 Honeynet

Uma honeynet é uma rede planejada para ser utilizada em estudos de comportamentos do tráfego e que, para isso, possui mecanismos de controle que podem ser utilizados para prevenção de ataques de negação de serviço. As honeynets possibilitam que seja realizado um estudo do comportamento do ataque malicioso. Para construir com sucesso uma honeynet, é preciso tratar dois requisitos críticos: controle e captura de dados. Qualquer falha nesses requisitos implica uma falha geral [74]. As honeynets são compostas por todos os dispositivos de redes tradicionais, podendo ser implementadas de forma física ou virtual. As honeynets virtuais funcionam baseadas em um grupo de sistemas virtuais, ou seja, emuladores de sistemas operacionais, criando um ambiente de produção virtual que não possui as desvantagens da estrutura física convencional, que apresenta um custo elevado para sua implementação. Na solução proposta, foram utilizadas máquinas virtuais com Kally Linux. Para aplicar corretamente listas de acesso restringindo a comunicação, seja de dispositivos internos à rede participantes de uma botnet, seja de dispositivos infectados por vírus, é essencial que se obtenha maiores informações sobre o ataque, como endereço de destino do ataque, porta, protocolo, endereço IP da controladora, entre outros. Mediante da captura de dados, é possível coletar todas as atividades que ocorrem dentro da rede honeynet. Quanto maior o número de camadas (métodos de captura), maior a possibilidade de novos ataques serem detectados. Na arquitetura proposta neste trabalho, o software detector de intrusões IDS Snort é configurado com regras atualizadas para controle do tráfego [16].

3.2.4 Integração Firewall / Snort

A integração do firewall iptables ao IDS Snort é realizada por meio da API fwsnort [92]. Inicialmente, foi desenvolvida uma API própria para realizar essa integração, porém foram identificados diversos softwares livres existentes que atuavam de maneiras mais eficazes nessa

funcionalidade. A API fwsnort realiza a análise dos arquivos detectados pelo Snort elaborando um conjunto de regras iptables que são utilizadas na identificação do tráfego indesejado. O fwsnort utiliza o módulo de correspondência de string iptables para detectar ataques em nível de aplicativo [92]. Pode-se utilizar linhas de comando para restrição de pacotes, os quais são bloqueados no firewall pelas regras de iptables. Dessa forma, os pacotes maliciosos não chegam ao IDS e, consequentemente, não são inclusos na tabela de regras da API. O fwsnort constrói um conjunto de regras iptables que não apenas registra ataques, mas também descarta pacotes e redefine conexões [92]. Essa integração foi utilizada para realizar o bloqueio de ataques com origem na rede externa. O foco deste trabalho está em identificar ataques originados na rede local, porém ao longo de seu desenvolvimento foi identificada a oportunidade de melhorar a segurança do perímetro da rede contra ataques externos também com a integração do IDS ao firewall, montando assim uma solução de IPS.

A integração do Snort ao iptables é simples e está prevista em sua documentação oficial [92], sendo realizada por meio do comando executado na figura 3.14, onde pode ser notada a integração do firewall mediante utilização da API fwsnort na figura 3.15.

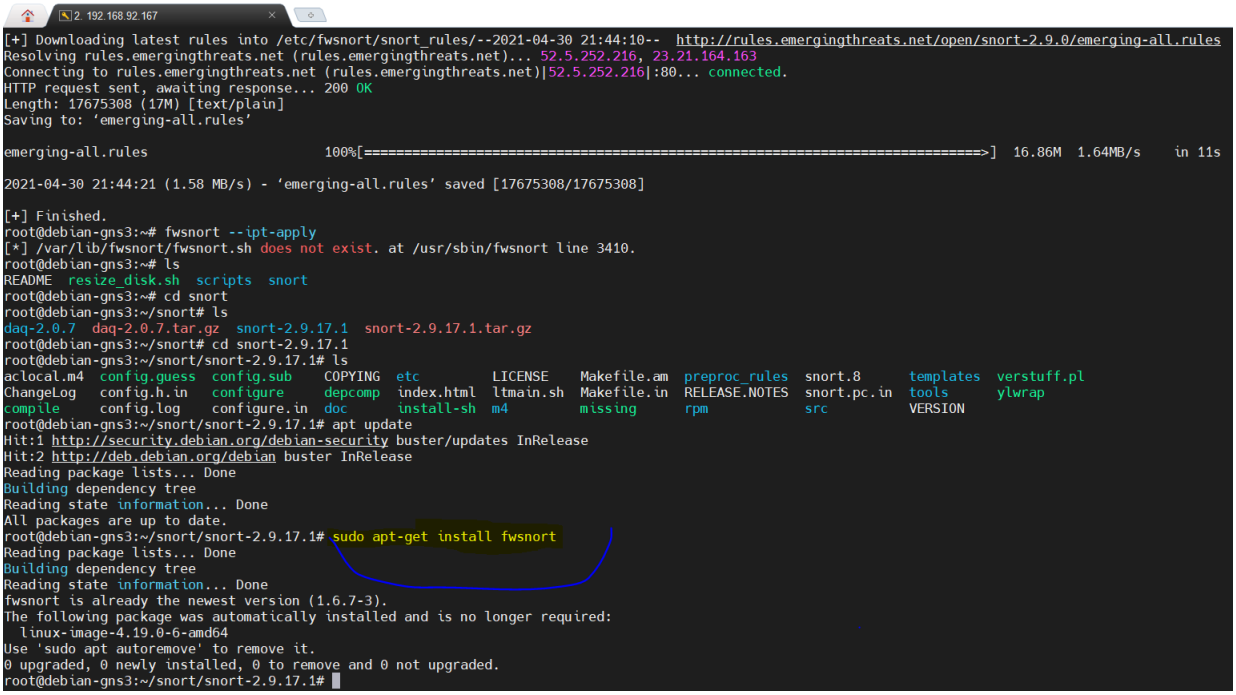

Figura 3.14: Instalação do fwsnort.
Autor .

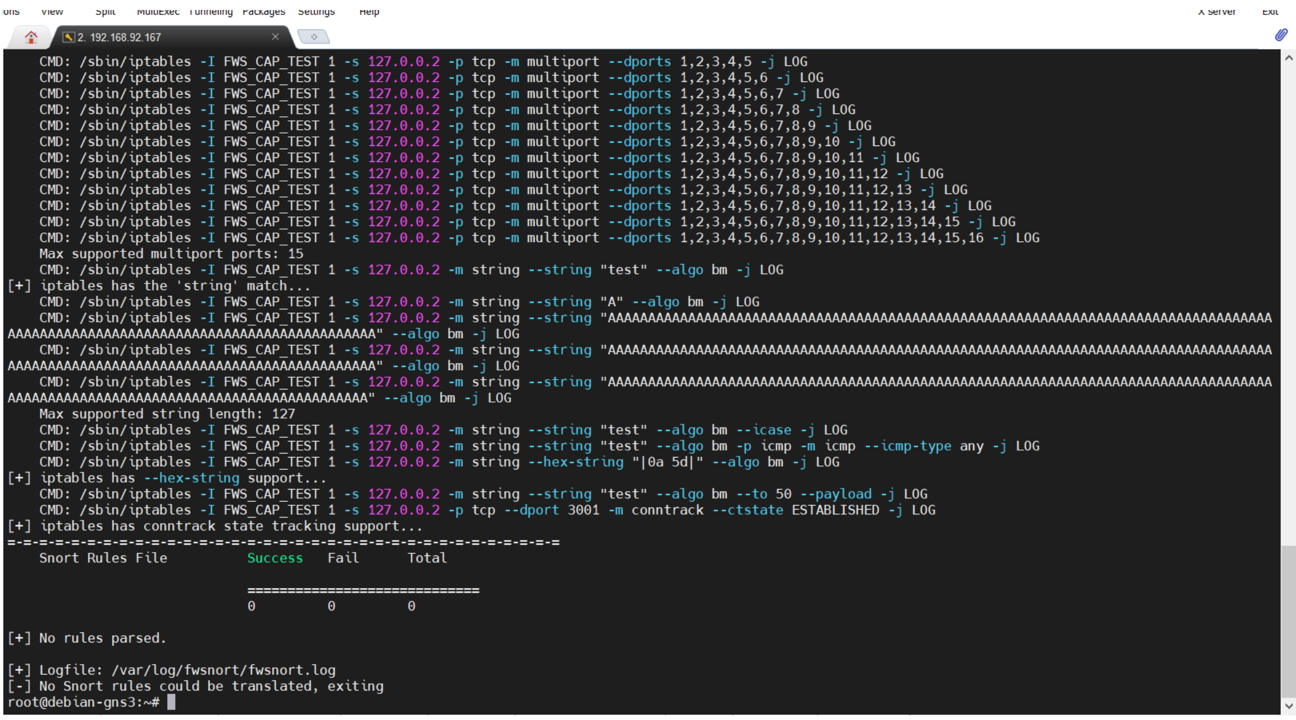

Figura 3.15: Integração entre *firewall* e IDS utilizando fwsnort.
Autor

3.2.5 Integração Controlador Ryu / Snort

Um fator importante para a escolha do Ryu foi sua integração nativa com o IDS Snort. De acordo com sua documentação específica [35], ele dispõe de duas formas de integração:

- *a primeira forma é em uma máquina executando o IDS e o controlador, o que torna o experimento mais simples;*
- *a segunda forma, quando o controlador está em uma máquina e o IDS em outra.*

As regras necessárias para identificação do tráfego anômalo são configuradas diretamente no Snort, que é o responsável por encaminhar os dados ao controlador Ryu quando houver alguma situação que divirja aos parâmetros desejados. O Snort deixa então de se comportar como IDS, apenas identificando os eventos de falha, para se tornar um IPS, realizando também o controle de intrusão, acionando o controlador para que ele bloqueie o tráfego anômalo ou realize uma tomada decisão.

Os programas para a controladora Ryu podem ser escritos utilizando Python 3. A princípio, foi desenvolvido um script que realiza as configurações na Controller a partir de eventos de anormalidade percebidos pelo Snort através de chamadas REST, porém foi observado que a API fwsnort é mais eficiente para a análise dos arquivos detectados pelo Snort, pois elabora as regras iptables para serem utilizadas na identificação do tráfego indesejado. Dessa forma, decidimos por utilizar essa ferramenta free.

Ao detectar um tráfego malicioso, o Snort gera um evento de alerta informando o IP de origem que gerou o ataque, um script realiza uma chamada REST para a controladora, contendo o endereço IP de origem e protocolo que deve ser direcionado para a honeynet. Imediatamente é gerado um alerta para o administrador, notificando-o que foi detectado um evento atípico na rede. O IP de origem fica impedido de transmitir pacotes por um intervalo determinado pelo administrador. A figura 3.16 monstra o Ryu e o Snort integrados.

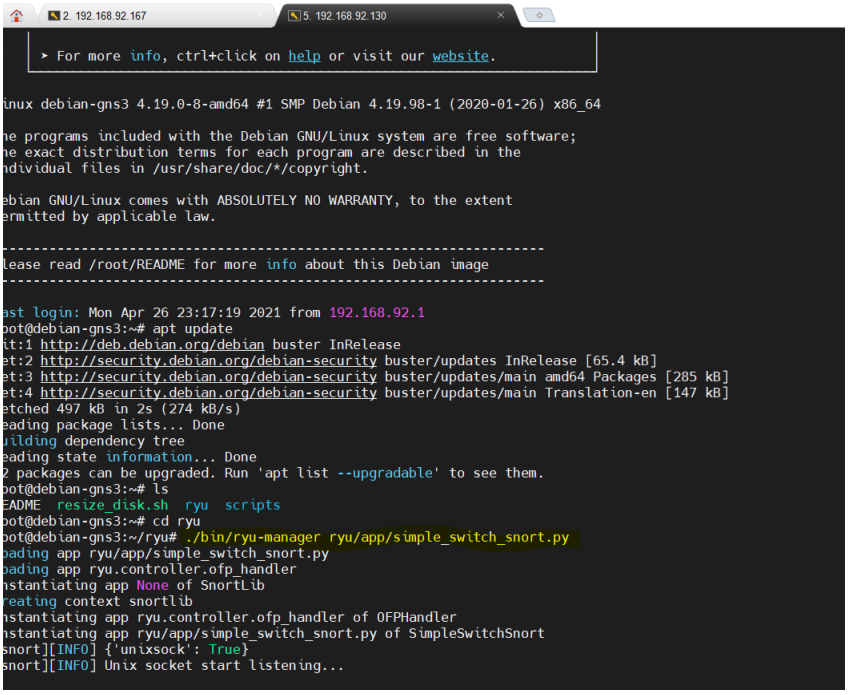

Figura 3.16: Snort em execução no controlador Ryu
Autor

3.2.6 Ambiente de simulação

Para a implementação do ambiente de testes, foi utilizado o GNS3 para facilitar a interconexão das diferentes máquinas virtuais utilizadas. O GNS3 utiliza diversos softwares para virtualização das VMs como o docker e o QEMU, que foram utilizados neste trabalho. Os testes foram executados em um notebook Intel com processador I5, 16 GB de memória RAM, SSD de 128 GB. As máquinas virtuais foram montadas com as seguintes configurações:

- *Máquina virtual Debian 10 mininmal com 2GB de memória RAM e um núcleo de processamento com o Snort;*
- *Máquina virtual com Kali Linux rodando a honeynet na versão 1.8;*
- *Máquina virtual contendo o Mininet e o Ryu utilizando 2GB de memória RAM;*
- *VM que foi utilizada para gerar o tráfego HTTP, utilizando o sistema operacional Linux Debian 10 mininmal;*

- *Máquina Virtual emulada no QEMU com o Open Virtual Switch na versão 2.4.0.*

3.3 FLUXOGRAMA DE FUNCIONAMENTO DO PROGRAMA PARA INTEGRAÇÃO

A figura 3.17 demonstra a solução por meio do fluxograma.

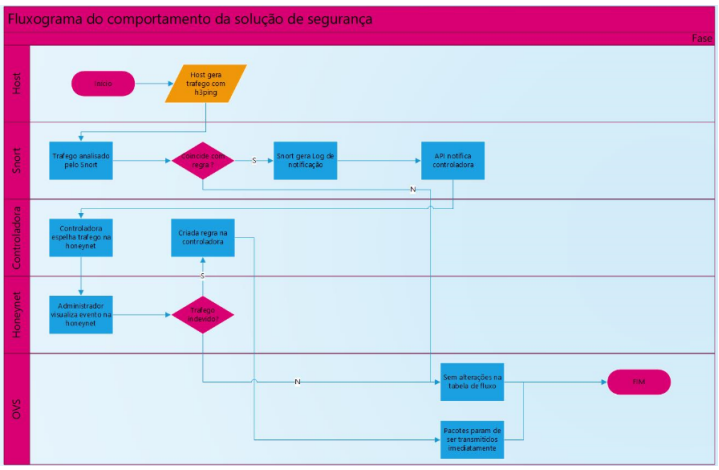

Figura 3.17: Fluxograma
Autor

O host atacante gera o tráfego indevido utilizando o aplicativo de terminal hping3 a partir do comando "sudo hping3 -S –flood -V -p 80 192.168.92.130". O IDS está posicionado na saída da rede LAN, monitorando o tráfego direcionado para a WAN e DMZ, bem como o tráfego encaminhado à controladora. Caso as informações não se enquadrem nas regras definidas no Snort, os ataques do tipo DoS originários da rede local e direcionados ao servidores externos são facilmente detectados e encaminhados para a rede honeynet. O IDS notifica o controlador local SDN, informando o IP de origem da VM que está gerando o tráfego indevido, sua porta e o protocolo do host que está promovendo o ataque. Não sendo identificado tráfego que não esteja de acordo com as regras das "rules"do Snort, os pacotes seguem seu fluxo.

