PUCRS

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

ALEX MULATTIERI SUAREZ OROZCO

BALANCEAMENTO ENTRE SEGURANÇA E DESEMPENHO NA COMUNICAÇÃO ENTRE OS PLANOS DE CONTROLE E DADOS EM REDES DEFINIDAS POR SOFTWARE

Porto Alegre
2018

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

BALANCEAMENTO ENTRE SEGURANÇA E DESEMPENHO NA COMUNICAÇÃO ENTRE OS PLANOS DE CONTROLE E DADOS EM REDES DEFINIDAS POR SOFTWARE

ALEX MULATTIERI SUAREZ OROZCO

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Avelino Francisco Zorzo

Porto Alegre
2018

Ficha Catalográfica

O74b Orozco, Alex Mulattieri Suarez

Balanceamento entre segurança e desempenho na comunicação entre os planos de controle e dados em Redes Definidas por Software / Alex Mulattieri Suarez Orozco . – 2018.

165 p.

Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Avelino Francisco Zorzo.

1. redes definidas por software. 2. SDN. 3. segurança. 4. desempenho. 5. balanceamento. I. Zorzo, Avelino Francisco. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Alex Mulattieri Suarez Orozco

BALANCEAMENTO ENTRE SEGURANÇA E DESEMPENHO NA COMUNICAÇÃO ENTRE OS PLANOS DE CONTROLE E DADOS EM REDES DEFINIDAS POR SOFTWARE

Tese apresentada como requisito parcial para obtenção do grau de Doutor em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 07 de Novembro de 2018.

BANCA EXAMINADORA:

DR. CARLOS ALBERTO MAZIERO - PPGINF - UFPR

DR. LISANDRO ZAMBENEDETTI GRANVILLE - PPGC - UFRGS

DR. TIAGO COELHO FERRETO - PPGCC - PUCRS

Dr. Avelino Francisco Zorzo (PPGCC/PUCRS - Orientador)

BALANCEAMENTO ENTRE SEGURANÇA E DESEMPENHO NA COMUNICAÇÃO ENTRE OS PLANOS DE CONTROLE E DADOS EM REDES DEFINIDAS POR SOFTWARE

RESUMO

Atualmente, a criptografia é utilizada como um padrão para proteger o tráfego de dados pela Internet. Por exemplo, em uma rede definida por software, é possível proteger o canal de comunicação do plano de controle utilizando criptografia. Entretanto, essa funcionalidade pode aumentar o uso de recursos e consequentemente comprometer o desempenho de *switches*, principalmente ao considerar *switches* de baixo custo.

Este trabalho visa a possibilidade de balancear segurança e desempenho na comunicação envolvendo controlador-switch em redes definidas por software. Para viabilizar essa possibilidade, a comunicação é gerenciada levando em consideração as mensagens conhecidas da API do *Southbound* para realizar um ataque. As mensagens que são consideradas sensíveis são gerenciadas com foco em segurança e as mensagens atualmente consideradas inofensivas são gerenciadas com foco em desempenho. Os resultados obtidos mostram que é possível gerenciar com sucesso o balanceamento de segurança e desempenho.

Palavras-Chave: SDN, Openflow, segurança, desempenho.

SECURITY AND PERFORMANCE BALANCING IN COMMUNICATION BETWEEN CONTROL AND DATA PLANES IN SOFTWARE-DEFINED NETWORKING

ABSTRACT

Nowadays, cryptography is being widely used as a standard for securing data exchange on the Internet. For instance, in software-defined networking, it is possible to apply cryptography in the control plane channel. However, it may increase resource usage, and consequently it could compromise the switch performance mainly in low-cost switches. This work brings the possibility of balancing security and performance in the controller-switch communication. To achieve that, we manage the communication by considering the security flaws known from southbound API messages in order to perform an attack. On the one hand, messages that are considered sensitive are more securely managed but, on the other hand, messages that are considered harmless are handled with a focus on performance. Our results show that it is possible to successfully balance security and performance.

Keywords: SDN, Openflow, security, performance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE PSEUDOCÓDIGOS

LISTA DE SIGLAS

AAA – Authentication, Authorization and Accounting
AES – Advanced Encryption Standard
API – Application Programming Interface
CBC – Cipher Block Chaining
CDN – Content Delivery Network
CFB – Cipher FeedBack
CLI – Command Line Interface
CE – Control Element
CPU – Central Processing Unit
DOS – Denial of Service
DDOS – Distributed Denial of Service
DSL – Domain Specific Language
FE – Forward Element
IDS – Intrusion Detection System
IETF – Internet Engineering Task Force
IPS – Intrusion Prevention System
LFB – Logical Function Block
LISP – Locator/ID Separation Protocol
MITM – Man in the Middle
NE – Network Element
RPC – Remote Procedure Calls
RTT – Round-Trip Time
SDN – Software-defined Networking
SMS – Systematic Mapping Study
SNMP – Simple Network Management Protocol
TLS – Transport Layer Security
VPN – Virtual Private Network
XML – eXtensible Markup Language
YANG – Yet Another Next Generation

SUMÁRIO

1. INTRODUÇÃO

Durante os últimos trinta anos, as tecnologias voltadas às redes de computadores têm evoluído com o objetivo de dar suporte a crescente dependência da humanidade pela computação. A grande quantidade de informações produzidas e armazenadas em computadores tem demandado tecnologias de comunicação mais rápidas e sofisticadas. Entretanto, os avanços na área de redes de computadores precisaram manter a compatibilidade com tecnologias legadas, ao ponto que essa dependência começou a limitar a inovação. Logo se percebeu que, para continuar o processo evolutivo, a abordagem atual de redes de computadores precisa ser completamente repensada e requer uma mudança de paradigma.

Isso porque as redes de computadores tradicionais são elaboradas com um amplo espectro de dispositivos, tais como roteadores, *switches*, e vários tipos de *middleboxes* (ou seja, manipuladores de tráfego por motivos diferentes do encaminhamento de pacotes, como um *firewall*, por exemplo). Estes dispositivos possuem diversos protocolos para o seu correto funcionamento, tornando o processo de administrar a rede uma atividade altamente complexa [418].

Essa situação impacta nos administradores de rede, que por sua vez, são responsáveis por configurar políticas encarregadas de definir o correto funcionamento da rede, com o objetivo de responder a vasta diversidade de eventos e aplicações. Esses profissionais necessitam transformar essas políticas, expressas normalmente em uma estrutura com um alto nível de abstração, em configurações, expressas em uma linguagem com um baixo nível de abstração, necessitando traduzir essas políticas em linguagens específicas de cada fabricante dos equipamentos, enquanto se adaptam às alterações das condições da rede, onde frequentemente realizam essas tarefas complexas utilizando ferramentas limitadas. Como resultado, estes profissionais são passíveis de cometerem erros constantemente.

Outra questão que os administradores necessitam enfrentar é o conservadorismo que a estrutura de rede atual impõe. Esse conservadorismo (também chamado de ossificação) ocorre em virtude da dependência que as tradicionais redes de comunicação possuem em compatibilizar diversas tecnologias e protocolos que foram surgindo no decorrer do tempo, o que torna difícil o processo de inovação, seja em termos de infraestrutura física ou protocolos, assim como em outras infraestruturas críticas da sociedade (como energia elétrica, água, etc.), torna-se difícil promover algum avanço significativo. Isso ocorre porque os atuais dispositivos de rede apresentam uma integração vertical, ou seja, agrupam as atividades de gerenciar o controle da rede e gerenciar os dados a serem trafegados nessa rede (muitas vezes com as funcionalidades implementadas em hardware), amarrando assim a infraestrutura e ficando dependente da administração individualizada de cada equipamento, mas com o olhar no escopo de toda a rede.

Nesse sentido, surgiu o conceito de Rede Definida por Software (SDN – *Software-defined Networking*), um novo paradigma de redes que possibilita romper a barreira da ossificação. Esse conceito permite a quebra dessa integração vertical, tirando a responsabilidade dos equipamentos de rede (tais como roteadores e *switches*) em controlar a rede, ficando somente com a tarefa de gerenciar o encaminhamento dos dados na rede. No conceito SDN, o controle da rede deixa de ficar distribuído entre cada dispositivo da rede e fica logicamente centralizado, o que faz com que o controle da rede possa ser gerenciado via software, e assim adquire a habilidade de programar a rede [302]. Dessa forma, pode-se organizar as responsabilidades em planos, onde a gestão da rede com alto nível de abstração fica a cargo do plano de gerenciamento (gerência via software), a concretização desse gerenciamento em termos de regras de baixo nível de abstração para instrução dos equipamentos de rede fica a cargo do plano de controle, e o encaminhamento dos dados que trafegam na rede fica a cargo do plano de dados. Em um modelo de rede clássico, esses três planos são associados dentro dos equipamentos de rede (também chamado de integração vertical), porém, em SDN, esses planos são desassociados (conhecido como quebra de integração vertical). Isso permite separar as definições de políticas de rede e as suas respectivas implementações nos dispositivos de rede.

A SDN tem atraído uma atenção significativa do meio acadêmico e da indústria. Por exemplo, profissionais da indústria criaram recentemente a Open Network Foundation [423], uma organização dirigida à indústria para promover a SDN e definir padronizações sobre o tema. Outra iniciativa, mas da perspectiva acadêmica, provém da criação da OpenFlow Network Research Center [424], com foco em pesquisas relacionadas a SDN.

Esta quebra de paradigma (abordagem de rede tradicional vs SDN) tem gerado um espectro de novos desafios (por exemplo, como compatibilizar esses paradigmas, como prover o paradigma SDN nos mais diversos cenários de redes que existem atualmente, etc.), assim como trazendo de volta vários problemas encontrados nas redes tradicionais, como desempenho, resiliência, escalabilidade, confiabilidade e segurança. Isso oferece a oportunidade de áreas correlatas investirem esforços na solução dos desafios que a SDN impõe. Nesse sentido, segurança é uma das áreas que SDN apresenta desafios complexos em aberto [258][198].

1.1 Definição do problema

Esta tese tem o objetivo de responder a questão sobre qual a forma mais adequada de promover uma comunicação segura, entre controladores e *switches*, de forma a não comprometer o desempenho.

De acordo com Feghali [167], a ausência de criptografia na troca de mensagens entre o controlador e *switches* expõe uma série de vulnerabilidades nas comunicações; a

ausência de um mecanismo de autenticação neste nível pode levar a uma série de ataques, o que torna mais fácil para o atacante espionar o tráfego da rede, com o intuito de identificar, além de quais dados estão trafegando na rede, quais tráfegos são possíveis de serem realizados na rede.

O trabalho de Samociuk [489] destaca a adoção do TLS pelos fabricantes envolvidos com Openflow. Como ilustrado na Tabela 1.1, mais de 50% dos fabricantes relacionados não oferecem suporte a TLS.

Esse alto índice de falta de suporte a TLS motiva o questionamento do motivo pelo qual o protocolo Openflow não reforça o uso de uma comunicação segura nessa camada de controle da rede. Três situações podem ser deduzidas como possíveis justificativas:

1. O protocolo TLS não ser seguro.

2. Não ser oferecida implementação de versões do TLS mais atuais.

3. O custo de prover uma comunicação segura ser muito alto.

Quanto à primeira justificativa, o trabalho de Al Fardan [19] provou que o protocolo TLS é inseguro, enfraquecendo a confiabilidade do protocolo. Este estudo mostra que é possível obter alguma informação do protocolo, mas, em termos práticos, não torna o protocolo totalmente inseguro, pois não permite obter um conjunto de informações significativo, sendo somente uma prova formal. Como prover algum nível de segurança é melhor do que não prover segurança alguma, essa afirmação não justifica a dificuldade de adoção de TLS.

Tabela 1.1 – Suporte a TLS, por fabricante Openflow

Fabricante	Suporte a TLS
HP Switch	Não
Brocade Switch	Somente porta do Controlador
Dell Switch	Não
NEC Switch	Parcial
Indigo Switch	Não
Pica8 Switch	Somente novas versões
Open vSwitch	Sim
NOX controller	Não
Brocade Vyatta controller	Sim
POX controller	Não
Beacon controller	Não
Floodlight controller	Não
MuL controller	Não
FlowVisor	Não
Big Network controller	Sim
Controladores *Open Source* (ex.: Ryu, OpenDaylight)	Sim

Ao analisar a segunda justificativa, como vários fabricantes implementaram o protocolo TLS em seus produtos, e o protocolo possui uma especificação bem definida [31], esta situação também não justifica o uso opcional de TLS.

Sendo assim, resta a terceira justificativa ser analisada. Não foram localizados em trabalhos relacionando o custo de proteger a comunicação entre controlador-*switch*. Para critérios de comparação, foi realizado um estudo para avaliar o custo de uma comunicação segura em um ambiente de nuvem distribuída, apresentado no Apêndice A. Nesse estudo, foram observados os impactos do uso de criptografia simétrica para proteger os dados trafegados nesse tipo de comunicação. Entre os resultados obtidos nesse estudo, o algoritmo que apresentou o menor overhead com um tamanho de chave de 128 bits (AES, no modo de operação CBC), apresentou um uso de CPU de aproximadamente 56%.

Se uma comunicação segura apresentou um *overhead* nessa magnitude em um cenário simples, como o analisado nesse estudo, se o *overhead* produzido pela criptografia for equivalente na comunicação envolvendo controlador-*switch*, em um ambiente SDN, esse impacto pode ser expressivo. Isso em virtude do controle da rede SDN ser logicamente centralizada, fazendo com que o uso de CPU durante as atividades de operação da rede seja somado ao uso de CPU para as atividades de criptografia. Inclusive, pensando em um cenário complexo, como SDN aplicado em *datacenters* ([543], [185]), formado por vários *switches* e controladores, ou um cenário envolvendo *switches* de baixo custo, com um hardware extremamente limitado para poder efetuar comunicações onde o tráfego de controle é intenso, como uma SDN desenvolvida para casas inteligentes [349], torna possível que o uso de CPU nos *switches* para proteger a comunicação possa representar a inviabilidade do uso de uma comunicação segura, a fim de prover um melhor desempenho da rede.

Hipótese

Esta pesquisa visa investigar a seguinte hipótese: *É possível balancear segurança e desempenho na comunicação do plano de controle de um ambiente SDN, entre controladores e switches, com um nível de segurança tal que não comprometa a segurança da rede.* Para embasar a hipótese, as seguintes perguntas de pesquisa serão respondidas:

1. Quais são as principais vulnerabilidades, ataques e soluções existentes para prover segurança em ambientes SDN? Esta questão visa compreender o atual cenário da segurança em SDN, de forma a prover um arcabouço de conhecimentos sobre o assunto para definir os requisitos a serem atendidos para poder embasar a hipótese. Embora vários trabalhos, que visam organizar as pesquisas sobre o tema, já foram desenvolvidos ([507], [176], [28], [13], [37], [506], [523]), nenhum dos estudos existentes na literatura sobre segurança em SDN utilizou uma metodologia bem definida, visando abranger todo o escopo de trabalhos envolvendo o tema. Além disso, como

SDN é um *hot topic*, novos trabalhos surgem a todo o momento, necessitando constantemente de uma atualização do estado da arte.

2. Quais são os impactos em termos de desempenho, ao prover segurança na comunicação entre controlador e *switch*, em um ambiente SDN? Ao responder esta pergunta, será possível avaliar o nível de preocupação necessária com desempenho ao prover segurança na comunicação envolvendo controlador e *switch* em ambientes SDN.

3. Como prover uma solução para balancear desempenho e segurança, na comunicação entre controlador e *switch*, em um ambiente SDN? Uma vez que os requisitos de segurança e desempenho são analisados e compreendidos, a resposta deste questionamento proporciona a criação de uma abordagem para proporcionar um ambiente de comunicação entre controlador e *switch*, em ambientes SDN, que leva em consideração tais requisitos para promover uma comunicação segura sem comprometer o desempenho da infraestrutura da rede.