Em resumo, com base nas informações fornecidas pelo IDS, o controlador SDN toma as seguintes decisões:

- *Direciona o tráfego malicioso identificado pelo IDS para um honeypot. Dessa forma, o sistema registrará todas as atividades do atacante e as enviará para o administrador, fornecendo pistas do que o atacante pretende conseguir;*
- *O Administrador analisará se o fluxo de dados que foi notificado se trata de um ataque, solicitando o bloqueio do fluxo diretamente na controladora;*
- *Caso não seja identificada nenhuma falha, o tráfego continuará a ser enviado normalmente.*

Utilizando os dados obtidos nos logs do IDS e no *honeypot*, o administrador da rede poderá confirmar se realmente se trata de um tráfego indevido e, caso positivo, realizar as correções necessárias no *host*. Os componentes utilizados nesta proposta serão detalhados ao longo desta seção.

Ataques que possuem como origem a Internet, destinados ao ambiente de servidores da rede local também pode ser identificados pelo IDS Snort. Neste caso, o tráfego malicioso será direcionado para a rede *honeynet* dando ao atacante a falsa sensação que está sendo bem-sucedido, pois todo o seu tráfego será analisado em um ambiente de estudo.

3.3.1 Honeynet utilizada na solução

O ambiente *honeynet* utilizado como modelo nesta pesquisa foi desenvolvido no Laboratório de Tecnologias da Tomada de Decisão – LATITUDE/UNB [16]. Este ambiente é composto por três *honeypots* configurados com os serviços de DNS, FTP e HTTP, para onde é direcionado o tráfego malicioso a fim de ser realizada a anáĺise desse conteúdo. Dessa forma, podemos identificar mais informações sobre o ataque, como endereço IP de origem e destino, quantidade de pacotes enviados, tendo assim a percepção do comportamento do ataque além de equipamentos que podem estar infectados com vírus e que fazem parte de uma botnet [93]. Para o trabalho, a *honeynet* foi configurada com máquinas virtuais, utilizando a distribuição Kali Linux.

3.3.2 Mininet e Open Virtual Switch

Os *softwares* Mininet e Open Virtual Switch foram utilizados para montar um ambiente de rede de testes, composto por uma topologia linear com um *switch* e três clientes. Os ataques de negação de serviço foram gerados a partir da VM, utilizando ferramentas hping3 [94]. Os ataques foram direcionados para IPs dentro da rede interna do laboratório, um ambiente de testes isolado, desenvolvido para tal finalidade.

Diferentemente da ferramenta ping, que transmite apenas pacotes ICMP, o hping3 transmite pacotes TCP, UDP, ICMP, protocolos RAW IP e realiza *traceroute*, identificando o caminho que os pacotes percorreram. O hping3 pode ser utilizado para gerar um grande fluxo de dados para servidores web, DNS ou qualquer aplicação que opere na pilha TCP/IP, sendo frequentemente utilizado para testes de penetração e identificação de vulnerabilidades [94].

3.3.3 Comunicação entre Open Virtual Switch e controladora

Conforme definição [95], o Open vSwitch permite a criação de um *switch* virtual, com suporte a protocolos de camada dois do modelo OSI como *Spanning Tree*, LACP, RSPAN, podendo interagir com *switches* reais, montando uma arquitetura híbrida de rede. A configuração do *switch* pode ser feita utilizando *comand line* interface ou a partir do protocolo OpenFlow, que é o foco deste trabalho. Antes de realizar o encaminhamento de um pacote, o *switch* pergunta para a

controladora como ele deve ser encaminhado, montando em sua memória local uma tabela de encaminhamento de pacotes (flows). Os próximos pacotes que combinam com as regras descritas na tabela são encaminhados imediatamente pelo switch, sem a necessidade de nova consulta à controladora, aumentando assim a performance no encaminhamento. O switch pode fazer uma nova consulta à controladora, verificando se há uma nova regra e se as regras vigentes continuam válidas.

O switch com suporte a OpenFlow faz a contabilização da quantidade de pacotes que combinaram com determinada regra, conforme pode ser visto na figura 3.18.

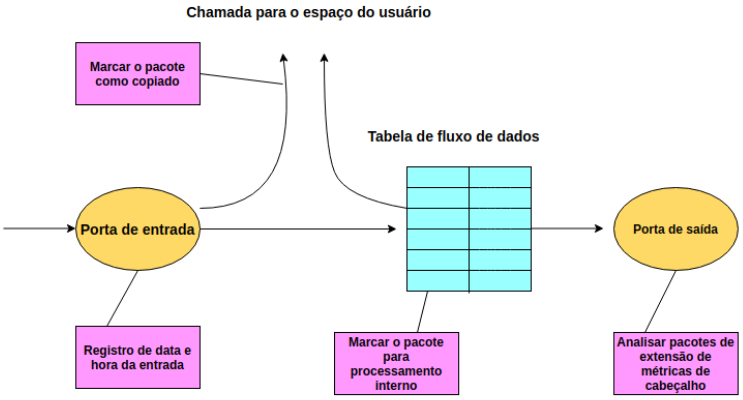

Figura 3.18: Comunicação OpenFlow.
Adaptado de [96]

Novas funcionalidades podem ser incluídas sem a necessidade de adquirir novos equipamentos ou fazer upgrade de firmware. Os switches SDN podem conviver com switches convencionais, pois funcionalidades como Spanning Tree, Link Aggregation, Port Mirroring podem ser facilmente adicionadas a partir de softwares incluídos na controladora.

No teste realizado, foi configurado o intervalo para consultas às novas regras a cada sessenta segundos. Esse tempo foi escolhido visando a reduzir a quantidade de consultas do switch para a controladora, fazendo com que novas regras possam ser incluídas em um intervalo curto de tempo.

4 RESULTADOS

4.1 ARQUITETURA UTILIZADA

A prova de conceito foi realizada em um ambiente controlado, onde é possível realizar testes sem afetar as redes de produção. Outro benefício é a possibilidade de realizar uma prova de conceito sem a necessidade de comprar switches SDN e appliances de segurança. A topologia de rede foi montada conforme a imagem 4.1, utilizando máquinas virtuais com os softwares QEMU e VM. Foi utilizada a versão do GNS3 2.2.17 para configuração do ambiente. O GNS3 permite a elaboração de redes complexas, virtualizando switches, roteadores, firewalls e até mesmo servidores. Grandes empresas como Cisco, HP, Dell e Fortinet permitem que seus produtos de rede e segurança sejam emulados no GNS3, possibilitando assim a elaboração de ambientes complexos e mais próximos da realidade.

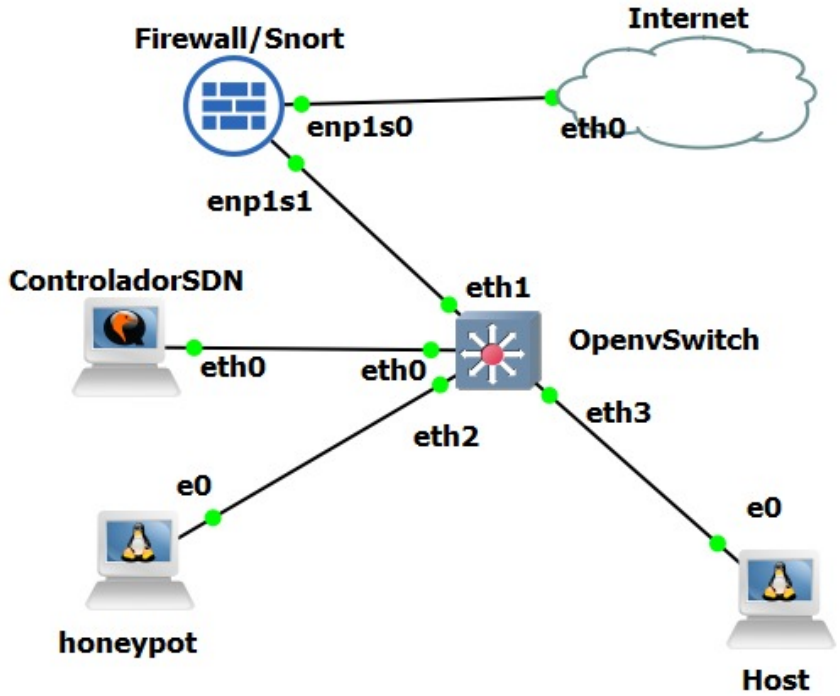

Figura 4.1: Arquitetura desenvolvida no GNS3
Autor

Foi utilizado o modo cliente / servidor do GNS3. O servidor foi configurado no GNS3 server VM, rodando em uma VM. Nesse servidor, foram instaladas as imagens do Debian, Open Virtual Switch, Kaly Linux e Debian 10 minimal. Foram utilizados os seguintes ativos de rede para compor o ambiente de testes:

- *Switch Virtual OVS;*
- *Firewall iptables com suporte a filtro de pacotes e IPS, rodando em uma VM Debian;*
- *Ryu como controladora SDN e Mininet rodando em uma VM docker;*
- *Computador para simular ataques de negação de serviço utilizando uma VM Debian 10 minimal;*
- *Imagem do Kaly Linux, distinta da utilizada no host, operando como honeynet.*

As VMs foram otimizadas para reduzir o consumo de memória RAM e processador, possibilitando a execução dos testes.