1.2 Organização

Este documento é organizado da seguinte maneira:

1. O Capítulo 2 apresenta o embasamento teórico sobre SDN.

2. O Capítulo 3 aborda um estudo realizado sobre as questões de segurança envolvendo SDN, com o intuito de embasar a justificativa do tema.

3. O Capítulo 4 é dedicado à abordagem desenvolvida para balancear segurança e desempenho na comunicação envolvendo o controlador e *switch* de ambientes SDN.

4. O Capítulo 5 apresenta os experimentos realizados para validar a abordagem exposta no Capítulo 4, assim como as discussões envolvendo os resultados obtidos.

5. O Capítulo 6 são abordadas as conclusões finais, além de um resumo da pesquisa, que inclui a validação da hipótese baseada nas contribuições da pesquisa, as respostas para as perguntas de pesquisa, além dos trabalhos futuros a serem realizados.

2. REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos fundamentais que embasam esta tese. Nesse sentido, são descritas as bases envolvendo SDN. Em seguida, é apresentada a especificação Openflow, o protocolo *de facto* de um dos principais componentes da arquitetura SDN.

2.1 Redes definidas por software

As redes de comunicação estão crescendo em tamanho e complexidade em um ritmo cada vez maior, assim como a infra-estrutura, sistemas de rede e pilha de protocolos, que dificilmente fornecem soluções adequadas para as demandas de redes contemporâneas [189]. Isso provocou o surgimento de uma abordagem diferente para a arquitetura de sistemas de rede, denominada Rede Definida por Software (*Software-defined Networking* - SDN).

De acordo com a definição oficial da *Open Networking Foundation*, SDN é uma arquitetura de rede emergente, onde o controle da rede é desacoplado do encaminhamento de pacotes e diretamente programável. Essa migração de controle, antes estreitamente ligado a dispositivos de redes individuais, para dispositivos de computação acessíveis (como um computador, por exemplo), habilita a infraestrutura subjacente (como roteadores e *middleboxes*) ser abstraída para aplicações e serviços de rede, que podem tratar a rede como uma entidade lógica ou virtual [425].

Em uma arquitetura de rede tradicional, as redes de computadores podem ser divididas em três planos de funcionalidade: o plano de dados, de controle e de gerenciamento; o plano de dados corresponde aos dispositivos de rede, que são responsáveis por encaminhar, de forma eficiente, os dados na rede; o plano de controle representa os protocolos utilizados para popular as tabelas de encaminhamento dos elementos do plano de dados; o plano de gerenciamento inclui os serviços de software, como as ferramentas baseadas no SNMP (*Simple Network Management Protocol*) [459], por exemplo, usadas para monitorar remotamente e configurar as funcionalidades de controle. No plano de gerenciamento também são definidas as políticas de rede, onde o plano de controle reforça essas políticas e o plano de dados as executa, encaminhando os dados [302].

Em redes tradicionais, os planos de controle e dados são fortemente acoplados, embarcados no mesmo dispositivo de rede e toda a estrutura é altamente descentralizada, como ilustrados na Figura 2.1, onde cada componente de um plano tem o seu respectivo componente nos demais planos, em uma proporção praticamente unitária. Esta característica foi considerada importante para o projeto da Internet no período inicial, pois parecia

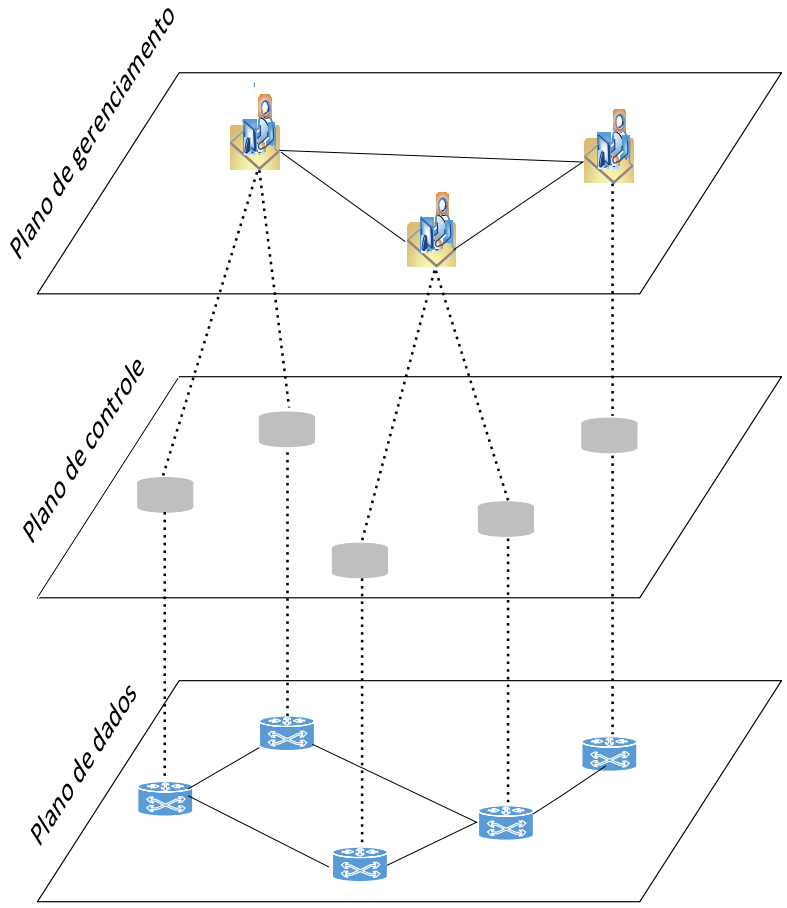

Figura 2.1 – Visão em camadas das funcionalidades de rede tradicional[302]

ser a melhor maneira de garantir resiliência. Realmente, esta abordagem foi efetiva em termos de desempenho, em virtude do rápido aumento de conexões. Entretanto, o resultado é uma arquitetura muito complexa e relativamente estática ([72] [469] [191] [290] [435]). Este resultado é o motivo pelo qual as redes tradicionais são rígidas e complexas de gerenciar e controlar. Esta rigidez e complexidade são responsáveis por uma indústria integrada verticalmente, dificultando a possibilidade de inovação [302].

Em virtude dos problemas enfrentados nas redes tradicionais, as empresas oferecem soluções próprias, compostas por hardware, sistema operacional e aplicativos de rede específicos, exigindo que os profissionais da área adquiram e mantenham diferentes soluções de gerenciamento, assim como equipes especializadas em cada solução. Essa demanda implica em alto custo de projeto, implantação e manutenção de uma infraestrutura de rede, com longos ciclos de retorno de investimento. Esses problemas relatados são agrupados no termo *ossificação* [371], que reflete o alto acoplamento envolvendo os planos de dados e controle. Isto significa que as decisões sobre o fluxo de dados através da rede são tomadas dentro de cada dispositivo de rede. Nesse ambiente, a implantação de novas aplicações ou funcionalidades não ocorre de forma trivial, necessitando ser implementada diretamente dentro da infraestrutura [418].

Neste cenário, mesmo a realização de tarefas rotineiras, como a implementação de configurações ou definição de políticas, podem requerer esforços significativos para viabilizar a sua aplicabilidade, devido a falta de uma interface de controle comum a todos os dispositivos de rede. Como alternativas, têm sido usadas soluções, como o uso de *middleboxes* (*Firewall*, Sistema de Detecção de Intrusão (IDS - *Intrusion Detection System*), Sistema de Prevenção de Intrusão (IPS - *Intrusion Prevention System*), tradutores de endereços de rede, etc.) superpostos acima da infraestrutura de rede, visando contornar o efeito da ossificação da rede, tendo como um exemplo de alternativa as CDNs (*Content Delivery Networks*) [443]. Por outro lado, a SDN foi desenvolvida para facilitar a inovação e habilitar um controle simples, de forma programável, do tráfego de dados na rede. Como visualizado na Figura 2.2 (as linhas sólidas representam as ligações do plano de dados e as linhas tracejadas as do plano de controle), a separação entre o hardware de encaminhamento de dados e a lógica de controle permite, de forma mais simples, a implantação de novos protocolos e aplicações, a visualização e gerenciamento da rede e a consolidação de vários *middleboxes* em controles via software, através do Controlador. Ao invés de impor políticas e protocolos que funcionam em uma convolução de dispositivos dispersos, a rede é reduzida para um hardware de encaminhamento "simples"e o Controlador responsável pela tomada de decisões [418].

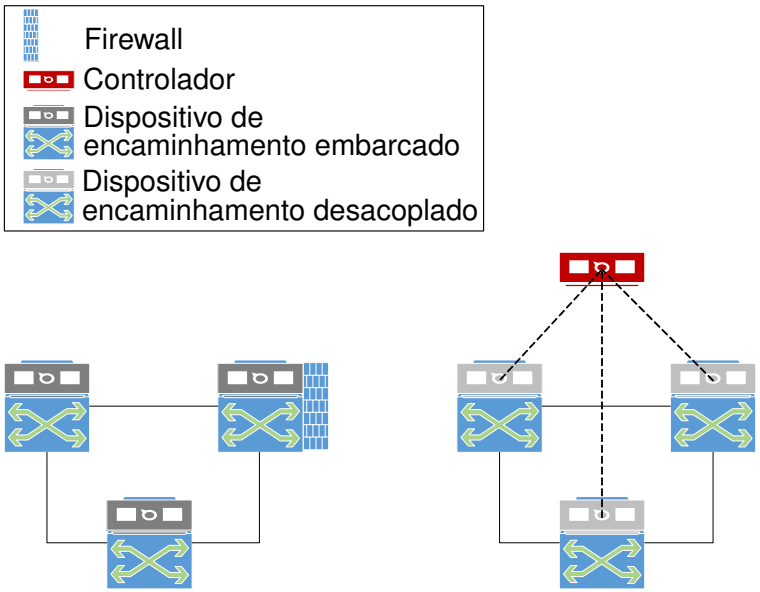

Figura 2.2 – Arquitetura tradicional vs. Arquitetura SDN. [418]

Nesse sentido, como ilustrado na Figura 2.3, uma SDN pode ser estruturada pelo Controlador e pelos dispositivos de encaminhamento de dados. O Controlador atua como um sistema operacional de rede, agindo como um intermediário entre os aplicativos de rede e os dispositivos. Exemplos de Controladores são POX [458], NOX [202], Opendaylight [372], Ryu [468], HPE [560], Cisco [119, 120] e Floodlight [387]. Esta estrutura permite que os dados do plano de controle sejam fornecidos aos aplicativos de rede e um certo nível de programabilidade através do uso de *plug-ins* entre as funções do Controlador e os protoco-

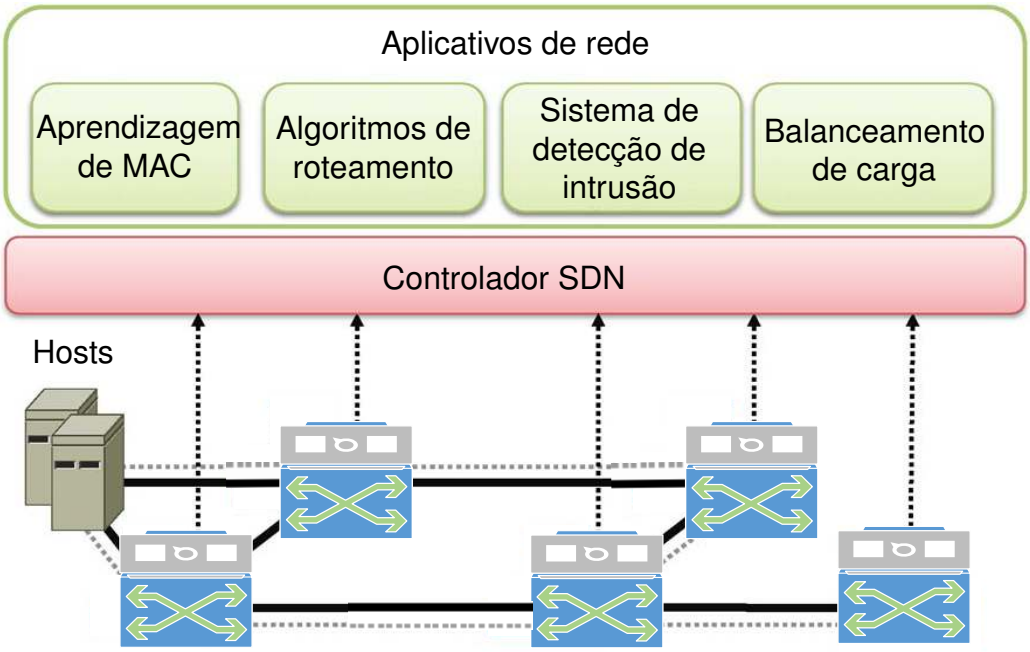

Figura 2.3 – Arquitetura SDN [302]

los de rede. Isso permite que programar esses aplicativos seja uma tarefa mais fácil, uma vez que as abstrações providas pela plataforma de controle e pelas linguagens específicas de domínio de SDN (como Pyretic [470] e Frenetic [172], por exemplo) possam ser compartilhadas. Com isso, todos os aplicativos podem tirar vantagem das mesmas informações da rede (tendo assim a mesma visão global da rede) conduzindo assim a decisões de políticas mais consistentes e efetivas. Esses aplicativos podem realizar ações (por exemplo, reconfigurar os dispositivos de encaminhamento) em qualquer parte da rede. Sendo assim, não há necessidade de conceber uma estratégia precisa sobre a localização de uma nova funcionalidade. Dessa forma, a integração de diferentes aplicativos fica mais direta. Por exemplo, os aplicativos de balanceamento de carga e roteamento podem ser combinados sequencialmente, com as decisões de balanceamento de carga tendo precedência sobre políticas de roteamento.

Para que as funcionalidades dos aplicativos sejam efetivadas, é necessário que as informações necessárias e/ou providas por esses aplicativos permeie os diferentes níveis de abstração, desde os aplicativos até os dispositivos de rede. Para isso, a SDN possui uma arquitetura formada por camadas, e essas camadas são interligadas por mecanismos de comunicação. A figura Figura 2.4 ilustra a arquitetura SDN. Essa arquitetura é formada pela camada de infraestrutura, composta pelos dispositivos de rede, a camada de controle, formada pelo Controlador ou pelos Controladores, e pela camada de aplicação, formada pelos aplicativos de rede.