4.1.1 Switch Virtual OVS

Toda topologia foi elaborada utilizando a ferramenta GNS3 para facilitar a administração de conexões entre os diferentes componentes utilizados na solução. O firewall, controlador SDN, honeypot e host atacante foram interligados utilizando o Open Virtual Switch (OVS) na versão 2.4.0. Ao ser inicializado o OVS funciona como um switch de camada dois, permitindo a interligação de diferentes dispositivos na mesma VLAN. Para essa prova de conceito, o switch OVS foi associado à controladora, conforme demonstrado em 4.2, separando assim o plano de dados do plano de controle.

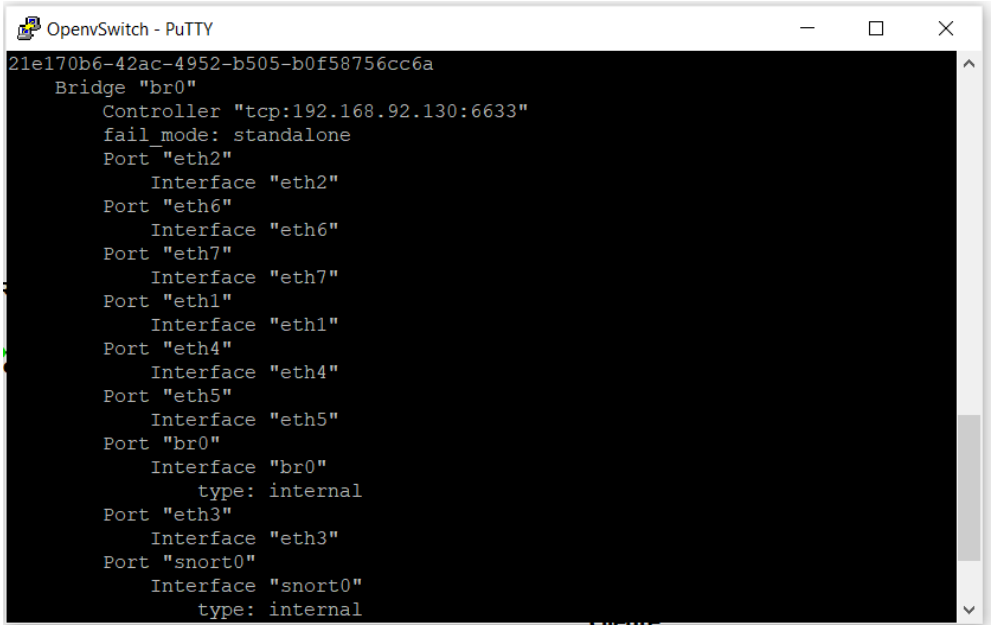

Figura 4.2: Associação do OVS ao controlador
Autor

As tabelas de fluxos de encaminhamento de pacotes foram definidas na controladora SDN da solução.

4.1.2 Firewall / IPS

A solução proposta conta com uma VM com Linux Debian e os *softwares* iptables e Snort devidamente instalados e configurados para operar como *firewall* e IPS da rede, inspecionando e filtrando pacotes que saem da rede interna destinados à Internet. Para fins de teste, foram definidas regras no Snort para detectar ataques vindos da rede interna com características de negação de serviço, como excesso de conexões HTTP direcionadas para um servidor externo. Quando o Snort percebe uma situação anormal de tráfego, é gerado um log no sistema operacional e é utilizado um script que gera uma chamada REST API para a controladora Ryu quando é detectada alguma situação que não esteja de acordo com o padrão predeterminado e que precise ser bloqueada.

4.1.3 Identificação de ataque de negação de serviço

A identificação de ataques de negação de serviço utilizando Snort é alvo de diversos estudos no meio científico, conforme pode ser visto em [97], [98], [99], e consiste em identificar a quantidade de pacotes destinados a determinado endereço IP de destino dentro de um intervalo de tempo preestabelecido. Quando o Snort percebe que a condição definida na regra foi atendida, é gerado um log, que fica armazenado na pasta /var/log/snort. Esse log pode ser exportado, utilizando a ferramenta syslog, para um servidor concentrador de eventos SIEM, que fica responsável por armazenar, associar e apresentar os eventos para o administrador da rede.

Para os testes realizados neste ambiente, foi utilizado um *script* que promove a integração entre o Snort e o Ryu, exportando os logs de falha para a controladora utilizando chamadas REST API.

A regra utilizada para identificar o ataque de negação de serviço conta a quantidade de conexões TCP para porta 80 no intervalo de 10 segundos direcionados para o mesmo endereço IP. Quando o Snort receber uma quantidade de pacotes superior a 70 nesse intervalo, será gerado um alerta, que será lido pelos *scripts* de automação do controlador SDN e *firewall*. A regra utilizada segue abaixo:

alert tcp $!HOME_NET any -> $HOME_NET 80 (flags: S; msg:"Possible TCP DoS"; flow: stateless; threshold: type both, track by_src, count 70, seconds 10; sid:10001;rev:1;)

Para validar o cenário proposto, será utilizado o hping3 no PC atacante para gerar volume de tráfego superior à regra do Snort. A quantidade de pacotes gerados foi pequena para facilitar a simulação do ataque.

4.1.4 PC Atacante

Uma máquina virtual simula o *host* atacante e, para isso, foi utilizada a ferramenta hping3, disparando *floods* conforme demonstra a figura 4.3 direcionados para o servidor web da topologia montada com o GNS3.

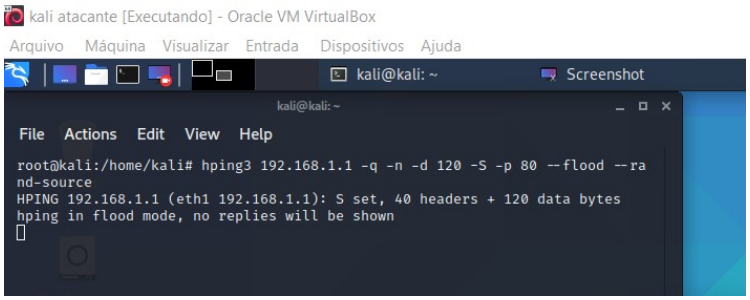

Figura 4.3: Ataque iniciado a partir da VM *host* atacante

4.1.5 Servidor Web

Os pacotes simulando o ataque foram direcionados para uma máquina virtual contendo um servidor web com a finalidade de reproduzir um ambiente de produção.

A figura 4.4 ilustra o início de um ataque em que o Snort ainda não identificou o tráfego anômalo gerado pelo cliente da rede LAN. A ferramenta hping3 instalada no PC inicia ataque de negação de serviço destinado a um IP da DMZ. Até o Snort detectar o ataque, solicitar a inclusão da regra na controladora, o servidor de destino continua recebendo todo o volume de tráfego gerado pelo atacante.