Na camada de controle formada por um único Controlador, denomina-se que essa camada é logicamente centralizada e fisicamente centralizada. Na camada de controle formada por Controladores distribuídos, cada Controlador tem a visão de seu domínio de

atuação. Por exemplo, considerando uma rede formada por um Controlador (c1) e dois *switches* (s1 e s2) e uma outra rede formada por um outro Controlador (c2) e dois outros *switches* (s3 e s4). Cada uma dessas redes é um domínio de atuação. Considere que essas redes estão interligadas pelos *switches* s2 e s3. O domínio de atuação do Controlador c1 se limita aos *switches* s1 e s2, fazendo com que esse Controlador tenha uma visão limitada de toda a rede. O Controlador só pode administrar de forma centralizada a rede quando tem uma visão de toda a rede. Sendo assim, os Controladores necessitam ter um estado global da rede, que é a representação do estado da rede através de múltiplos domínios. Para construir um estado global da rede, cada Controlador precisa distribuir parte do estado local da sua rede com os outros Controladores. Essa distribuição gera o estado global da rede. Com isso, a camada de controle formada por múltiplos Controladores é denominada logicamente centralizada e fisicamente distribuída [421]. Nesse sentido, a camada de controle formada por múltiplos Controladores pode ser definida basicamente como uma estrutura vertical (também denominada hierárquica), onde um ou alguns Controladores possuem o estado global da rede e uma estrutura horizontal (também denominada plana) onde todos os controladores possuem o estado global da rede. Para que a camada de controle se comunique com as demais camadas, ou que os Controladores se comuniquem entre si, dentro da camada de controle, a arquitetura SDN é composta por mecanismos de comunicação, os quais são:

- *Northbound*: Esta comunicação ocorre entre os elementos das camadas de aplicação (os aplicativos de rede) e controle (Controlador ou Controladores), através de APIs (*Application Programming Interface*). Essas APIs são categorizadas em APIs *ad-hoc* de baixo nível, baseadas em *Web services* e baseadas em linguagens específicas de domínio. As APIs *ad-hoc* de baixo nível são fortemente dependentes da plataforma do Controlador. Essas APIs permitem que os desenvolvedores implementem aplicativos diretamente no Controlador, no formato de módulos, de forma fortemente acoplada ao Controlador, e desenvolvidos na linguagem de programação nativa do Controlador. As APIs do *Northbound* baseadas em *Web Services* são providas normalmente na arquitetura REST [367]. Esses *Web Services* permitem que aplicações externas e independentes ao Controlador (clientes) acessem as funcionalidades e serviços disponibilizados pelo Controlador (servidor). Por fim, as APIs baseadas em linguagens específicas de domínio são APIs com alto nível de abstração que utilizam o conceito de linguagens de programação específicas de domínio (*Domain Specific Language* - DSL) como uma forma indireta de aplicativos interagirem com o Controlador. Essas APIs são projetadas para aumentar o nível de abstração, com o objetivo de permitir o desenvolvimento flexível de aplicativos e a especificação de políticas de rede de alto nível [62].

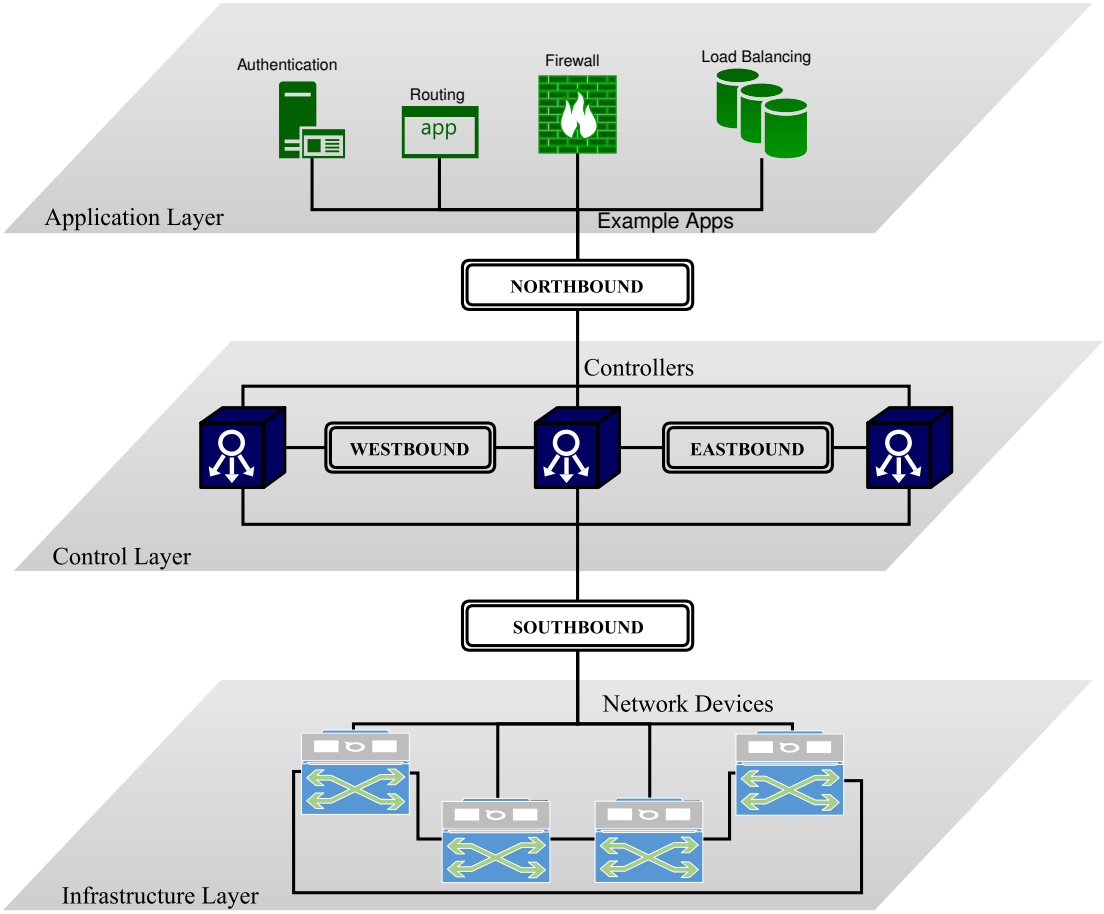

Figura 2.4 – Arquitetura SDN

- *East/West-bound*: Esta comunicação ocorre para o gerenciamento de uma arquitetura SDN distribuída, onde várias instâncias de Controladores necessitam comunicar-se frequentemente, compartilhando informações de controle e gerenciamento.

- *Southbound*: Esta comunicação ocorre entre a camada de controle e os dispositivos de rede. Através desta interface de comunicação, as instruções do Controlador são transformadas para instruções de baixo nível. Exemplos de possibilidades de comunicação no Southbound incluem NetConf [442], LISP [477], ForCES [209], POF [546], OpFlex [539], e Openflow [371].

2.1.1 NETCONF

NETCONF é um protocolo de gerenciamento de rede que foi padronizado pela IETF e publicado em 2006. Surgiu como uma necessidade dos operadores, porque os protocolos anteriores que foram projetados para serem usados na configuração de dispositivos eram difíceis de usar para gerenciamento, e eram usados principalmente para monitoramento de rede. Assim, os operadores eram obrigados principalmente a configurar seus

dispositivos na rede por outros meios, como o CLI (*Command Line Interface*). Como dispositivos de diferentes fornecedores podem coexistir na mesma rede, seu gerenciamento se tornou uma tarefa difícil para os operadores. YANG (*Yet Another Next Generation*) é uma linguagem de modelagem projetada especificamente para NETCONF. Ele possui uma abordagem orientada a objetos e um formato legível, que ajuda a descrever o modelo de dados usado no protocolo NETCONF [551].

Os operadores podem, usando o protocolo NETCONF, escrever dados de configuração em um dispositivo de rede, assim como recuperar os dados de um dispositivo de rede. Todos os dados são codificados utilizando XML (*eXtensible Markup Language*) e transmitidos via chamadas de procedimento remotas (RPC - *Remote Procedure Calls*), através de *sockets* ou TLS. O NETCONF é estruturado em conjuntos de dados de configuração, conhecidos como armazenadores de dados (*Data Stores*). São três tipos de armazenadores de dados possíveis: *Running*, *Candidate*, e *Startup*. O armazenador de dados *Running* contém as configurações atualmente em uso pelo dispositivo de rede. Alguns dispositivos possuem o armazenador de dados *Startup*, que contém os dados de configuração de quando o dispositivo iniciou o seu funcionamento. O armazenador de dados *Candidate* é uma funcionalidade do dispositivo opcional. Quando disponível, esse local armazena um conjunto de dados de configuração que o Controlador pode usar para atualizar o armazenador de dados *Running*, e assim modificar a operacionalidade do dispositivo.

A comunicação entre o Controlador e o dispositivo de rede ocorre no formato cliente-servidor, onde o cliente é o Controlador e o servidor é o dispositivo de rede. O NETCONF não possui um mecanismo padrão para configurar os dispositivos, ficando a critério de cada fabricante de dispositivos. Para suportar dispositivos de múltiplos fabricantes na rede, o suporte às funcionalidades dos dispositivos devem ser implementados no Controlador. O que o NETCONF oferece é um conjunto de mensagens para que o Controlador e o dispositivo possam trocar os dados de configuração das funcionalidades. Por exemplo, o comando *get-config* requisita os dados de um armazenador de dados, o comando *edit-config* modifica as configurações em um armazenador de dados, *get* requisita os dados de configuração do armazenador de dados *Running*, assim como os dados do estado do dispositivo.

2.1.2 LISP

O *Locator/ID Separation Protocol* (LISP) LISP oferece um desacoplamento entre os plano de dados e controle, programabilidade de rede e um controle centralizado através de um sistema de mapeamento e uma série de componentes. Como vantagens, o LISP foi projetado para ser implantável de forma incremental e alavancar as redes atuais baseadas em IP. Qualquer rede IP existente pode incorporar recursos SDN comuns simplesmente

atualizando alguns roteadores para os roteadores de túnel LISP e conectando-os a um sistema de mapeamento. Além disso, as deficiências dos protocolos SDN tradicionais estão motivando o surgimento de propostas híbridas SDN que combinam SDN com soluções tradicionais de rede. Curiosamente, devido à sua escalabilidade e interoperabilidade, o LISP facilita a implementação das redes SDN híbridas mencionadas acima, especialmente porque o LISP pode ser implementado de forma incremental. Além disso, graças à sua flexibilidade, o LISP é adequado para acomodar protocolos futuros e novas abordagens de rede. Por fim, em contraste com os protocolos SDN comuns projetados para operar principalmente em um único domínio, o LISP permite que as diretivas SDN sejam aplicadas em todos os domínios. Os elementos LISP bem posicionados tornam possível uma implantação de SDN programável em uma rede de trânsito (por exemplo, a Internet), algo que é mais complexo de realizar com os protocolos SDN tradicionais [478].

O LISP divide o espaço de endereços em dois - o *Endpoint IDentifiers* (EID) e o *Route LOCator* (RLOC). Ao realizar essa separação, o LISP introduz um nível de referenciamento que permite que as redes especifiquem preferências para múltiplos *gateways* (chamados de *locators*). A disponibilidade de múltiplos *locators* para o mesmo destino aumenta a diversidade de caminhos disponíveis, uma vez que as redes de origem estão habilitadas a encaminhar o tráfego para um destino através de múltiplos *locators*. Isso permite a criação de um sistema de rotas baseadas no encapsulamento dos pacotes em uma camada extra, gerenciada através de um sistema de mapeamentos. Com isso é possível separar o plano de controle do plano de dados. Com isso, o sistema de mapeamento assume o papel equivalente a um Controlador e os *gateways* aos dispositivos de encaminhamento. Logo, a comunicação do plano de controle ocorre entre o sistema de mapeamento e os *gateways*, de forma a definir ou alterar uma rota dinamicamente.

2.1.3 ForCES

O ForCES (*Forwarding and Control Element Separation*) é composto por um *framework* e um protocolo, com o objetivo de padronizar a troca de informações entre o plano de controle e o plano de dados. Os elementos presentes no plano de controle são denominados CEs (*Control Elements* e os elementos do plano de dados são denominados FEs (*Forward Elements*).

Para isso, o ForCES define uma entidade composta por um ou mais CEs e um ou mais FEs, denominada NE (Network Element). Para as entidades externas ao NE, o NE representa um ponto de gerenciamento, ou uma entidade de processamento de pacotes. Nessa organização, os CEs se comunicam com os FEs através do protocolo ForCES.

A distribuição dos elementos do NE podem ser locais (dentro de um dispositivo) ou distribuídos, com isso o ForCES propõe uma abordagem mais flexível ao gerenciamento

de rede tradicional, sem alterar a arquitetura atual da rede, ou seja, sem a necessidade de um Controlador externo logicamente centralizado. Os planos de controle e dados são separados, mas podem ser mantidos no mesmo elemento de rede. No entanto, a parte de controle do elemento de rede pode ser atualizada em tempo real com *firmware* de terceiros.

Os FEs e CEs podem ser desenvolvidos em hardware ou virtuais, e cada FE virtual é criado usando vários componentes interligados, chamados de LFBs (Logical Function Blocks). Cada LFB implementa uma função específica, sendo capaz de receber, transmitir ou modificar os pacotes. O LFB é uma descrição em XML de todas as informações que devem ser trocadas entre o FE e o CE, de forma que um administrador pode dinamicamente modificar e controlar as regras e políticas de processamento de pacotes, através da programabilidade existente no ForCES.

Para realizar a comunicação entre os FEs e os CEs, o protocolo ForCES opera em duas camadas: uma camada denominada TML (*Transport Mapping Layer*) e outra camada denominada PL (*Protocol Layer*). A TML é responsável pelo canal de comunicação, podendo utilizar diferentes protocolos (como SCTP, IP, TCP, UDP, ATM, Ethernet, etc. [488]), e a PL é responsável pelas mensagens trocadas entre os envolvidos na comunicação.

A TML utiliza três canais de comunicação entre o FE e o CE, que são os canais de alta, média e baixa prioridade. Para cada tipo de prioridade, são definidos conjuntos de mensagens que devem ser transmitidos por cada canal de comunicação.

Em termos de segurança, a especificação do ForCES define que a TML deve fornecer um mecanismo de autenticação entre os CEs e os FEs, em nível de transporte, além de um mecanismo que garanta autenticação das mensagens da PL e um mecanismo para garantir a confidencialidade dos dados da PL. A escolha das funcionalidades de segurança que serão empregadas fica a cargo do administrador da rede. A especificação trata que essa escolha é oferecida para respeitar requisitos de desempenho, quando existentes. As opções são: Uso de IPsec [280], troca de chaves através de *Internet Key Exchange* (IKE) [222], uso de Encapsulating Security Payload (ESP) [279], proteção de integridade através de HMAC-SHA1-96 [357], e confidencialidade das mensagens através de AES-CBC com tamanho de chave de 128-bit [178].

2.1.4 OpFlex

Outra proposta de interface é o OpFlex [539]. Similar ao ForCES, uma das ideias por trás do OpFlex é distribuir parte da complexidade do gerenciamento da rede de volta aos dispositivos de encaminhamento, com o objetivo de melhorar a escalabilidade. O OpFlex foi projetado para permitir a troca de dados de um conjunto de objetos gerenciados, que é definido como parte de um modelo informacional. O próprio OpFlex não dita o modelo de informações, podendo ser usado com qualquer modelo abstrato baseado em árvore, no qual

cada nó da árvore tenha um identificador de recurso universal (URI - Universal Resource Identifier) associado a ele. O protocolo é projetado para suportar XML e JSON e para usar mecanismos padrão de chamada de procedimento remoto (RPC), como JSON-RPC sobre TCP. O uso de um canal seguro através de TLS também é recomendado.

O OpFlex apresenta uma arquitetura formada pelos componentes: Repositório de Políticas, Elemento de Política, Registro de *Endpoint* e Observador. O repositório de políticas (PR - *Policy Repository*) é uma entidade logicamente centralizada, contendo a definição de todas as políticas que regem o comportamento do sistema. A autoridade de política manipula solicitações de resolução de política de cada Elemento de Política. Um elemento de política (PE - *Policy Element*) é uma abstração lógica para um dispositivo físico ou virtual que implementa e aplica a política. Os elementos de política são responsáveis por solicitar partes da política, à medida que novos pontos se conectam, desconectam ou mudam de estado. Além disso, os elementos de política são responsáveis por mapear uma política abstrata em uma política concreta. Esse processo é uma operação local e pode funcionar de maneira diferente em cada dispositivo, desde que a semântica da política seja respeitada. O Registro de *Endpoint* (ER - Endpoint Registry) armazena o estado atual da operação de cada dispositivo conectado na rede (*Endpoint*). O ER recebe informações sobre cada *Endpoint*, oriundas do PE e, em seguida, pode compartilhá-lo com outros PEs no sistema. O ER pode estar fisicamente localizado, mas também pode ser distribuído na própria malha de rede, na forma de um banco de dados distribuído. Já o Observador coleta estatísticas, falhas e eventos de cada PE.

As mensagens do protocolo podem ser resumidas na seguinte forma: A *Identity* é a primeira mensagem entre um PR e um PE. A mensagem *Policy Resolution* retorna para um PE um conjunto de políticas oriundas de um PR. A mensagem *Policy Update* é enviada pelo PR para um PE quando a definição de uma política, envolvendo esse PE, foi alterada. A *Policy Trigger* é enviada entre PEs para que seja disparada uma resolução de políticas na PE de destino. A *Endpoint Declaration* indica que um novo *Endpoint* foi anexado na rede, sendo enviado do PE ao ER. A *Endpoint Request* realiza a consulta por um *Endpoint* usando um conjunto de identificadores, como o endereço MAC, por exemplo, e essa mensagem é enviada do PE ao ER. *A Endpoint Policy Update* é enviada pelo ER quando ocorre uma alteração relacionada a política do *Endpoint*. Por fim, a mensagem *State Report* é enviada por cada PE para o Observador com dados sobre falhas, eventos e estatísticas.

2.1.5 Openflow

Para o Controlador agir como um sistema operacional de rede, os dispositivos de rede necessitam de uma forma de se comunicarem com o Controlador. Nesse sentido,

como descrito por McKeown e outros [371], o Openflow foi proposto com o objetivo de padronizar a comunicação entre os *switches* e o Controlador no *Southbound* da arquitetura SDN, em virtude desse pioneirismo, o protocolo Openflow atraiu a atenção tanto da academia, quanto da indústria. Esse interesse fez com que o Openflow seja o protocolo *de facto* para Southbound em ambientes SDN [642].

No modelo SDN, a gerência do tráfego de informações abstrai o tradicional conceito de pacotes e opera orientada a fluxos. O Openflow surgiu em virtude da dificuldade da comunidade científica testar novas ideias no hardware de rede tradicional. Isto ocorre porque os códigos-fonte dos softwares que são executados pelos *switches* não poderem ser modificados de forma simples, além da infraestrutura de rede estar ossificada. Isso não permite que novas ideias, referente ao campo de pesquisa de redes, sejam testadas em configurações realistas de tráfego.

Os criadores do Openflow identificaram características em comum nas tabelas de fluxos (*Flow Tables*) dos *switches* Ethernet, e desenvolveram uma especificação de protocolo padrão para controlá-las através de software. Este protocolo provê um meio de controlar um *switch* sem necessitar que os fabricantes exponham os códigos de seus dispositivos. Dessa forma, um administrador de rede pode particionar o tráfego em fluxos de produção e de pesquisa, por exemplo, permitindo que os pesquisadores possam controlar seus próprios fluxos, definindo as rotas em que seus pacotes irão trafegar e como serão processados. Dessa forma, os pesquisadores podem testar novos protocolos, modelos de segurança, esquemas de endereçamento, ou mesmo alternativas ao protocolo TCP/IP. Nessa mesma rede, o tráfego de produção é isolado e processado da mesma forma que ocorre atualmente, porém o tráfego de pesquisa também pode ser processado e isolado, sem comprometer a infraestrutura [371].

O Openflow consiste em três componentes principais [312]: Um *switch* compatível com Openflow, um canal de comunicação e um Controlador. O *switch* utiliza um *pipeline* de tabelas de fluxos para encaminhar pacotes, numeradas sequencialmente, como ilustra a Figura 2.5. Um pacote pode ser eliminado em qualquer parte do *pipeline*, e a medida que os pacotes cruzam o pipeline, um conjunto de ações podem ser definidas. Uma tabela de fluxos é composta por uma lista de entradas de fluxos (*flow entries*). Cada entrada de fluxo é composta por regras de fluxo (*flow rules*), que possuem campos de correspondência (*match fields*), contadores e instruções da ação a ser realizada. A Figura 2.6 ilustra a estrutura de uma regra de fluxo.

Os pacotes que chegam no *switch* são comparados com os campos de correspondência de cada entrada e, se a entrada é correspondente a alguma entrada de fluxo, o pacote é processado, de acordo com a ação definida nas instruções deste fluxo. Os contadores são usados para manter as estatísticas sobre os pacotes. O pacote pode também ser encapsulado e enviado ao Controlador. O Controlador é responsável por gerenciar os

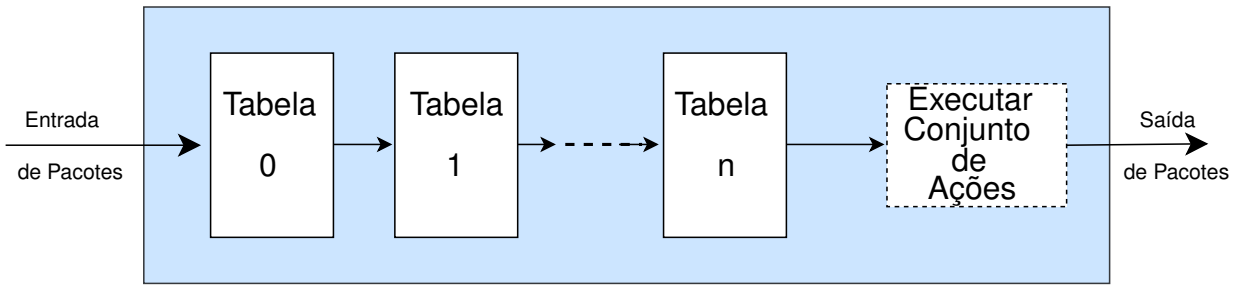

Figura 2.5 – Estrutura do *pipeline* de um *switch* Openflow

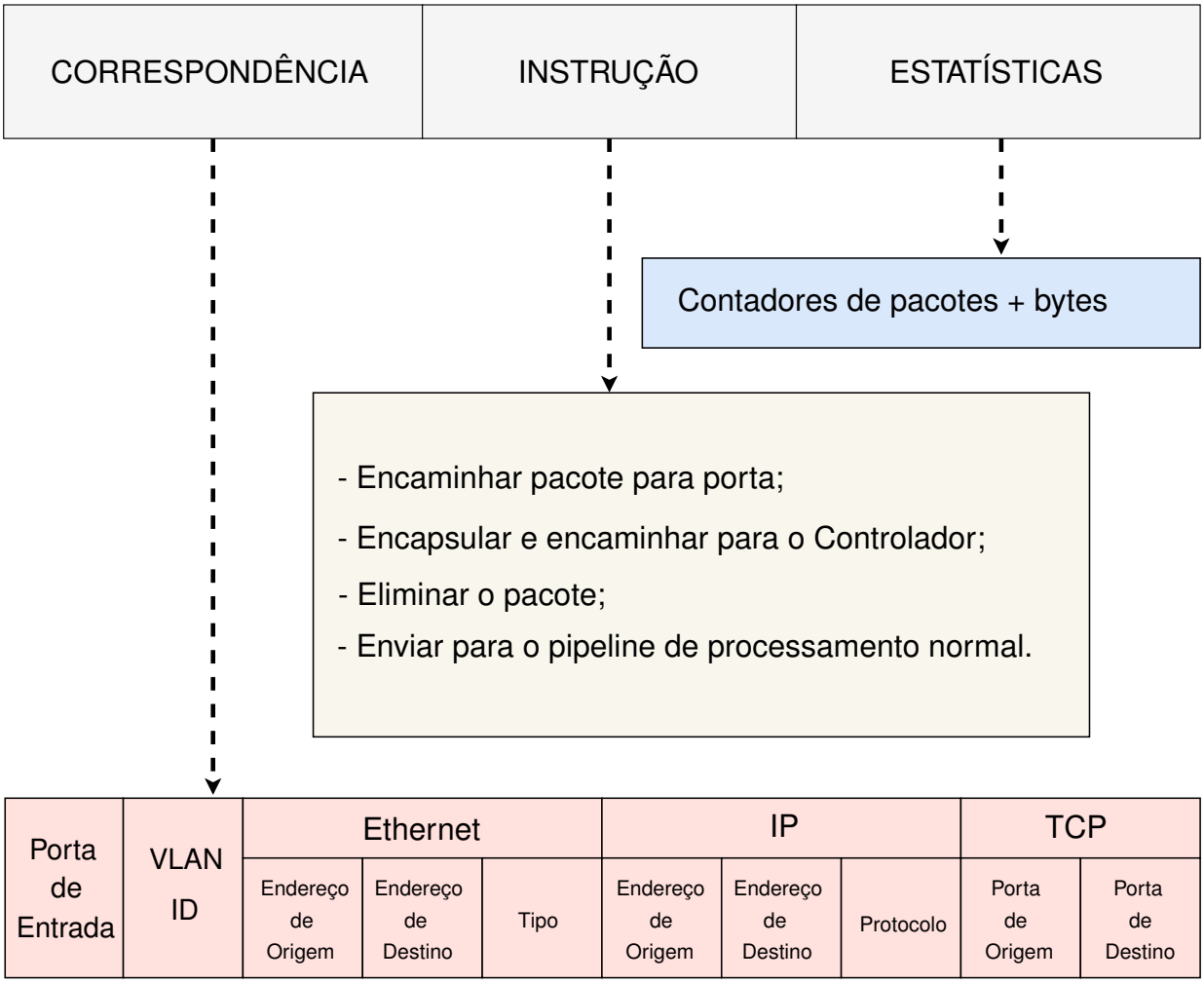

Figura 2.6 – Estrutura da regra de fluxo

switches (manipular as tabelas de fluxos), além de receber e enviar pacotes para estes equipamentos.

Um *switch* compatível com Openflow deve ser capaz de encaminhar pacotes de acordo com as regras de fluxos existentes na tabela de fluxos. Internamente, um *switch* desenvolvido exclusivamente em *hardware* usa as memórias TCAM (*Ternary Content Addressable Memory*) e um *switch* virtualizado usa as memórias RAM (*Random Access Memory*) para armazenar as tabelas de fluxos. Isso impacta diretamente no tamanho das tabelas de fluxos e no desempenho do processo de leitura e escrita das regras de fluxos, pois o switch

virtualizado pode alocar recursos de memória para o armazenamento dos fluxos, à medida que são necessárias, a partir de configurações. Por outro lado, embora a quantidade de memória do *switch* em hardware seja fixa, o desempenho da memória TCAM é superior à memória RAM.

Para que o protocolo Openflow seja utilizado, tanto um *switch* quanto o Controlador precisam possuir a implementação da especificação Openflow. Atualmente, a especificação mais atualizada encontra-se na versão 1.5 [175], mas quase a totalidade dos equipamentos disponíveis no mercado opera apenas na versão 1.0 [173] [418] [312] [231]. Esse fato impacta tanto nas funcionalidades (por exemplo uso de Controladores distribuídos, disponível a partir da versão 1.3, ou de novos tipos de mensagens) quanto em requisitos não funcionais, como segurança. Durante a evolução da especificação, várias falhas de segurança vem sendo corrigidas, o que aumenta a probabilidade dos dispositivos com versões mais antigas sofrerem ataques bem sucedidos.

Nesse sentido, a especificação Openflow suporta três tipos de mensagens: Controlador para *switch*, Assíncronas e Simétricas, cada tipo com múltiplos subtipos. A Tabela 2.1 apresenta todas as mensagens definidas nas especificações Openflow.

As mensagens Controlador para *switch* são inicializadas pelo Controlador e são usadas para gerenciar ou inspecionar diretamente o estado do *switch*. As mensagens assíncronas são iniciadas pelo *switch* e são usadas para atualizar o Controlador sobre eventos na rede e alterações no estado do *switch*. As mensagens simétricas são iniciadas tanto pelo *switch* quanto pelo Controlador, e são enviadas sem uma solicitação prévia.

As mensagens envolvidas nesses tipos são organizadas de forma a comporem comunicações de gerenciamento entre Controlador e *switch*. Por exemplo, para iniciar uma conexão, a mensagem simétrica OFPT_HELLO é enviada por ambos os envolvidos na comunicação. Nesse ponto, as versões Openflow suportadas por cada envolvido são apresentadas, e caso não haja compatibilidade de versões, é enviada uma mensagem simétrica OFPT_ERROR e encerrada a comunicação. Caso isso não ocorra, o Controlador envia uma mensagem OFPT_FEATURES_REQUEST (do tipo Controlador para *switch*) solicitando as informações básicas do *switch*. Ao receber essa mensagem, o *switch* responde essa solicitação com a mensagem de resposta OFPT_FEATURES_REPLY, contendo as informações sobre o *switch*. Embora a mensagem OFPT_FEATURES_REPLY seja enviada do *switch* para o Controlador, como esse envio só ocorre devido ao *switch* ter recebido previamente uma mensagem de requisição por parte do Controlador, todas as mensagens que se encaixam nessa condição fazem parte do tipo de mensagem Controlador para *switch*. Após esse processo, paulatinamente os envolvidos na comunicação trocam as mensagens simétricas OFPT_ECHO_REQUEST e OFPT_ECHO_REPLY, de forma a indicar a latência, largura de banda e/ou se uma conexão está ativa.

Enquanto a comunicação estiver em curso, várias situações podem ocorrer. Por exemplo, o Controlador está habilitado a definir e consultar os parâmetros de configuração

Tabela 2.1 – Tipos de mensagens OpenFlow

OFP 1.0	OFP 1.1	OFP 1.2	OFP 1.3	OFP 1.4	OFP 1.5
Mensagens de Controlador para *switch*					
OFPT_FEATURES_REQUEST					
OFPT_FEATURES_REPLY					
OFPT_GET_CONFIG_REQUEST					
OFPT_GET_CONFIG_REPLY					
OFPT_SET_CONFIG					
OFPT_PACKET_OUT					
OFPT_FLOW_MOD					
OFPT_PORT_MOD					
OFPT_BARRIER_REQUEST					
OFPT_BARRIER_REPLY					
—	OFPT_GROUP_MOD				
—	OFPT_TABLE_MOD				
—		OFPT_ROLE_REQUEST			
—		OFPT_ROLE_REPLY			
—			OFPT_GET_ASYNC_REQUEST		
—			OFPT_GET_ASYNC_REPLY		
—			OFPT_SET_ASYNC		
—			OFPT_METER_MOD		
OFPT_STATS_REQUEST			OFPT_MULTIPART_REQUEST		
OFPT_STATS_REPLY			OFPT_MULTIPART_REPLY		
OFPT_QUEUE_GET_CONFIG_REQUEST				—	
OFPT_QUEUE_GET_CONFIG_REPLY				—	
—				OFPT_BUNDLE_CONTROL	
—				OFPT_BUNDLE_ADD_MESSAGE	
Mensagens Assíncronas					
OFPT_PACKET_IN					
OFPT_FLOW_REMOVED					
OFPT_PORT_STATUS					
—				OFPT_ROLE_STATUS	
—				OFPT_TABLE_STATUS	
—				OFPT_REQUESTFORWARD	
—					OFPT_CONTROLLER_STATUS
Mensagens Simétricas					
OFPT_HELLO					
OFPT_ERROR					
OFPT_ECHO_REQUEST					
OFPT_ECHO_REPLY					
OFPT_VENDOR	OFPT_EXPERIMENTER				

do *switch*, com as mensagens OFPT_SET_CONFIG e OFPT_GET_CONFIG_REQUEST, respectivamente. Consequentemente, o *switch* envia os seus parâmetros de configuração com uma mensagem OFPT _GET_CONFIG_REPLY. Outra comunicação comum ocorre durante a configuração das tabelas de fluxos, onde as entradas de fluxo são adicionadas ou modificadas usando a mensagem OFPT_FLOW_MOD. O Controlador pode também configurar o estado dinâmico de uma tabela de fluxo usando a mensagem OFPT_TABLE_MOD.