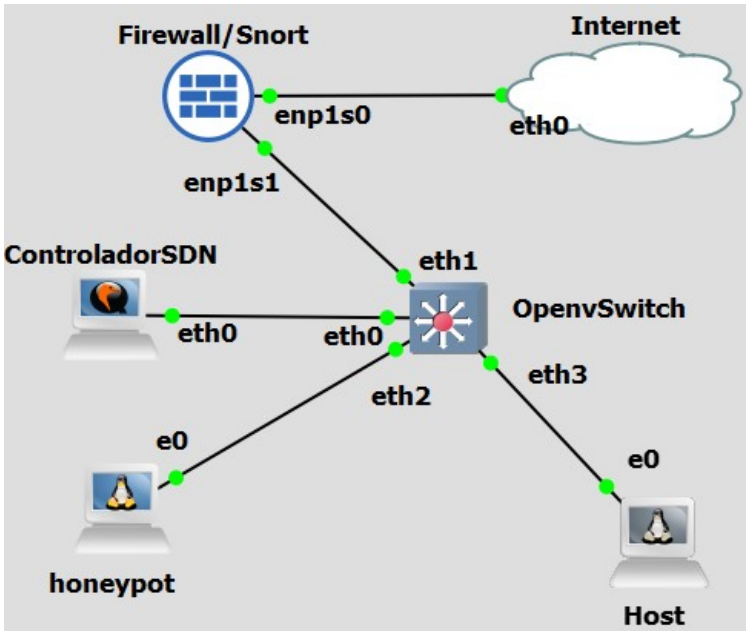

Figura 4.4: Arquitetura desenvolvida no GNS3 para prova de conceito
Autor

Soluções semelhantes têm sido desenvolvidas para permitir a rápida convergência da rede local para proteção de ataques externos e internos, conforme pode ser visto em [100] e [101]. Este trabalho de pesquisa tem como diferencial encaminhar uma cópia dos pacotes identificados como indevidos para um servidor instalado em uma rede honeynet, permitindo que o administrador possa analisar se o evento recebido é referente a uma tentativa de ataque ou

apenas um fluxo normal da rede. A figura que retrata a rede após a convergência está descrita em 4.5. Quando é detectado um evento anômalo, o administrador pode visualizar analisando os logs gerados pela plataforma, tendo a opção de solicitar à controladora o bloqueio imediato do equipamento de origem.

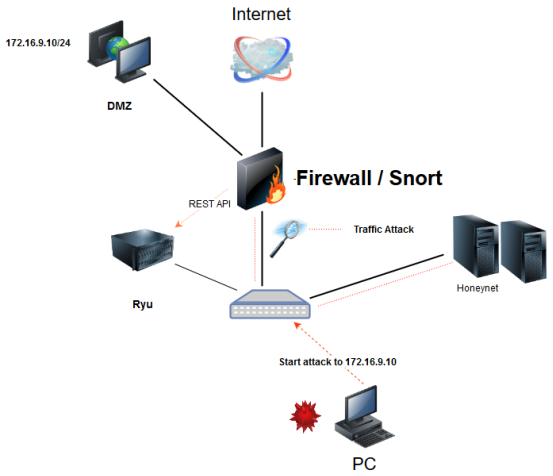

Figura 4.5: Convergência de segurança da rede SDN

4.1.6 Resultados

Para validar o conceito, foram executados 100 testes, automatizados a partir de scripts em shell Linux, para medir o tempo em que o Snort identifica o tráfego anômalo, faz a solicitação para incluir um novo flow na tabela e força o switch a encaminhar o tráfego para a honeynet. Neste caso, estamos medindo o tempo de convergência descrito na figura 4.5.

O tempo de convergência, em que ocorre a proteção da rede, obedece a uma função linear:

$$C(x, y, z) = (x + z) + y \tag{4.1}$$

na qual x é o tempo que o Snort leva para identificar o tráfego anômalo e gravar a regra na controladora via API, e y é o tempo que a rede SDN leva para convergir, conforme regra definida em 4.4. A variável z representa o tempo utilizado para analisar o volume de conexões no intervalo x definido na regra do Snort. Regras para identificação de ataques de flood utilizam intervalos de tempo em segundos, pois as características desse tipo de ataque é tentar encher o buffer da interface de rede do servidor, deixando-o sem capacidade para responder a novas requisições. O tempo de convergência total é influenciado por z, que poderá ser modificado para atender a realidade da rede. Assim sendo, o tempo de convergência C é a uma função linear resultante da soma das variáveis $x + y + z$. O tempo de detecção de um ataque pelo Snort (x) será diretamente influenciado por z. Um z reduz o tempo de reação da rede, mas traz alguns problemas como falsos positivos e aumento no consumo da CPU. Neste teste, optamos por $z = 10$ para conseguir executar os testes com melhor desempenho.

Utilizando hping3 com os parâmetros de flood habilitados, direcionando o tráfego para a porta 80 do servidor instalado no ambiente de DMZ e executando os testes por 100 vezes identificamos os seguintes valores de x apresentados na tabela 4.1.

O tempo que a controladora leva para cadastrar novas regras (y) também foi medido nos testes, e os resultados são apresentados na tabela 4.1.

Durante a execução dos testes, verificamos um aumento expressivo do tempo de identificação do ataque pelo Snort. Podemos constatar um aumento no consumo de CPU e memória da máquina virtual. Infelizmente, pelo cenário de pandemia que estamos vivendo, não foi possível aplicar a solução proposta utilizando os switches SDN que foram adquiridos para este propósito, assim como executar toda solução em servidores com maior capacidade computacional. O aumento no consumo da CPU do hospedeiro justifica o aumento das variáveis x e y.

A figura 4.6 mostra o gráfico de resumo do resultado da convergência das variáveis x, y e z:

Figura 4.6: Resultado de 100 testes realizados, medindo os fatores que compõem e tempo de convergência

A tabela 4.1 mostra alguns dos resultados obtidos na bateria de testes.

Tabela 4.1: Resultado dos testes

Período da regra no Snort	***Tempo de identificação do ataque pelo Snort***	***Tempo que a rede SDN levou para convergir***
10	*26*	*107*
10	*28*	*64*
10	*21*	*91*
10	*22*	*85*
10	*28*	*89*
10	*27*	*72*
10	*15*	*97*

Continua na próxima página

Tabela 4.1 – Continuação da página anterior

Período da regra no Snort	***Tempo de identificação do ataque pelo Snort***	***Tempo que a rede SDN levou para convergir***
10	*14*	*88*
10	*28*	*114*
10	*13*	*49*
10	*22*	*68*
10	*18*	*64*
10	*24*	*92*
10	*19*	*71*
10	*22*	*90*
10	*16*	*64*
10	*24*	*111*
10	*10*	*78*
10	*11*	*98*
10	*15*	*75*
10	*10*	*50*
10	*23*	*110*
10	*20*	*105*
10	*22*	*68*
10	*22*	*56*
10	*27*	*116*
10	*25*	*72*

O resumo dos dados apresentados pode ser visto na tabela 4.2. O desvio padrão das variáveis x e y são menores que os valores médios, indicando assim baixa variabilidade na série de dados.

Tabela 4.2: Resumo dos resultados.

	Identificação do ataque pelo Snort (X)	Tempo que a rede SDN levou para convergir (Y)
Média	19,13131313	80,49494949
Desvio Padrão	5,783373006	19,16297658

Para verificar como está a variabilidade dos dados, foi gerado um gráfico de boxplot das variáveis x e y, no qual podemos observar uma menor dispersão dos valores em x representado pela figura 4.7.

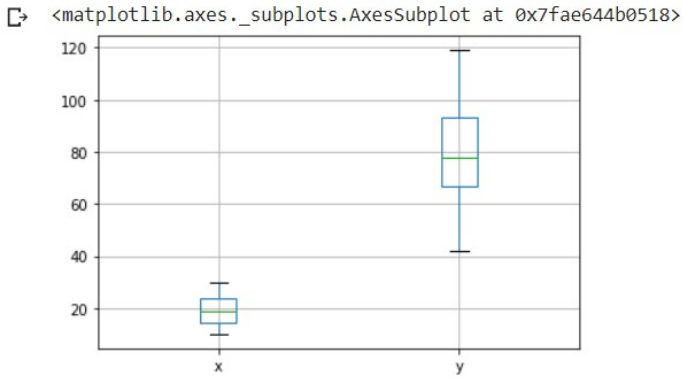

Figura 4.7: Gráfico BoxPlot das variáveis X e Y
[Autor]

Os valores apresentados demonstram que há pouca variação na média do tempo relativo à identificação de um ataque pelo Snort variável x, o bloqueio é realizado de forma imediata fazendo com que o tráfego malicioso não possa contaminar outros componentes da rede.