Porém, quando o estado de uma tabela de fluxos altera, o Controlador precisa ser informado com uma mensagem OFPT_TABLE_STATUS.

Além das mensagens OFPT_FLOW_MOD e OFPT_TABLE_MOD, são disponibilizadas a mensagem OFPT_GROUP_MOD para realizar modificações nos grupos de tabelas, a mensagem OFPT_PORT_MOD para modificar o comportamento de uma porta do *switch* e a mensagem OFPT_METER_MOD, responsável por alterar um medidor (*Meter*). Se espera que o *switch* envie mensagens OFPT_PORT_STATUS sempre que o estado da configuração de uma porta seja alterado. Esses eventos incluem uma alteração no status da porta (se foi desligada diretamente por um usuário, por exemplo) ou como definido na especificação 802.11D [517]. Um medidor é um elemento do *switch* que pode medir e controlar a taxa de pacotes que transitam no *switch*. Quando um Controlador modifica o estado de grupos ou de medidores, a mensagem que modificou o estado pode ser encaminhada para outros controladores, caso a rede esteja organizada em controladores distribuídos. Para isso, os outros controladores são informados com a mensagem OFPT_REQUESTFORWARD.

Quando uma entrada de fluxo é adicionada no *switch*, em virtude de uma solicitação de modificação de fluxo, um valor de tempo de inatividade indica quando uma entrada de fluxo deve ser removida, assim como um valor de *hard timeout* indica quando a entrada de fluxo deve ser removida, independente de atividade. Quando isso ocorre, o *switch* comunica esse fato ao Controlador através de uma mensagem OFPT_FLOW_REMOVED. As solicitações de modificação de fluxos, que implicam em deletar fluxos, podem também causar o envio da mensagem assíncrona OFPT_FLOW_REMOVED.

Como algumas requisições ou respostas possuem uma grande quantidade de dados, a especificação Openflow define uma categoria de mensagens para codificar esses dados, os quais normalmente não seriam possíveis colocar em uma mensagem Openflow simples, que é limitada a 65KB. As mensagens OFPT_STATS_REQUEST ou OFPT_ MULTIPART_REQUEST (dependendo da versão da especificação Openflow) são usadas primordialmente para requisitar estatísticas. Caso ainda seja interessante conhecer as informações sobre o estado de um *switch*, isso pode ser obtido com as mensagens OFPT_STATS_REPLY ou OFPT_MULTIPART_REPLY (a escolha de qual dessas mensagens depende da versão da especificação Openflow). Da mesma forma, o Controlador pode criar, excluir ou executar um grupo de mensagens inter-relacionadas, no modelo de lote. Isso pode ser realizado através da mensagem OFPT_BUNDLE_CONTROL. Para o Controlador adicionar mensagens nesse lote, utiliza-se a mensagem OFPT_BUNDLE_ADD_MESSAGE. Como as mensagens de modificação de estados do *switch* podem ser executadas em uma ordem arbitrária, o Controlador pode definir pontos de sincronização dessas modificações, através da mensagem OFPT_BARRIER_REQUEST. Quando a execução da OFPT_BARRIER_REQUEST é concluída, o *switch* responde utilizando a mensagem OFPT_BARRIER_REPLY

Outra situação que ocorre frequentemente, durante uma comunicação Openflow, é quando o switch não identifica uma entrada de fluxo que um determinado pacote se en-

quadre. Quando isso ocorre (ou se um pacote corresponde a uma entrada de fluxo com a ação "Enviar para o Controlador"), o *switch* envia para o Controlador uma mensagem OFPT_PACKET_IN, enviando nessa mensagem o referido pacote, de forma ao Controlador analisá-lo e definir qual deve ser o seu destino. Por outro lado, quando um Controlador necessita enviar um pacote ao plano de dados, ele o faz através de uma mensagem OFPT_PACKET_OUT.

Embora o *switch* detenha um conjunto de mensagens o qual pode enviar ao Controlador sem uma solicitação prévia do Controlador (tipo de mensagens assíncronas), o Controlador pode configurar quais mensagens o *switch* poderá enviar de forma assíncrona. Para isso, o Controlador está habilitado a definir e consultar as configurações dessas mensagens usando a mensagem OFPT_SET_ASYNC e OFPT_GET_ASYNC_REQUEST, respectivamente. Consequentemente, o *switch* responde a uma solicitação OFPT_GET_ ASYNC_REQUEST com a mensagem OFPT _GET_ASYNC_REPLY.

Quando a rede está estruturada em um modelo de controladores distribuídos, qualquer alteração na hierarquia desses controladores precisam ser comunicadas aos *switches*. Para alterar no *switch* o papel de um determinado Controlador (mestre ou escravo), utiliza-se a mensagem OFPT_ROLE_ REQUEST. Quando o papel de um Controlador é alterado, o *switch* deve enviar uma mensagem OFPT_CONTROLLER_STATUS para todos os controladores conectados a este *switch*. Quando um controlador tem o seu papel alterado no *switch*, sem ter sido pelo próprio Controlador, o Controlador será informado pela mensagem OFPT_ROLE_STATUS.

Por fim, a especificação Openflow destinou um tipo de mensagens para que os fabricantes de equipamentos e soluções possam definir funcionalidades adicionais. Para isso, são destinadas as mensagens OFPT_VENDOR ou OFPT_EXPERIMENTER (dependendo da versão da especificação Openflow utilizada)

2.1.6 POF

O POF (*Protocol-Oblivious Forwarding*) tem como um dos principais objetivos aprimorar o atual plano de dados de SDN. Para atingir seu objetivo, o POF propõe um conjunto genérico de instruções de fluxo que abstrai o protocolo do plano de dados. Um dispositivo de encaminhamento não precisa saber, por si só, nada sobre o formato do pacote antecipadamente. Assim, dispositivos de encaminhamento são vistos como caixas brancas com apenas recursos de processamento e encaminhamento. No POF, a análise de pacotes é uma tarefa do Controlador, que resulta em uma sequência de chaves genéricas e instruções de consulta de tabela instaladas nos dispositivos de encaminhamento. O comportamento dos dispositivos do plano de dados é, portanto, completamente sob o controle do Controlador. Semelhante a uma CPU em um sistema de computador, um dispositivo de

encaminhamento é agnóstico em termos de aplicativos e protocolos. Dessa forma, o controle da rede é formado pelo Controlador, os dispositivos de encaminhamento compatíveis e um canal de comunicação.

O dispositivo de encaminhamento não precisa entender o formato do pacote. Tudo o que precisa fazer, sob a instrução de seu Controlador, é extrair e montar as chaves de busca do cabeçalho do pacote, conduzir as pesquisas na tabela e, em seguida, executar as instruções, sendo essas instruções independentes de protocolos. Como resultado, o dispositivo de encaminhamento poderá suportar facilmente novos protocolos e requisitos de encaminhamento de pacotes.

Para alcançar isto, os metadados de pacotes são expandidos em um bloco associado a cada pacote no *pipeline* de processamento. Com isso, o Controlador pode usar esse bloco livremente para armazenar dados temporários (por exemplo, porta de entrada e os campos de cabeçalho do pacote, resolvidos até o momento) durante o tempo de vida de um pacote.

O processamento de pacotes no POF são baseados em uma sequência de chaves de busca e instruções em tabelas de fluxos. A chave de busca define os campos de correspondência para executar determinadas instruções em um determinado fluxo. A chave de busca de um campo de correspondência é definida por uma tupla <deslocamento, tamanho>, onde deslocamento indica a localização do bit inicial de um campo em um pacote e o tamanho informa o tamanho do campo em bits. Por exemplo, um campo de endereço IPv4 de um pacote poderia ser representado como <208,32> indicando que o início do pacote é acessado através do bit 208 e possui um tamanho de 32 bits. O POF também inclui um conjunto de instruções genéricas de fluxos POF-FIS (FIS - *Flow Instruction Set*), de forma a facilitar que um dispositivo de encaminhamento possa converter, editar e encaminhar os pacotes [651]. A Tabela 2.2 ilustra o conjunto de instruções.

Tabela 2.2 – Tipos de instruções POF

Categoria	Instruções
Edição	SET_FIELD, ADD_FIELD, DEL_FIELD, ALG, CALCULATE_CHECKSUM, SET_FIELD_UPDATE_CHECKSUM, INC_FIELD, DEC_FIELD, AND_FIELD, OR_FIELD. SRL_FIELD, SLL_FIELD, XOR_FIELD, NOR_FIELD, NOT_FIELD
Encaminhamento	GOTO_TABLE, COUNTER, OUTPUT, GROUP, MOVE_PACKET_OFFSET, SET_PACKET_OFFSET
Entrada	SET_TABLE_ENTRY, ADD_TABLE_ENTRY, DEL_TABLE_ENTRY
Salto	BRANCH, COMPARE, JUMP
Fluxo	SET_FLOW_METADATA, GET_FLOW_METADATA, ORDER_ENFORCE

A categoria de instruções denominada Edição são usadas para editar os dados do pacote. A edição de dados por pacote é a parte mais importante durante o processo de encaminhamento, pois quase todas as regras do protocolo precisam editar os dados do

pacote, como escrever, armazenar, copiar e calcular. O comando SET_FIELD, por exemplo, define qualquer campo de pacote com qualquer valor, como o endereço MAC de destino no cabeçalho Ethernet. Já os comandos ADD_FIELD e DEL_FIELD podem inserir ou excluir um campo personalizado nos dados de pacote ou a partir dos dados do pacote. Essas são as três instruções mais úteis.

Usando essas três instruções, os usuários podem definir um campo totalmente personalizado, como um campo denominado Administração e Manutenção da Operação (OAM - *Operation Administration and Maintenance*). Quando um pacote IPv4 entrar na rede local, o campo OAM pode ser inserido nos dados do pacote pela instrução ADD_FIELD no *gateway* de ingresso. Dentro da rede local, o campo OAM pode ser gravado com qualquer valor. O campo OAM pode ser excluído no *gateway* de saída, se o pacote for enviado da própria rede local, de forma a se tornar um pacote IPv4 normal. O uso do campo OAM é bastante amplo, incluindo para ações de firewall, troca de etiquetas, correspondência de prioridades, estatísticas e assim por diante. Usando os comandos de edição, também pode ser decidido livremente onde o campo OAM está localizado, ou quanto tempo o campo OAM deve existir, etc.

As demais instruções de edição, como ALG, INC_FIELD, DEC_FIELD, CALCULATE_CHECKSUM e algumas outras operações lógicas, são todos tipos de cálculos dos dados de pacote. O ALG pode fazer alguma aritmética, como cálculo de *hash*. O campo *Time-To-Live* (TTL) no cabeçalho IPv4 pode ser diminuído em 1, usando a instrução DEC_FIELD, durante o encaminhamento IPv4. Todas as operações em nível de bits podem ser manipuladas pelas instruções lógicas.

A categoria de instruções de Encaminhamento são usadas para encaminhamento de pacotes. Todo o encaminhamento de um pacote pode conter vários processos. Os processos podem ser separados em algumas tabelas de fluxo diferentes, de acordo com a funcionalidade. Quando o processamento em uma tabela de fluxo é concluído, pode ser executada a instrução GOTO_TABLE para enviar os dados de pacote da tabela de fluxo anterior para a próxima tabela de fluxo. Já a instrução COUNTER pode contar o número de pacotes que já foram manipulados. A instrução OUTPUT envia os dados do pacote para fora dos elementos da rede, através da especificação de uma porta. Enquanto isso, pode ser decidido onde o pacote começa, e se envia os metadados antes dos dados do pacote ou não. A instrução GROUP é usada para multicast. Por fim, as instruções MOVE_PACKET_OFFSET e SET_PACKET_OFFSET podem mover os ponteiros base dos pacotes para frente ou para trás ou para um local especificado. Essas duas instruções são muito úteis para manipular os pacotes em diferentes camadas. Por exemplo, usando a instrução SET_PACKET_OFFSET, os usuários podem definir o deslocamento do pacote para 112 bits, que é a posição inicial do cabeçalho IPv4 em um pacote Ethernet normal. Não importa em qual camada do protocolo os dados estejam, esses dados sempre podem ser manipulados, usando essas duas instruções relacionadas a deslocamento.

Já as instruções da categoria Entrada permitem que os elementos de rede operem a entrada de fluxo de forma independente. A instrução SET_TABLE_ENTRY define o parâmetro e as informações de correspondência das entradas de fluxo. Az instruções ADD_TABLE_ENTRY e DEL_TABLE_ENTRY podem inserir uma nova entrada de fluxo em uma tabela de fluxo ou excluir uma entrada de fluxo existente de uma tabela de fluxo. A operação da entrada de fluxo pelo elemento de rede é muito útil para as regras de protocolo que precisam estudar as informações sobre a rede, como topologia, roteamento e vizinhança.

Por fim, as categorias Salto e Fluxo são compostas por instruções avançadas. As instruções de Salto estão habilitadas a alterar o processamento de dados de pacote. As instruções de Fluxo provêm algumas operações sobre o *status* global do fluxo de dados. Para prover a programabilidade da rede, com um alto nível de abstração, as instruções podem ser programadas na camada de gerenciamento do modelo SDN, com linguagens de descrição de alto nível, como P4 [82], C [171] e Frenetic [172].

2.2 Considerações

Com base nas informações apresentadas durante este capítulo, pode-se resumir a arquitetura SDN na representação de quatro pilares [302]:

1. Os planos de dados e controle são desacoplados. As funcionalidades de controle são removidas dos dispositivos de rede, que torna os equipamentos simples elementos de encaminhamento de pacotes.

2. As decisões de encaminhamento são baseadas em fluxos, ao invés de serem baseadas no destino. Um fluxo é composto por um critério de filtro e um conjunto de ações. Esse critério analisa os dados de um conjunto de campos existentes nos pacotes, e se os dados corresponderem ao critério, o filtro é aplicado e as ações são executadas. No contexto da SDN, um fluxo é composto por uma sequência de pacotes trafegando entre uma origem e um destino. Os dispositivos de encaminhamento aplicam as mesmas políticas de serviço para todos os pacotes de um mesmo fluxo [202]. A abstração em fluxos permite unificar o comportamento de diferentes tipos de dispositivos de rede, incluindo roteadores, *switches*, Firewalls, e Middleboxes [249].

3. A lógica de controle é movida para uma entidade externa, o Controlador SDN ou Sistema Operacional de Rede (NOS). O Controlador é uma plataforma de software e provê os recursos e abstrações necessárias para facilitar a programação de dispositivos de encaminhamento. Esta estrutura é baseada em uma visão da rede abstrata e logicamente centralizada. A proposta do Controlador é similar a de um sistema operacional tradicional. A centralização da lógica de controle oferece vários benefícios.

Em primeiro lugar, como a modificação de políticas de rede ser mais simples e menos susceptível a erros, se comparada às configurações específicas de dispositivos de baixo nível. Esta melhoria ocorre porque as modificações são realizadas através de linguagens de alto nível e componentes de software. Em segundo lugar, um programa de controle pode automaticamente reagir a alterações espúrias nos estados da rede, mantendo as políticas de alto nível intactas. Em terceiro, a centralização da lógica de controle, em um Controlador com conhecimento global do estado da rede, simplifica o desenvolvimento de funções, serviços e aplicações de rede mais sofisticadas.

4. A rede é programável através de aplicativos de software, sendo executados em cima do Controlador que, por sua vez, interage com os dispositivos de rede.

Como apresentado, a arquitetura SDN pode ser organizada em camadas (aplicação, controle e infraestrutura), interligadas por meio de canais de comunicação (Northbound, Southbound e East-Westbound). Esses canais são utilizados por meio de protocolos de comunicação. Foram comentados os principais protocolos disponíveis na literatura.

Em virtude da situação pouco madura do paradigma SDN, percebeu-se a necessidade de desenvolver um estudo mais aprofundado sobre as questões relacionadas à segurança em SDN, de forma a embasar a justificativa desta tese. Sendo assim, foi realizado um estudo envolvendo um mapeamento sistemático, tópico que será apresentado e discutido no Capítulo 3.

3. MAPEAMENTO SISTEMÁTICO

Para auxiliar no processo de elaboração da proposta de tese, foi realizado um estudo de mapeamento sistemático (SMS - *Systematic Mapping Study*) [448] sobre o atual estado da segurança em SDN. Um SMS é uma metodologia comum para auxiliar na condução de pesquisas, como na área médica e, desde 2007, tem crescido substancialmente a aplicabilidade na área da Computação, especialmente em Engenharia de Software [59].

De acordo com Petersen [448], um SMS provê uma estrutura de categorização, oferecendo um sumário visual, o mapeamento e seu resultados. Isso requer menos esforços, ao passo que provê uma visão geral com uma melhor seleção das informações.

Inicialmente, o SMS foi voltado a engenharia de software e era recomendado normalmente para pesquisar áreas onde havia um leque de estudos em estágios iniciais relevantes e de alta qualidade. Como a segurança em SDN está no estágio inicial e estudos de alta qualidade estão emergindo, o SMS pode contribuir para categorizar esses estudos.

O processo de mapeamento sistemático, como ilustrado na Figura 3.1, inicia com a definição das questões de pesquisa (1), por exemplo, "Quais são as principais vulnerabilidades em SDN?". O processo de SMS deve identificar estudos em fase inicial, de forma a direcionar as questões de pesquisa. Esse passo resulta no escopo da pesquisa (2). Uma questão crítica em um SMS é realizar a pergunta certa. Nesse sentido, a pergunta correta é normalmente uma pergunta que seja significativa e importante para os pesquisadores, assim como para os demais profissionais da área. Esse aspecto precisa ser considerado, pois os pesquisadores podem estar interessados se descobrir, por exemplo, se uma técnica de mitigação de DoS é efetiva no contexto SDN, mas a indústria pode estar interessada em saber quais são as técnicas existentes, a fim de elaborar um mecanismo de defesa híbrido. Além disso, deve conduzir a alterações no *status quo* da área ou aumentar a confiança nas práticas atualmente desenvolvidas. Por exemplo, a academia e a indústria podem estar interessadas em descobrir em quais condições um Controlador consegue ser resiliente a um ataque de DoS e em quais condições o Controlador não consegue suportar esse ataque, de forma a reforçar ou alterar as práticas de defesa atualmente realizadas. Por fim, a pergunta deve identificar discrepâncias entre crenças comumente concebidas e a realidade. Por exemplo, a comunidade pode acreditar que segurança em SDN é um campo de pesquisa pouco explorado, e ao realizar o SMS perceber que é um *hot topic*.

Em seguida, é necessário definir como conduzir a busca por artigos. Para realizar a busca, é necessário estruturar *strings* de busca e definir as bases de dados que serão consultadas (3). Kitchenham [292] sugere estruturar as *strings* de busca baseado na população, intervenção, comparação e efeito. A população define o escopo geral da pesquisa, podendo ser um assunto genérico ou específico. Em SMS voltado à medicina, a população é definida com o objetivo de reduzir o número de estudos em fase primária. Porém, como

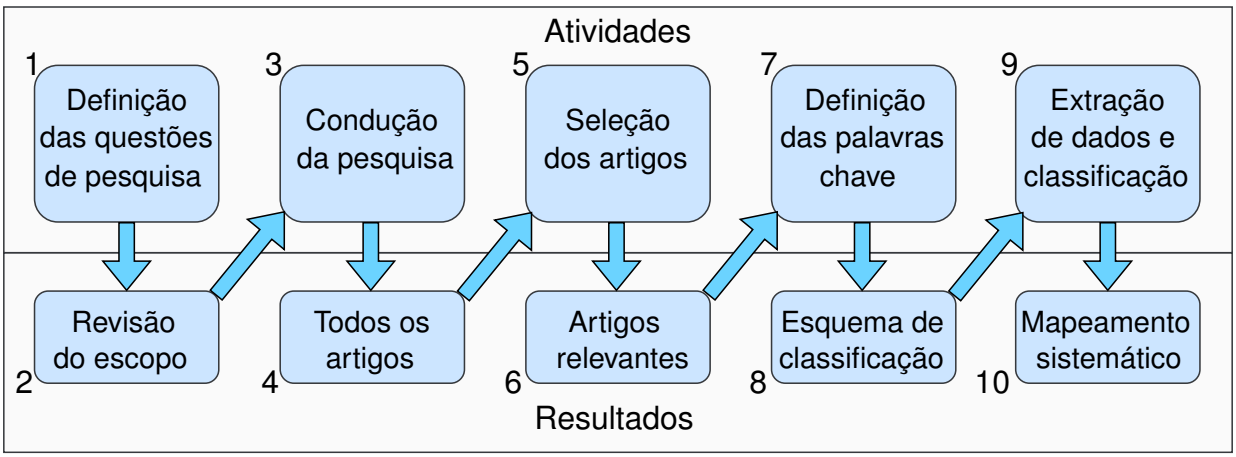

Figura 3.1 – Processo de SMS

o objetivo deste trabalho é abranger esses estudos iniciais, a definição da população não deve considerar essa restrição. A intervenção é a variável que direciona para uma questão específica, por exemplo, principais ataques em SDN. A comparação define sob qual ótica a intervenção deve ser comparada. Por exemplo, ataques em SDN realizados em ambiente real *versus* ataques realizados em ambientes simulados. A comparação ocorre normalmente em SMS sobre assuntos já maduros, onde o refinamento se faz necessário para aprofundar o tema. Por fim, os efeitos devem relatar os fatores que importam no resultado da pesquisa. Por exemplo, "prover segurança sem degradar o desempenho da rede".

Este passo resulta em todos os artigos, oriundos das bases de dados, que estão de acordo com os critérios das *strings* de busca (4). Como os critérios de seleção do estudo visam identificar os estudos preliminares que proporcionam evidências diretas sobre a pergunta de pesquisa, os critérios devem ser decididos durante a definição do protocolo, embora eles podem ser refinados durante o processo de busca. A definição da *string* de busca e das bases de dados é importante porque o processo do SMS deve visar ser transparente e replicável. Sendo assim, o SMS deve ser documentado em detalhes suficientes que os leitores possam avaliar o rigor da pesquisa. Posteriormente, é necessário filtrar os artigos através de critérios de inclusão e exclusão, com o objetivo de excluir artigos irrelevantes para responder as questões de pesquisa (5). Para chegar a esse resultado, os critérios de inclusão e exclusão dos artigos devem ser baseados na questão de pesquisa. Esses critérios precisam ser elaborados de forma a garantir que sejam interpretados, durante a sua aplicação, de forma correta, que possam assim auxiliar na classificação dos estudos corretamente. Este passo resulta somente nos artigos relevantes (6). É importante armazenar os artigos não relevantes para uma futura reanálise, caso necessário, assim como facilitar o processo de filtro, no caso de uma atualização ou replicação do SMS.

Após, deve-se classificar os artigos. Para prover essa classificação, os resumos são lidos em busca de extrair palavras-chave e conceitos relevantes, que permitem identificar o contexto da pesquisa (7). A extração dos dados dos artigos deve ser projetada com

o intuito de coletar todas as informações necessárias para o direcionamento das questões de pesquisa. O ideal é que a extração de dados seja realizada, sempre que possível, por dois ou mais pesquisadores, de forma independente. Esses dados deverão ser comparados e as não conformidades resolvidas, através de consenso entre os pesquisadores ou de forma arbitrária por um pesquisador independente adicional. Para pesquisadores individuais, como um estudante de doutorado, podem ser utilizadas outras técnicas de verificação da qualidade do trabalho em progresso. Por exemplo, o orientador pode extrair uma amostra de alguns dados, de forma aleatória, e os seus resultados serem confrontados com os resultados do estudante. Outra possibilidade é um processo de teste-reteste, onde o pesquisador realiza uma segunda extração dos dados a partir de uma seleção aleatória dos trabalhos, de forma a verificar a consistência da extração realizada inicialmente. Quando este estágio está completo, o conjunto de diferentes artigos são combinados para alcançar um alto nível de entendimento sobre a natureza e contribuição da pesquisa. Esta combinação ajuda o desenvolvimento de um conjunto de categorias a respeito da população em questão. Quando os resumos não são suficientes, os revisores filtram as seções de introdução e conclusão dos artigos. Isso resulta em um conjunto de categorias para compor o mapeamento (8).

Finalmente, os artigos são agrupados em categorias estabelecidas por suas palavras-chave, permitindo a análise e cálculo das frequências de publicações em cada categoria (9). Os resultados permitem descobrir quais categorias tem sido enfatizadas nas pesquisas e identificar lacunas e possibilidades para estudos futuros (10) [448].

3.1 Questões de pesquisa

O SMS foi organizado, inicialmente, definindo as questões de pesquisa (passo 1 da Figura 3.1). A pergunta foi: "Quais os principais aspectos relacionados com segurança em SDN?". Para isso, precisa-se considerar se esta área de pesquisa é recente. Sendo assim, o número total de artigos publicados nos últimos anos pode ser um dado interessante, porque essa informação pode ajudar a entender se, ano a ano, os pesquisadores estão investindo mais esforços para resolver questões relacionadas ao tema, e se ainda é interessante investir nessa área. Naturalmente, a primeira questão de pesquisa foi:

QP1: Quantos artigos envolvendo segurança em SDN foram publicados entre 2008 e 2018?

Em seguida, foi necessário pensar sobre as oportunidades que a segurança em SDN produz. Sendo assim, foram definidas categorias básicas relacionadas com o tema: vulnerabilidades, ataques e mecanismos de defesa. Todas as situações podem ser agrupadas nessas categorias, permitindo a definição das seguintes questões de pesquisa (passo 2 da Figura 3.1):

QP2: Quais são as principais vulnerabilidades em SDN?

QP3: Quais são os principais ataques em SDN?

QP4: Quais são as principais abordagens para prover segurança em SDN?

3.2 Estrutura das Questões de Pesquisa

O próximo passo foi definir as *strings* de busca (passo 3 da Figura 3.1). Usando a metodologia proposta por [292], os quatro itens para a estrutura são:

- População: artigos publicados em SDN.
- Intervenção: segurança.
- Comparação: não aplicável porque o objetivo não é focar em algum assunto específico.
- Efeito: os resultados esperados foram publicados entre 2008 e 2018. Os mecanismos de filtro, oferecidos pelos motores de busca das bases de dados, somente proporcionam a definição da busca para os intervalos em anos, não permitindo a definição de critérios mais específicos, como meses e dias. Essa limitação restringe a consulta, interferindo nos resultados retornados caso se busque o ano corrente. Como novos estudos tem sido publicados continuamente, se a String de busca fosse aplicada hoje, os resultados poderiam ser diferentes da mesma execução no dia seguinte, dificultando a replicação do estudo para validação.

Baseado nessa estrutura, foi definida uma tabela de sinônimos para a identificação de palavras-chave para a construção da String de busca, como ilustrado na Tabela 3.1. Esses sinônimos são contextuais, criando um relacionamento semântico entre os termos.

Tabela 3.1 – Estrutura de sinônimos

Estrutura	Termos	Sinônimos
População	Software Defined Network	SDN Software-defined Networking Software-defined Network Openflow
Intervenção	Security	Secure Attack Vulnerability Authentication Integrity Confidentiality Availability

3.3 Processo de busca

Antes da criação da string de busca, é importante definir onde ela será aplicável. Para realizar a busca, somente foram utilizadas bases de dados que possuem um motor de busca baseado na *Web*, possuem um mecanismo de busca habilitado para busca por palavras-chave e contenha artigos na área de computação. Essa seleção incluiu ACM Digital Library, IEEE Xplore e SCOPUS devido ao fato dessas bases serem extremamente representativas em termos de publicações em computação. Outras bases de dados não foram utilizadas em virtude, ou da complexidade (ou impossibilidade) de instanciar a *string* de busca, ou por apresentar um alto índice de artigos repetidos com as bases IEEE e ACM, ou por realizarem apenas uma indexação para as bases (como Google Scholar, por exemplo).

3.4 *String* de busca

Para a definição da *string* de busca, foram utilizadas as operações booleanas "OR"para selecionar palavras opcionais e sinônimos e "AND"para selecionar os termos relacionados a população, intervenção e efeito, resultando na *string* de busca ((“software defined network” OR “software-defined networking” OR “SDN” OR “openflow”) AND (“security” OR “secure” OR “attack” OR “vulnerability” OR “authentication” OR “integrity” OR “confidentiality” OR “availability”)).

3.5 Critérios de inclusão e exclusão

Com o objetivo de filtrar os artigos aceitáveis, foi definido um conjunto de três critérios de inclusão e três critérios de exclusão, os quais são listados abaixo:

- CI1: O estudo em fase inicial deve propor uma solução para segurança em SDN;
- CI2: O estudo em fase inicial deve descrever ataques ou vulnerabilidades em SDN;
- CI3: O estudo em fase inicial deve discutir questões de segurança em SDN;
- EC1: O estudo deve estar escrito em uma língua diferente do inglês;
- EC2: O estudo não foi publicado entre 2008 e 2018;
- EC3: O estudo não é relacionado a segurança e SDN (Software-defined Networking). Por exemplo, *Student Doctor Network* ou *Science and Democracy Network*

3.6 Resultados

Cada base de dados oferece uma estrutura específica para executar a *string* de busca. Algumas bases oferecem caixas de seleção para selecionar as opções de filtro da busca; algumas oferecem um campo de texto para escrever as opções, como o escopo anual; algumas oferecem peculiaridades gramaticais para a escrita da *string* de busca. Esses requisitos demandam adaptações na *string* de busca para cada base de dados. Após as adaptações para cada base, a execução das *strings* resultaram em 2899 artigos (passo 4 da Figura 3.1), como identificado na Tabela 3.2. Então, foram removidos os artigos duplicados e filtrados através dos critérios de inclusão e exclusão (passo 5 da Figura 3.1). Como resultado, a amostra foi reduzida para 571 artigos (passo 6 da Figura 3.1).

Tabela 3.2 – Número de artigos retornados e selecionados em cada base de dados

Bases	Artigos Retornados	Artigos Selecionados
ACM Library	331	95
IEEExplore	986	442
Scopus	1582	34
TOTAL	2899	571

Para responder as questões, foram avaliadas as palavras-chave do artigo, além de terem sido extraídas mais palavras-chave dos títulos e resumos dos artigos. Quando o título e resumo não proveram as palavras-chave relacionadas, as palavras-chave foram selecionadas através da introdução e conclusão. Se as etapas prévias não foram suficientes, as palavras-chave foram extraídas do artigo como um todo (passo 7 da Figura 3.1).

As palavras-chave foram então categorizadas baseadas nas questões de pesquisa interessadas nas vulnerabilidades, ataques e mecanismos de segurança (passo 8 da Figura 3.1). Em seguida, os artigos foram classificados em tabelas (passo 9 da Figura 3.1), resultando assim no mapeamento sistemático (passo 10 da Figura 3.1).

3.6.1 QP1: Quantos artigos envolvendo segurança em SDN foram publicados entre 2008 e 2018?

A questão de pesquisa QP1 pode ser mensurada como ilustrado na Tabela 3.3. Os resultados indicaram que as publicações que esta metodologia identificou, relacionadas à segurança em SDN, começaram a surgir a partir de 2010. Isso pode ser compreendido pelo fato das pesquisas em SDN, embora não tenham começado com o surgimento do Openflow, terem começado a ganhar força com o início do Openflow [371]. Ou seja, até meados de 2010, o foco da comunidade era tornar SDN algo funcional, deixando de lado os requisitos não funcionais, como segurança.