Conforme apresentado na seção resultados, a solução proposta se mostrou eficaz para identificar ataques e realizar ações para mitigar ataques de negação de serviço de dispositivos oriundos da rede interna. O tempo de convergência é influenciado diretamente pelas regras definidas no Snort, assim como pelo tempo de atualização das regras entre a controladora e o IDS. A redução desses intervalos aumenta o volume de tráfego entre a controladora e o switch, retardando decisões de encaminhamento.

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 CONCLUSÃO E TRABALHOS FUTUROS

A solução proposta apresentou ser viável de execução em um ambiente controlado, identificando ataques oriundos da rede interna e realizando a convergência da rede para inibir a continuidade desses ataques. Após a publicação do artigo que deu origem a esta publicação na conferência WCNPS, surgiram diversos trabalhos semelhantes, reforçando a importância do tema proposto.

No decorrer da execução dos resultados obtidos neste trabalho, foram encontradas algumas oportunidades de melhorar a segurança proposta. Ataques originados da Internet para a rede DMZ em geral podem causar indisponibilidades por um período prolongado devido à grande quantidade de tráfego, e essas indisponibilidades são muito difíceis de serem corrigidas sem uma ação da operadora de telecomunicações. Foi estudada então a possibilidade de integrar a detecção realizada no Snort em associação à controladora local com a operadora fornecedora do serviço de dados. Será realizado o trabalho futuro de informar à operadora, via BGP, a lista de IPs que precisam ser bloqueados. Outra sugestão de melhoria prevista neste trabalho é o desenvolvimento de aplicativos que enviem dinamicamente para as operadoras, via BGP, os IPs de origem que precisam ser bloqueados, impedindo que o tráfego anômalo passe pelo link de internet e consuma os recursos de banda contratados.

Para este projeto, foram adquiridos switches SDN e servidores para elaboração de um ambiente controlado, que futuramente seria colocado em produção. No entanto, com o avanço da pandemia iniciada em 2020, encerraram-se as atividades presenciais na UnB, impedindo que o ambiente de testes fosse elaborado. A título de trabalho futuro, vamos realizar os testes feitos neste documento num cenário com switches e servidores reais. A forma como foi realizada a identificação dos ataques também será modificada, não apenas por regras estáticas, mas analisando o comportamento do tráfego ao longo do tempo, utilizando regras heurísticas por meio de processo predeterminado para esse objetivo.

A solução proposta pretende interagir com dispositivos de redes convencionais, o que aumenta ainda mais o perímetro de atuação deste trabalho de pesquisa, conforme pode ser visto em [102]. A popularização de dispositivos IoT vem crescendo de forma exponencial ao longo dos anos [103]. Dispositivos IoT vêm se transformando em elementos de potencial falha de segurança [104], tornando assim fundamental desenvolver soluções de segurança que sejam capazes de interagir com gateways e dispositivos IoT associados a redes SDN para identificar e mitigar possíveis ataques e falhas de segurança. [105].

REFERÊNCIAS BIBLIOGRÁFICAS

1 JUNIOR, G.; SILVA, L. da. Virtualização de funções de rede em nuvem para instituições públicas. Pós-Graduação em Ciência da Computação, 2017.

2 JÚNIOR, E. C. d. A. Wi-Flow: uma arquitetura baseada em SDN para o gerenciamento e mobilidade em redes Wi-Fi com suporte à autenticação 802.1 x. *Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2016.*

3 NOX, year = 2020, month = outubro, company = gta.ufrj.br, urldate = 2021, janeiro,21, url = https://www.gta.ufrj.br/ensino/eel879/trabalhos$_v f_2 010_2 / rodrigo / nox.html, .$

4 JÚNIOR, S.; GONÇALVES, J. F. Storm ids: um sistema de detecção de intrusão escalável e distribuído. 2016.

5 BARBOSA, R. R. Migração de redes tradicionais para SDN. *Tese (Doutorado) — Universidade de São Paulo, 2018.*

6 SILVA, A. C. M.; MAIA, L. C. G.; VILLELA, H. F. Redes definidas por software–sdn–um estudo sobre as vantagens e suas características. Computação & Sociedade, *v. 1, n. 1, 2019.*

7 MARCHESAN, G. et al. Uma análise comparativa entre paradigmas de virtualização de redes. Universidade Federal de Santa Maria, 2018.

8 KREUTZ, D.; RAMOS, F. M.; VERISSIMO, P. E.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, *Ieee, v. 103, n. 1, p. 14–76, 2014.*

9 FERNANDEZ, M. P. Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive. In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). *[S.l.: s.n.], 2013. p. 1009–1016. ISSN 1550-445X.*

10 MODA, C. S. et al. Uma proposta de redirecionamento de fluxos de rede usando openflow para migração de aplicações entre nuvens. Universidade Federal de São Carlos, 2014.

11 AMIN, R.; REISSLEIN, M.; SHAH, N. Hybrid SDN Networks: A Survey of Existing Approaches. IEEE Communications Surveys Tutorials, *v. 20, n. 4, p. 3259–3306, Fourthquarter 2018. ISSN 1553-877X.*

12 GARCIA, T. O. Definição de novas regras para o ids snort em redes definidas por software. 2016.

13 MILLER, S.; BUSBY-EARLE, C. The role of machine learning in botnet detection. In: IEEE. 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST). *[S.l.], 2016. p. 359–364.*

14 CHANG, W.; MOHAISEN, A.; WANG, A.; CHEN, S. Measuring botnets in the wild: Some new trends. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. *[S.l.: s.n.], 2015. p. 645–650.*

15 MARZANO, A.; ALEXANDER, D.; FAZZION, E.; FONSECA, O.; CUNHA, I.; HOEPERS, C.; STEDING-JESSEN, K.; CHAVES, M. H.; GUEDES, D.; JR, W. M. Monitoramento e caracterização de botnets bashlite em dispositivos iot. In: SBC. Anais Principais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. *[S.l.], 2018.*

16 OLIVEIRA, J. G. A. d. Honeyselk: um ambiente para pesquisa e visualização de ataques cibernéticos em tempo real. *2016.*

17 FARES, A. A. Y. R.; FILHO, F. L. de C.; GIOZZA, W. F.; CANEDO, E. D.; MENDONÇA, F. L. L. de; NZE, G. D. A. Dos attack prevention on ips sdn networks. In: IEEE. 2019 Workshop on Communication Networks and Power Systems (WCNPS). *[S.l.], 2019. p. 1–7.*

18 MATTOS, D. M. F.; DUARTE, O. C. M. B.; JANEIRO-RJ-BRASIL, R. de. Authflow: Um mecanismo de autenticaç ao e controle de acesso para redes definidas por software. *2014.*

19 FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn: an intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, *ACM New York, NY, USA, v. 44, n. 2, p. 87–98, 2014.*

20 LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. *[S.l.: s.n.], 2010. p. 1–6.*

21 KLAUBERG, A. F. Isolamento das redes virtuais (vlan) em uma infraestrutura em nuvem usando uma abordagem sdn/openflow. Monografia (Tecnólogo em Sistemas de Telecomunicações)-Instituto Federal de Santa Catarina, R. José Lino Kretzer, *p. 88103–310, 2016.*

22 PACHECO, D. H.; ISHIMORI, A. N.; FARIAS, F. N. N.; ABELÉM, A. J. G. Uma estratégia para o serviço de cálculo de caminhos em redes definidas por software. Revista Brasileira de Computação Aplicada, *v. 8, n. 2, p. 71–81, 2016.*

23 ANDRIOLI, L.; RIGHI, R. da R.; AUBIN, M. R. Analisando métodos e oportunidades em redes definidas por software (sdn) para otimizações de tráfego de dados. Revista Brasileira de Computação Aplicada, *v. 9, n. 4, p. 2–14, 2017.*

24 ROTHENBERG, C. E.; NASCIMENTO, M. R.; SALVADOR, M. R.; MAGALHÃES, M. F. Openflow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. Cad. CPqD Tecnologia, Campinas, *v. 7, n. 1, p. 65–76, 2010.*

25 LARA, A.; KOLASANI, A.; RAMAMURTHY, B. Network innovation using openflow: A survey. IEEE communications surveys & tutorials, *IEEE, v. 16, n. 1, p. 493–512, 2013.*

26 GUEDES, D.; VIEIRA, L. F. M.; VIEIRA, M.; RODRIGUES, H.; NUNES, R. V. Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC, *v. 30, n. 4, p. 160–210, 2012.*

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

28 TATSCH, C. G. Sistema de detecção e prevenção de ataques port scan em redes openflow sdn. 2017.

29 MARCELLO, G. H. dos S. Nox: Sistema operacional para redes.

30 MONTEZELO, G. H. Práticas laboratoriais em sdn usando mininet e pox. Universidade Federal de Uberlândia, 2017.

31 CHRISTÓFARO, A. C. d. O. Análise de técnicas de otimização multiobjetivo para o posicionamento de controladores em redes sdn. 2018.

32 LOPEZ, M. A.; FIGUEIREDO, U. da R.; LOBATO, A. G. P.; DUARTE, O. C. M. B. Broflow: Um sistema eficiente de detecç ao e prevenç ao de intrusao em redes definidas por software. CSBC, Centro de Convenç oes Brasil, v. 21, 2014.

33 HOT Standby Router Protocol Features and Functionality. Disponível em: <http://www.cisco.com/c/en/us/support/docs/ip/hot-standby-router-protocol-hsrp/9234-hsrpguidetoc.html>.

34 BHOLEBAWA, I. Z.; DALAL, U. D. Performance analysis of sdn/openflow controllers: Pox versus floodlight. Wireless Personal Communications, Springer, v. 98, n. 2, p. 1679–1699, 2018.

35 RYU. 2020. Disponível em: <https://github.com/faucetsdn/ryu>.

36 RYUBOOK. Disponível em: <https://osrg.github.io/ryu-book/en/Ryubook.pdf>.

37 KUBO, R.; FUJITA, T.; AGAWA, Y.; SUZUKI, H. Ryu sdn framework: Open-source sdn platform software. NTT Tech. Rev, v. 12, n. 8, p. 1–5, 2014.

38 GOVINDRAJ, S.; JAYARAMAN, A.; KHANNA, N.; PRAKASH, K. R. Openflow: Load balancing in enterprise networks using floodlight controller. University of Colorado, 2012.

39 FLOODLIGHT. Disponível em: <https://github.com/floodlight/floodlight/commit/70379784592392bf75ff672eddbb3c405464452d>.

40 ALMEIDA, R. S. Uma comparação entre controladores de redes definidas por software em cenários de internet das coisas. 2019.

41 GARCIA, B. R. OpenDaylight SDN controller platform. Dissertação (B.S. thesis) — Universitat Politècnica de Catalunya, 2015.

42 OPENDAYLIGHT. Disponível em: <https://www.opendaylight.org/>.

43 JUNIOR, D. M. M. Segurança da informação: uma abordagem sobre proteção da privacidade em internet das coisas. Tese (Doutorado) — Pontifícia Universidade Católica de São Paulo, Brazil, 2018.

44 FONTES, E. L. G. Segurança da informação. [S.l.]: Saraiva Educação SA, 2017.

45 ABNT, N. IEC 27.002: 2005 (antiga NBR ISO/IEC 17799: 2005)-Código de Prática para a Gestão da Segurança da Informação. 2013.

46 KREUTZ, D.; RAMOS, F. M.; VERISSIMO, P. Towards secure and dependable software-defined networks. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. [S.l.: s.n.], 2013. p. 55–60.

47 GONÇALVES, D. G.; FILHO, F. L. de C.; MARTINS, L. M. E.; KFOURI, G. d. O.; DUTRA, B. V.; ALBUQUERQUE, R. d. O.; SOUSA, R. T. de. Ips architecture for iot networks overlapped in sdn. In: IEEE. 2019 Workshop on Communication Networks and Power Systems (WCNPS). [S.l.], 2019. p. 1–6.

48 FIORENZA, M.; KREUTZ, D. Firewalls em redes definidas por software: Estado da arte. In: SBC. Anais da XVII Escola Regional de Redes de Computadores. [S.l.], 2019. p. 81–88.

49 LEITE, H. O. O impacto da segurança da informação nas empresas de prestação de serviços bancários: um estudo em uma empresa personalizadora de cartões de pagamento bandeirados. Tese (Doutorado) — Mestrado em Sistemas de Informação e Gestão do Conhecimento, 2017.

50 RIGHI, M.; NUNES, R. Detecção de ddos através da análise da recorrência baseada na extração de características dinâmicas. Anais do XV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais-SBSeg, v. 2015, p. 314–317, 2015.

51 BARTHOLEMY, A.; CHEN, W. An examination of distributed denial of service attacks. In: IEEE. 2015 IEEE International Conference on Electro/Information Technology (EIT). [S.l.], 2015. p. 274–279.

52 SANTOS, M. A. B. d. Alocação de redes virtuais e controladores em redes definidas por software: uma análise de custo, rede e dependabilidade. Universidade Federal de Pernambuco, 2017.

53 MIAN, A. N.; MAMOON, A.; KHAN, R.; ANJUM, A. Effects of virtualization on network and processor performance using open vswitch and xen server. In: IEEE. 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. [S.l.], 2014. p. 762–767.

54 PANTOJA, M. R. d. S. Comparação de infraestruturas baseadas em SDN e virtualização de redes para suporte a serviços VoIP. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2017.

55 PINHEIRO, B. Uma Abordagem SDN para Virtualizaçao de Redes. Tese (Doutorado) — PhD thesis, Universidade Federal do Pará, 2016.

56 ROTANDARO, R. R. d. A.; GUEDES, M. Redes locais virtuais. In: FUNDAÇÃO DE ENSINO E PESQUISA DO SUL DE MINAS. II Congresso Internacional do Grupo Unis. [S.l.], 2016.

57 SHARMA, T.; YADAV, R. Security in virtual private network. International Journal of Innovations & Advancement in Computer Science, v. 4, p. 669–675, 2015.

58 BORGES, F.; FAGUNDES, B. A.; CUNHA, G. N. da. Vpn: Protocolos e segurança. S/D, v. 10, 2019.

59 VPN SSL. 2020. Disponível em: <https://www.cisco.com/c/pt_br/support/docs/ios-nx-os-software/authentication-authorization-accounting-aaa/116313-configure-anyconnect-00.html>.

60 OVERLAYNETWORK. 2020. Disponível em: <https://searchnetworking.techtarget.com/definition/overlay-network>.

61 MORENO, E. Z. M. S. Middleware de comunicação para redes oportunistas. 2015.

62 REDES SOBREPOSTAS-OVERLAY. 2020. Disponível em: <https://www.comutadores.com.br/tag/bgp/>.

63 GNS3. 2020. Disponível em: <https://docs.gns3.com/docs/>.

64 VMARE. 2020. Disponível em: <https://www.vmware.com/br/topics/glossary/content/virtual-machine.html>.

65 FILHO, V. B.; JUNIOR, O. S.; CAMARGO, L. S. d. A. de. Software simuladores de rede. 2015.

66 MININET. 2020. Disponível em: <http://mininet.org/overview/>.

67 APICISCO. 2020. Disponível em: <https://ciscoredes.com.br/2017/02/13/o-que-e-api-e-rest-api/>.

68 API. 2020. Disponível em: <https://rockcontent.com/br/blog/rest-api/>.

69 NORTHBOUND. 2020. Disponível em: <pt-BR&sl=en&u=https://www.cisco.com/c/en/us/td/docs/net_mgmt/packet_telephony_center_virtual_switch/2-1/developer/guide/VSchap1.html&prev=search&pto=aue>.