Com o aumento do interesse da academia e da indústria por SDN, as preocupações com segurança começaram a surtir, e as publicações foram aumentando lentamente até 2013, período onde as publicações começaram a ter um número expressivo. Esse salto quantitativo indica que segurança em SDN se tornou um tópico que começou a atrair a atenção da comunidade científica de forma intensa. A taxa mais alta de publicação foi detectada em 2017, demonstrando que a segurança em SDN é uma excelente área para realizar pesquisas, considerando que vários tópicos estão em estágio inicial, são necessários aprofundamentos e aperfeiçoamento para desenvolver, de forma madura, os temas de pesquisa. A alta taxa de artigos publicados em 2017 indica que o tema envolvendo segurança em SDN ainda está numa crescente em termos de pesquisas, ou seja, ainda não chegou na fase de amadurecimento. Isso indica que este tema ainda necessita de esforços significativos para que os problemas sejam resolvidos e as tecnologias se tornem maduras e robustas.

Como o mapeamento foi realizado durante o final do primeiro semestre de 2018, o baixo quantitativo de artigos publicados em 2018 não reflete uma tendência de queda, em virtude de ocorrer um intervalo de tempo, muitas vezes longo, entre a realização de eventos científicos e a efetiva publicação dos artigos desses eventos nas bases de dados. Sendo assim, para considerar o quantitativo de artigos de 2018, é necessário repetir o mapeamento sistemático no início de 2019.

Tabela 3.3 – Artigos publicados por ano

Ano	ACM	IEEE	SCOPUS	TOTAL
2008	0	0	0	0
2009	0	0	0	0
2010	1	1	0	2
2011	0	3	0	3
2012	5	5	0	10
2013	10	20	7	37
2014	12	39	13	64
2015	9	68	14	91
2016	27	123	0	150
2017	24	162	0	186
2018	7	21	0	28

3.6.2 QP2: Quais são as principais vulnerabilidades em SDN?

Durante o processo de mapeamento sistemático, foram categorizadas as principais vulnerabilidades existentes em SDN. As vulnerabilidades encontradas são apresentadas na Tabela 3.4. As vulnerabilidades relacionadas à DoS foram expressivamente mais investigadas. Outras vulnerabilidades identificadas foram Controle de Acesso, Autentica-

ção, Violação de Confidencialidade, *Misconfiguration*, Violação de Integridade, Falha no Controlador, Atualização na rede e Falta de TLS.

Tabela 3.4 – Artigos categorizados por vulnerabilidades

Contribuição	Artigos
DoS	[525] [24] [652] [329] [377] [444] [342] [177] [521] [524] [304] [500] [590] [323] [151] [400] [354] [588] [53] [223] [419] [610] [312] [118] [297] [219] [296] [282] [655] [589] [557] [295] [38] [143] [302] [644] [370] [69] [303] [350] [132] [252] [572] [602] [601] [107] [570] [351] [266] [640] [392] [387] [592] [262] [453] [616] [256] [533] [353] [606] [108] [139] [626] [147] [450] [658] [195] [221] [344] [623] [414] [352] [68] [485] [633] [73] [309] [66] [305] [89] [25] [298] [56] [431] [451] [286] [239] [635] [355] [490] [641] [595] [145] [123] [335] [55] [133] [109] [236] [347] [135] [673] [628] [122] [14] [600] [630] [260] [97] [155] [404] [475] [376] [16] [422] [161] [248] [115] [639] [230] [339] [515] [34] [210] [15] [213] [518] [192] [648] [382] [581] [643] [241] [261] [603] [540] [160] [472] [604] [398] [237] [81] [413] [456] [114] [99] [23] [184] [253] [397] [594] [484] [137] [675] [670] [226] [6] [183] [399] [656] [102]
Controle de Acesso	[302] [293] [529] [50] [246] [294] [678]
Autenticação	[400] [529] [591] [566] [267] [317] [596] [674] [144] [32]
Violação de Confidencialidade	[296] [557] [295] [405] [8] [52] [124]
Misconfiguration	[394] [369] [21] [140] [361] [321] [363] [208] [212] [20] [288] [179] [124] [206]
Violação de Integridade	[557] [295] [565] [325] [384]
Falha no Controlador	[512] [586] [667] [83]
Atualização na Rede	[346] [455] [575] [501] [380] [531] [340] [379]
Falta de TLS	[377] [500] [557] [295] [362] [190] [234] [28] [168] [311] [17] [13] [37] [302] [71] [506] [553] [73] [58] [4] [275] [641] [163] [133] [12] [415] [404] [605] [468] [199] [646] [437] [33] [391] [11] [271] [10] [460] [473] [79] [283] [461] [285] [18] [124] [657]

O DoS está dentro da categoria de vulnerabilidades porque, embora seja um ataque, os estudos selecionados focaram nas vulnerabilidades que são exploradas pelo ataque. Nela foram agrupados os trabalhos envolvendo os ataques de DoS e DDoS. Os trabalhos selecionados focam em explorar a dificuldade de manter o Controlador ou um *switch* funcionando com o exaurimento de recursos (uso de CPU e memória), a facilidade de aumentar a latência da rede com a inundação de pacotes e a dificuldade de identificar os dispositivos reais que existem na rede, aumentando o tamanho da rede com falsos dispositivos, de forma a tornar inviabilizar o gerenciamento da rede.

As vulnerabilidades relacionadas à categoria Controle de Acesso exploram a possibilidade de acessar os componentes SDN baseado nos problemas relacionados às políticas e delegações de permissões de acesso. A categoria Autenticação apresenta vulnerabilidades de prospectar o problema da impersonalização, ou seja, da dificuldade de identificar se uma entidade na rede é ela mesma (seja um dispositivo Controlador, *switch*, ou mesmo administrador da rede).

A categoria Violação de Confidencialidade organiza os estudos nos quais o foco é acessar informações sensíveis. As tabelas de fluxos e as regras de fluxos são exemplos de informações que podem ser sequestradas devido às vulnerabilidades desta categoria. Já a categoria relacionada a problemas de configuração (*Misconfiguration*) refere-se à vulnerabilidade causada por administradores de rede que, ou intencionalmente ou inconscientemente, injetam na rede erros de configuração. A categoria Violação de Integridade lida com a possibilidade de inserir informações falsificadas na rede SDN, onde os estudos nessa categoria focam em modificar os dados originais, por exemplo, dados de pacotes e regras de fluxos. Na categoria Falha do Controlador, são relacionados os problemas causados pelo mau funcionamento do Controlador. Este grupo de vulnerabilidades estuda as causas e consequências desse mau funcionamento. A categoria Atualização da Rede foca no lapso de tempo entre as alterações na configuração de rede, como regras de fluxos, *firmwares*, etc. Durante esse período, se um dispositivo depender da configuração do dispositivo sendo atualizado, conflitos podem ocorrer, e consequentemente, abrir brechas de segurança.

Por fim, os estudos da categoria Falta de TLS identificam os impactos da falta de uso de TLS para proteger a comunicação entre os dispositivos da SDN. Embora o protocolo TLS seja uma possível solução, são levantadas preocupações sobre como o TLS fora implementado pelos fabricantes dos dispositivos de rede, devido, por exemplo, a especificação do protocolo Openflow ser frágil nesse quesito, pois apenas sugere o uso de TLS na comunicação entre dispositivos, sem prover maiores detalhes sobre como deve ocorrer esse uso.

3.6.3 QP3: Quais são os principais ataques em SDN?

Os ataques relacionados diretamente com SDN são raros, devido à fase inicial das pesquisas relacionadas à segurança em SDN. Entretanto, alguns ataques foram encontrados na literatura, sendo representados na Tabela 3.5. Os ataques identificados foram *Control Plane Saturation*, que ocorre quando um atacante injeta continuamente requisições de fluxos, com o objetivo de inundar a comunicação entre o Controlador e o *switch*, *Interception*, que ocorre quando o atacante captura o tráfego do plano de controle, de forma a desviar o tráfego, inspecionar os dados ou modificá-los.

Tabela 3.5 – Artigos categorizados por ataques

Contribuição	Artigos
Control Plane Saturation	[524] [132] [601] [640] [592] [40] [41] [528] [42] [398] [484] [6]
Interception	[38] [180] [414] [352] [440] [579] [386] [18] [657] [384]
Flow Table Overflow	[570] [330] [266] [533] [353] [466] [36] [628] [630] [662] [629] [331] [409] [631] [461] [6] [184]
Poisoning	[521] [87] [393] [351] [392] [366] [108] [485] [126] [486] [26] [373] [417] [407] [408] [274] [99] [23] [184] [594] [137] [183] [656]
Fingerprinting	[139] [58] [130] [9] [58] [653] [428] [544] [545] [663] [332] [183]

Além desses ataques, foram identificados os ataques de *Poisoning*, onde um atacante injeta pacotes na rede, de forma a inundar o plano de dados, resultando em um DoS; o ataque *Flow Table Overflow*, onde essa categoria de ataques visa produzir novos fluxos, em uma quantidade tal que exceda a capacidade das tabelas de fluxos, e com isso causar um DoS nos *switches*. Isso pode ocorrer via Controlador ou através de um ataque *Men-in-the-Middle* (MitM) [657]. Por fim, o ataque de *Fingerprinting*, onde este tipo de ataque visa obter informações sobre o Controlador, rotas ou dispositivos chaves na rede, tais como servidores, por exemplo, a partir de variações em métricas da rede, como através da diferença de tempo ao trafegar pacotes pela rede.

3.6.4 QP4: Quais são as principais abordagens para prover segurança em SDN?

A categorização dos esforços retornados pelo SMS para solucionar os desafios de segurança em SDN, ou que utilizam as vantagens que SDN oferece para solucionar os tradicionais problemas de segurança, são exibidos nas Tabelas 3.6 e 3.7. A categoria de Detecção de Anomalias apresenta a maior parte das pesquisas, que envolvem a investigação de algoritmos para indicar onde ou quando a rede, ou algum dispositivo de rede, está sob ataque. Já a categoria *Middleboxes* apresenta trabalhos com o objetivo de desenvolver ferramentas para oferecer *Middleboxes* eficientes, por exemplo, *Firewall*, IPS e IDS. A categoria Autenticação agrupa abordagens relacionadas a autenticação de dispositivos para evitar a impersonalização.

Na categoria Arquitetura ou *Framework* de Segurança se concentram abordagens que visam ver a rede amplamente, fornecendo um conjunto de recursos para cobrir diferentes requisitos de segurança no mesmo sistema. A categoria Mitigação de Anomalias provê técnicas para suspender ações que surgem quando algo ocorre de forma inesperada na rede, causadas por ataques ou falhas. Já os trabalhos relacionados na categoria Gerenciamento Autônomo propõem agregar um grau de inteligência para o processo de prevenção, identificação e recuperação de ataques. A categoria Criptografia possui tra-

Tabela 3.6 – Artigos categorizados por mecanismos de segurança

Contribuição	Artigos
Detecção de Anomalias	[198] [105] [176] [525] [329] [177] [590] [323] [588] [53] [610] [655] [302] [303] [148] [362] [196] [233] [257] [487] [194] [85] [255] [374] [251] [625] [624], [536] [207] [454] [27] [186] [602] [601] [235] [107] [187] [392] [387] [264] [104] [108] [139] [509] [510] [626] [147] [416] [9] [580] [344] [623] [599] [352] [633] [555] [73] [309] [66] [388] [523] [238] [25] [103] [502] [239] [217] [123] [162] [385] [109] [673] [318] [54] [97] [541] [386] [434] [16] [439] [378] [7] [648] [110] [581] [643] [81] [80] [413] [514] [511] [253] [657] [484] [325] [675] [670] [399] [102]
Middleboxes	[525] [297] [302] [370] [148] [255] [625] [624] [536] [207] [27] [121] [84] [86] [659] [447] [257] [165] [166] [508] [462] [593] [530] [70] [49] [498] [113] [495] [389] [88] [569] [211] [276] [277] [60] [286] [158] [273] [145] [668] [467] [552] [359] [308] [427] [429] [474] [337] [585] [156] [631] [218]
Autenticação	[557] [303] [591] [207] [457] [278] [650] [567] [638] [368] [291] [170], [152] [566] [267] [317] [596] [420] [153] [4] [486] [263] [144] [289] [129] [573] [2] [184] [94] [397] [32]
Arquitetura ou *Framework* de Segurança	[24] [143] [457] [520] [232] [290] [1] [548] [313] [491] [607] [234] [350] [252] [268] [483] [393] [330] [321] [672] [180] [74] [112] [326] [47] [446] [39] [46] [606] [44] [364] [666] [328] [494] [441] [188] [613] [284] [126] [91] [645] [314] [401] [240] [242] [197] [95] [316] [224] [181] [192] [632] [30] [482] [463] [662] [5] [547], [604] [636] [243] [513] [99] [23] [6] [183] [656]
Mitigação de Anomalias	[198] [329] [590] [323] [588] [53] [610] [69] [625] [659] [343] [287] [132] [572] [476] [640] [496] [315] [592] [262] [453] [616] [100] [256] [366] [533] [349] [450] [664] [195] [412] [131] [68] [485] [615] [305] [89] [436] [348] [128] [431] [365] [451] [635] [26] [355] [490] [90] [106] [155] [417] [404] [582] [475] [422] [77] [161] [230] [339] [515] [125] [210] [271] [15] [629] [518] [409] [241] [261] [603] [540] [472] [381] [237] [214] [461] [594] [137] [226]
Gerenciamento Autônomo	[444] [354] [303] [117] [556] [45] [538] [327] [383] [298] [36] [55] [345] [479] [376] [516] [164] [3] [395] [654]
Criptografia	[400] [303] [529] [190] [201] [146] [320] [101] [311] [324] [471] [553] [627] [336] [634] [481] [163] [203] [519] [402] [403]
Controle de Acesso	[207] [368] [190] [550] [492] [505] [617] [63] [50] [246] [294] [43] [561] [430] [618] [445] [51] [64] [622] [574] [678]
Gerenciamento de Recursos	[589] [190] [300] [576] [111] [269] [568] [200] [363] [254] [221] [465] [621] [169] [612] [528] [205] [67] [248] [213] [611] [449] [96] [671] [310] [160]

balhos que focam em algoritmos e protocolos criptográficos para proteger a comunicação entre dispositivos na SDN.

A categoria Controle de Acesso agrupa estudos que objetivam prover técnicas para controlar as permissões de acesso a componentes da SDN, tanto dentro quanto fora

Tabela 3.7 – Artigos categorizados por mecanismos de segurança (Continuação)

Contribuição	Artigos
Verificação de Modelos	[644] [21] [538] [250] [542] [140] [361] [35] [503] [179]
Caminho Seguro	[75] [281] [341] [583] [597] [76] [493] [598] [124] [384]
Moving Target Defense	[151] [245] [265] [351] [356] [658] [559] [464] [93] [537] [56] [661] [653] [116] [660] [335] [669] [122] [115] [677] [609] [360] [225] [57] [608] [114]
Gerenciamento de Pacotes	[320] [526] [527] [61] [676] [338]
Gerenciamento de Políticas	[346] [232] [182] [333] [432] [340] [379] [433] [571] [620] [637] [270] [272] [534] [98] [455] [127] [575] [501] [380] [531] [438] [208] [288] [206]
Controlador Seguro	[522] [504] [92] [48] [619] [319] [647] [587] [22] [558] [665]

da rede. Na categoria Gerenciamento de Recursos são organizados os trabalhos que estão preocupados com o uso racional dos recursos da rede, por exemplo, memória, processador e largura de banda, normalmente frente a ataques DoS. Os estudos da categoria Verificação de Modelos recomendam a verificação formal para incrementar a segurança, normalmente usando linguagens de abstração específicas de domínio.