70 NORTHBOUND. 2020. Disponível em: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_v1_2015_2/SDN/architecture.html>.

71 CUNHA, M. N.; MATUSHIMA, R.; KOPP, S.; SILVEIRA, R. M.; FAUSTINO, J. C.; NUNES, A. C. F. et al. Desvendando o conceito de CDN e sua aplicabilidade nas Redes Acadêmicas. 2015.

72 CDN. Disponível em: <https://www.akamai.com/br/pt/cdn/what-is-a-cdn.jsp>.

73 SADEGHIAN, A.; ZAMANI, M. Detecting and preventing DDoS attacks in botnets by the help of self triggered black holes. In: 2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE). *[S.l.: s.n.], 2014. p. 38–42.*

74 JABOUR, E.; DUARTE, O. C. Honeynets: Invasores, ferramentas, técnicas e táticas. http://www. gta. ufrj. br/seminarios/CPE825/tutoriais/eugenia/honeynets. PDF>. Acesso em, *v. 5, p. 31, 2015.*

75 ANDRADE, E. S. et al. Zona desmilitarizada (dmz): Um estudo. Universidade Federal Fluminense.

76 DHAWAN, M.; PODDAR, R.; MAHAJAN, K.; MANN, V. Sphinx: Detecting security attacks in software-defined networks. In: Ndss. *[S.l.: s.n.], 2015. v. 15, p. 8–11.*

77 SHIN, S.; YEGNESWARAN, V.; PORRAS, P.; GU, G. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. *[S.l.: s.n.], 2013. p. 413–424.*

78 BONDKOVSKII, A.; KEENEY, J.; MEER, S. van der; WEBER, S. Qualitative comparison of open-source sdn controllers. In: IEEE. NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. *[S.l.], 2016. p. 889–894.*

79 FOX, E. F. B.-L. et al. Detecção de ataques syn-flooding em redes definidas por software. Universidade Federal da Paraíba, 2019.

80 MOHAMMADI, R.; JAVIDAN, R.; CONTI, M. Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. IEEE Transactions on Network and Service Management, IEEE, v. 14, n. 2, p. 487–497, 2017.

81 FICHERA, S.; GALLUCCIO, L.; GRANCAGNOLO, S. C.; MORABITO, G.; PALAZZO, S. Operetta: An openflow-based remedy to mitigate tcp synflood attacks against web servers. Computer Networks, Elsevier, v. 92, p. 89–100, 2015.

82 FERNANDES, H. S. Provendo segurança em redes definidas por software através da integração com sistemas de detecção e prevenção de intrusão. Niterói, 2017.

83 JESUS, W. P. de; SILVA, D. A. da; SOUSA, R. T. de; SOUSA, F. V. L. da. Analysis of sdn contributions for cloud computing security. In: IEEE. 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. [S.l.], 2014. p. 922–927.

84 BRITO, I. V. S.; RIBEIRO, A. V.; SAMPAIO, L. N. Experiências com uso da ferramenta sdn-ips no testbed fibre para práticas de ensino de redes e cibersegurança. In: SBC. Anais do III Workshop do Testbed FIBRE. [S.l.], 2018.

85 LUSTOSA, H. R. Testes de segurança em sdn's, utilizando honeypot. 2016.

86 YADAV, S. K.; SHARMA, K.; ARORA, A. Security integration in ddos attack mitigation using access control lists. International Journal of Information System Modeling and Design (IJISMD), IGI Global, v. 9, n. 1, p. 56–76, 2018.

87 KLETTENBERG, J. et al. Segurança da informação: Um estudo sobre o uso da engenharia social para obter informações sigilosas de usuários de instituições bancárias. 2016.

88 COMO-A-CLOUDFLARE-FUNCIONA, company = cloudflare, url = https://support.cloudflare.com/hc/pt-br/articles/205177068-Como-a-Cloudflare-funciona-,.

89 SWITCHOPENFLOW. 2020. Disponível em: <https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/1611/b_1611_programmability_cg/OpenFlow.html&prev=search&pto=aue>.

90 FIND, install and publish Python packages with the Python Package Index. Disponível em: <https://pypi.org/>.

91 SNORTINTEGRATE. 2020. Disponível em: <https://ryu.readthedocs.io/en/latest/snort_integrate.html>.

92 FWSNORT. 2020. Disponível em: <https://www.cipherdyne.org/fwsnort/>.

93 ANTONAKAKIS, M.; APRIL, T.; BAILEY, M.; BERNHARD, M.; BURSZTEIN, E.; COCHRAN, J.; DURUMERIC, Z.; HALDERMAN, J. A.; INVERNIZZI, L.; KALLITSIS, M. et al. Understanding the mirai botnet. In: 26th {USENIX} Security Symposium ({USENIX} Security 17). [S.l.: s.n.], 2017. p. 1093–1110.

94 HPING7 nmap.

95 OPEN Virtual Switch Reference. Disponível em: <https://www.openvswitch.org>.

96 GULENKO, A.; WALLSCHLäGER, M.; KAO, O. A practical implementation of in-band network telemetry in open vswitch. In: . [S.l.: s.n.], 2018. p. 1–4.

97 MEROUANE, M. An approach for detecting and preventing ddos attacks in campus. Automatic Control and Computer Sciences, *Springer, v. 51, n. 1, p. 13–23, 2017.*

98 Feinstein, L.; Schnackenberg, D.; Balupari, R.; Kindred, D. Statistical approaches to ddos attack detection and response. In: Proceedings DARPA Information Survivability Conference and Exposition. *[S.l.: s.n.], 2003. v. 1, p. 303–314 vol.1.*

99 CHOI, J.; CHOI, C.; KO, B.; KIM, P. A method of ddos attack detection using http packet pattern and rule engine in cloud computing environment. Soft Computing, *Springer, v. 18, n. 9, p. 1697–1703, 2014.*

100 GONÇALVES, D. G.; KFOURI, G. d. O.; DUTRA, B. V.; ALENCASTRO, J. F. de; FILHO, F. L. de C.; LUCAS, M.; MARTINS, R. d. O.; JR, R. T. de S. Arquitetura de ips para redes iot sobrepostas em sdn.

101 HUANG, N.-F.; WANG, C.; LIAO, I.-J.; LIN, C.-W.; KAO, C.-N. An openflow-based collaborative intrusion prevention system for cloud networking. In: IEEE. 2015 IEEE International Conference on Communication Software and Networks (ICCSN). *[S.l.], 2015. p. 85–92.*

102 KFOURI, G. d. O.; GONÇALVES, D. G.; DUTRA, B. V.; ALENCASTRO, J. F. de; FILHO, F. L. de C.; MARTINS, L. M. e; PRACIANO, B. J.; ALBUQUERQUE, R. de O.; JR, R. T. de S. Design of a distributed hids for iot backbone components. In: FedCSIS (Communication Papers). *[S.l.: s.n.], 2019. p. 81–88.*

103 CALDAS FILHO, F. L. d.; MARTINS, L. M. C. e.; ARAÚJO, I. P.; MENDONcA, F. L. L. d.; COSTA, J. a. P. C. L. d.; DE SOUSA JúNIOR, R. T. Gerenciamento de Serviços IoT com Gateway Semântico. In: Atas das Conferências IADIS Ibero-Americanas WWW/Internet 2017 e Computação Aplicada 2017. *[S.l.]: IADIS Press, 2017. p. 199–206. ISBN 978-989-8533-70-8.*

104 DUTRA, B. V.; MARTINS, L. M. C. E. Hids by signature for embedded devices in iot networks. In: SERVICIO DE PUBLICACIONES. Actas de las V Jornadas Nacionales de Ciberseguridad Junio 5-7, 2019, Cáceres. *[S.l.], 2019. p. 53–61.*

105 SPERLING, T. L. von; FRANÇA, B. d. A.; FILHO, F. L. de C.; MARTINS, L. M. e; ALBUQUERQUE, R. d. O.; SOUSA, R. T. de. Evaluation of an iot device designed for transparent traffic analysis. In: IEEE. 2018 Workshop on Communication Networks and Power Systems (WCNPS). *[S.l.], 2018. p. 1–5.*