Os trabalhos que sugerem aumentar a segurança através da disponibilização de rotas seguras dentro da SDN, para o tráfego de informações sensíveis, são agrupados na categoria Caminho Seguro. Já os estudos que são organizados na categoria *Moving Target Defense* propõem proteger a SDN com técnicas de alteração do endereço da vítima de forma continua, aumentando assim a dificuldade para o atacante descobrir o endereço correto para atacar. Na categoria Gerência de Pacotes, a ideia é romper com o princípio fundamental da SDN, ou seja, parar de pensar somente em fluxos e também manipular os pacotes, de forma a proteger a rede contra ataques.

A categoria Gerência de Políticas organiza trabalhos que visam investigar como executar atualizações na SDN, sem introduzir ambiguidades durante o processo. Por fim, na categoria Controlador Seguro estão os trabalhos que focam no desenvolvimento de um Controlador que lida, essencialmente, com os requisitos de segurança necessários para oferecer garantias de segurança para a rede.

3.7 Discussão

Como as vulnerabilidades em SDN ainda estão em fase de descoberta e solução, vários trabalhos que respondem a QP2 (principais vulnerabilidades) não possuem ainda um mecanismo de defesa, assim como vários trabalhos que respondem a QP4 (principais

mecanismos de defesa), exploram os benefícios da SDN para resolver problemas de segurança que existem nas redes tradicionais, não possuindo assim uma correlação direta com as vulnerabilidades categorizadas neste trabalho.

Por exemplo, o trabalho de Chang e Lin [94], da categoria *Middleboxes*, analisa a situação que, quando uma rede utiliza um Firewall, este serviço é executado normalmente na borda da rede, de forma que o serviço não protege contra atacantes que se encontram dentro da rede (*Insiders*). Como forma de solução, é proposto um firewall distribuído, que utiliza um algoritmo para a definição de regras de firewall, a partir de regras de fluxos, além de um outro algoritmo para a definição do local de armazenamento dessas regras de fluxos, a partir da localização dos *switches* na rede e a quantidade de memória disponível nos dispositivos para armazenar as tabelas de fluxos, de forma a interceptar o tráfego interno e proteger contra ataques na rede realizados por *insiders*. Essa proposta não protege contra alguma vulnerabilidade gerada pelo uso de SDN, e sim explora o conceito de SDN para propor uma solução para um problema de segurança clássico. Sendo assim, este trabalho somente se encontra nos mecanismos de defesa, na categoria *Middleboxes*.

Outro exemplo é o caso do trabalho de Vizarreta e outros [586], organizado na categoria Falha no Controlador, onde o objetivo desse artigo foi analisar, modelar e avaliar o impacto que as diferentes falha do Controlador têm na sua disponibilidade. Foi proposto um modelo de formalismo de Redes de Atividades Estocásticas (SANs - *Stochastic Activity Networks*) e aplicado a um estudo de caso de um Controlador hipotético, baseado em implementações de Controladores comerciais. No caso do estudo, foi mostrado como o modelo proposto pode ser usado para estimar a disponibilidade do Controlador, quantificar o impacto de diferentes modos de falha no Controlador, bem como os efeitos de falhas que são acumuladas durante o tempo, como fluxos já excluídos das tabelas dos *switches*, mas que ainda existem nos registros do Controlador, ou fluxos que atingiram o *timeout* e não foram excluídos. Este trabalho não propõe nenhuma solução para um problema originado pelo uso de SDN ou soluciona um problema através de SDN, e sim somente descreve e estima as vulnerabilidades, por isso se encontra somente em Vulnerabilidades, na categoria Falha no Controlador.

Porém, existem trabalhos que exploram uma vulnerabilidade bem definida, através de um ataque próprio para SDN e ainda apresentam uma proposta de solução, como, no caso de explorar vulnerabilidades relacionadas a DOS, o trabalho de Gao e outros [184] propõe uma solução, categorizada em *Middleboxes*, para proteger a rede contra ataques das categorias *Poisoning* e *Flow Table Overflow*, onde a solução proposta é composta por seix módulos funcionais: monitor de tráfego, monitor de status de *host*, controlador, abstração de aplicativo de controle, detecção de ataque e servidor de auditoria. O módulo de monitor de tráfego funciona como o plano de dados e é executado em um hardware de rede. Ele processa e monitora tanto e tráfego de entrada quanto de saída, com base nas regras de fluxo de sua tabela de fluxos. O módulo de monitor de status de *host* é uma apli-

cação, no *host*, que monitora as informações do *host*. Essa aplicação fornece informações do *host* para o módulo de abstração de controle, de forma a ativar o gerenciamento em nível de aplicativo e realizar assim uma detecção de ataque de forma precisa. O módulo de abstração de aplicativo de controle é uma camada intermediária entre o Controlador e os aplicativos de controle. Esse módulo recolhe informações de *hosts* e de tráfego da rede, e associa cada pacote com as informações do *host*. Além disso, a abstração do aplicativo de controle abstrai a linguagem de implementação do Controlador para uma linguagem de alto nível, fornecendo interfaces amigáveis para atualizar dinamicamente as políticas de segurança de rede. O módulo de detecção de ataque é um aplicativo de controle pré-instalado, que identifica o tráfego malicioso, com base nos *hosts* e no tráfego da rede. O servidor de auditoria é um dispositivo para detectar se o plano de controle está sendo atacado por um ataque de *poisoning*. Para isso, o servidor de auditoria verifica as regras de fluxo no monitor de tráfego e, periodicamente, o servidor coleta as informações de *host* e tráfego e usa o mesmo banco de dados e algoritmos de detecção de ataque do módulo de detecção de ataques para verificar a legalidade das regras de fluxo.

Outro trabalho da Categoria DoS visa proteger a rede dos ataques de *Flow Table Overflow* e *Control Plane Saturation*, através de uma arquitetura de segurança [6], para resistir aos ataques em ambientes de nuvem em SDN. Para isso, o ataque de *Flow Table Overflow* é impedido através da inspeção de pacotes que chegam aos *switches*. O ataque *Control Plane Saturation* é impedido através de uma arquitetura de Controladores múltiplos, que também minimiza os ataques de *Poisoning*. A arquitetura também possui um sistema de autenticação, visando mitigar que um usuário malicioso entre na rede. Essa autenticação ocorre através de uma assinatura digital com o algoritmo de *hash* SHA3. A localização dos Controladores é definida através da composição de um algoritmo genético e de um algoritmo de meta-heurística baseado em população (*Cuckoo Search*). Além disso, é utilizado um protocolo de roteamento baseado no algoritmo PSO (*Particle Swarm Optimization*). O PSO é um algoritmo de otimização baseado em população, que é formado com base no comportamento social de pássaros e peixes à procura de comida. Para a definição de rota, o algoritmo considera o congestionamento dos nós, o congestionamento dos *links* e o atraso no nó. Assim como Chen e outros [99] combate o ataque de *Poisoning* através de uma arquitetura de segurança formada por vários *middleboxes* (*Firewall*, *IPS*, *IDS*) distribuídos pela rede, na forma de virtualização de funções de rede.

Outro trabalho da categoria DoS, visa proteger a rede especificamente contra ataques de *Poisoning* e *Fingerprinting*, através de um *Framework* de segurança [183]. Para isso, o trabalho propõe combater aos ataques através de dois módulos: um agente de tráfego e um agente de visão global. O agente de tráfego atua entre os planos de dados e controle, processando alguns tráfegos, de forma a carga de trabalho do Controlador. Para isso, o agente de tráfego identifica e filtra pacotes oriundos de um ataque de DoS e também injeta um atraso em alguns pacotes (eliminando assim a possibilidade do atacante descobrir

alguma informação baseada em tempo de tráfego). Além disso, o agente verifica a existência dos *hosts*, enviando pacotes de sondagem (combatendo assim ataques de *Poisoning*). Já o agente de visão global é uma aplicação do Controlador que provê informações globais da rede para o agente de tráfego, e também repassa os fluxos válidos, oriundos do agente de tráfego, para outras aplicações do Controlador. Assim como Alasadi e Al-Raweshidy [23] propõe combater o ataque de *Poisoning* através de uma arquitetura de segurança que diminui as mensagens de descoberta de *hosts*, evitando ataques de DoS que exploram mensagens como ARP e DHCP. Para isso, o Controlador e os servidores desses tipos de serviço compartilham a responsabilidade de responder essas mensagens, dependendo do tipo de serviço solicitado, de forma que, para um mesmo serviço, várias portas de entrada sejam direcionadas para uma mesma porta de saída, diminuindo assim a quantidade de mensagens que circulam no plano de controle, e consequentemente os fluxos armazenados nas tabelas de fluxos. Com essa solução, as mensagens de *broadcast* são eliminadas, pois cada serviço tem como destino um servidor pré-determinado.

Alguns trabalhos visam utilizar mitigação de anomalias para combater DoS, como Wang e outros [594], que propõe combater ataques de *Poisoning*, através de um algoritmo que procura links com fluxos de alta densidade, que podem ser usados para a efetivação do ataque. Além disso, como o ataque pode congestionar os links do alvo, de forma a impedir a comunicação na área desse alvo, a abordagem proposta detecta o congestionamento de links, através de pacotes usados para sondar a rede, analisando o tempo dispensado para o pacote chegar ao alvo. Para isso, é proposto um módulo monitor de congestionamento de links, um módulo de seleção de links de alvos, um módulo de redefinição de rotas para o tráfego, e um módulo de bloqueio de tráfego malicioso, para atuarem como aplicações do Controlador. Foi desenvolvido um agente de monitoramento de congestionamento de links, que é implantado dinamicamente em um link alvo, para medir o congestionamento desse link.

No caso de Detecção de anomalias, como DoS, o trabalho de Sahoo, Tiwary e Sahoo [484] propõe um IDS para detectar ataques de DoS em eventos relâmpagos. Um evento relâmpago produz um alto tráfego no plano de controle, durante um intervalo extremamente curto. Essa característica faz com que um evento relâmpago seja difícil de diferenciar de um ataque DoS. Para detectar quando um ataque DoS está realmente ocorrendo durante um evento relâmpago, a abordagem proposta utiliza como métricas a informação baseada no conceito de entropia geral e distância da informação generalizada, de forma a diferenciar um ataque DoS de um evento relâmpago. A entropia geral mede a incerteza de um evento associada a uma dada distribuição de probabilidades, e a distância da informação generalizada compara duas distribuições de probabilidades. Quando o tráfego aumenta, a entropia de um evento relâmpago é quase igual a entropia de um ataque DoS. Por isso, a proposta calcula a entropia geral e a distância de informação generalizada, para os dois eventos (evento relâmpago e DoS), forma a discriminar esses dois eventos. O tra-

balho demonstra que essa abordagem diminui a quantidade falsos positivos para eventos relâmpago.

Já Jevtic, Lotfalizadeh e Kim [253] propõe combater DoS com um IDS bioinspirado, semelhante a função do sistema imunológico humano de detectar patógenos desconhecidos. Para isso, utiliza uma abordagem de monitoramento distribuído com técnicas de geração de padrões maliciosos de um sistema imunológico artificial adaptado. O sistema se baseia no conceito de toxidade, que considera como os servidores recebem e processam as requisições de serviços. A quantidade de toxidade que um fluxo acumula com cada recepção de pacote é relacionado com o tamanho máximo da fila do servidor, o tamanho do pacote, o protocolo utilizado. Com isso, a toxidade aumenta a cada evento. Baseado nisso, os níveis de toxidade são mantidos pelas estatísticas dos pacotes e dos bytes trafegados. O sistema utiliza também uma biblioteca de patógenos, formada por treinamentos realizados em um conjunto de dados de anomalias e pelas anomalias identificadas pelo sistema, de forma a manter um sistema imunológico adaptativo. Esse treinamento também gera um conjunto de anticorpos, que são ações a serem realizadas quando uma anomalia é identificada.

O trabalho de Zhu e outros [675] aborda o ataque de DoS entre domínios de rede diferentes. E para a proteção contra este tipo de ataque, propõe um esquema de detecção de anomalias formado por dois servidores, um servidor responsável pelo processamento dos dados enviados pelos domínios e outro servidor responsável pela detecção do ataque. O esquema utiliza o algoritmo kNN (k Nearest Neighbor) adaptado para classificar os dados e detectar o ataque. A criptografia com perturbação é utilizada para proteger a privacidade dos diferentes domínios de rede. As comunicações entre esses dois servidores e entre os servidores e os Controladores SDN ocorrem via TLS. Cada domínio produz uma 7-upla <SN, T, MPF, MBF, PCF, GOP, GSI> a partir dos dados da tabela de fluxos. SN é um número de série, gerado sequencialmente; T é o *timestamp* do fluxo; MPF é a média de pacotes por fluxo; MBF é a média de bytes por fluxo; PCF é a porcentagem de correlação de fluxos (considerando dois fluxos, onde o endereço IP da origem de um fluxo é o mesmo endereço IP de destino do outro fluxo, e vice-versa, usando o mesmo protocolo, PFC é a soma dos fluxos que se encaixam nessa situação, dividido pelo total de fluxos); GOP descreve a taxa de crescimento do número de portas, dentro de um intervalo fixo; e GSI representa a taxa de crescimento do número de endereços IP de origem, dentro de um intervalo fixo. Além disso, o domínio gera um parâmetro de perturbação e criptografa a 7-upla com esse parâmetro. O resultado da criptografia é enviado para o servidor de processamento, e o parâmetro de perturbação é enviado para o servidor de detecção. Em seguida, o servidor de processamento aplica parte do algoritmo kNN e envia o resultado do processamento, assim como a 7-upla, para o servidor de detecção. O servidor de detecção descriptografa a 7-upla com o parâmetro de pertubação e utiliza o resultado do processamento com a 7-upla

para concluir a execução do algoritmo kNN. Caso o resultado da execução seja a detecção de um ataque, o esquema envia um alarme (formado pelo SN e T) para o domínio.

Outro trabalho visa proteger um servidor contra DoS através de *Moving Target Defense* [114], onde este trabalho formulou um sistema de pontuação baseado em vulnerabilidades e alertas de IDS, para acionar uma contramedida através da técnica de alterar continuamente o número da porta de um determinado serviço, dificultando assim que o atacante consiga definir onde atacar.

Já o trabalho de Nagai e outros [397] visa combater o DoS com autenticação, de forma a proteger um servidor contra um ataque de inundação de mensagens TCP SYN, através de um autenticador de pacotes TCP baseado em Openflow. Quando um *switch* recebe um pacote TCP, com uma mensagem SYN, o *switch* encaminha esse pacote para o Controlador. Ao receber esse pacote, o Controlador gera um pacote SYN+ACK inválido, e encaminha ao *switch*, além de inserir um fluxo em uma tabela, existente no autenticador, chamada CHECKING_TCP. O *switch* envia esse pacote ao cliente, que faz com que o cliente envie um pacote com uma mensagem RST. O *switch* recebe essa mensagem e encaminha ao Controlador. O controlador, por sua vez, verifica se os dados do pacote correspondem com algum fluxo da tabela CHECKING_TCP. Caso corresponda, o autenticador registra o fluxo em uma tabela CHECKED_TCP, e registra o fluxo no *switch*, liberando o acesso ao servidor. Caso o cliente envie outra mensagem SYN, o Controlador verifica que já existe um fluxo, na tabela CHECKING_TCP, que corresponde aos dados do pacote, descartando assim o pacote e evitando um ataque de DoS.

Na vulnerabilidade envolvendo a facilidade de adulteração, o trabalho de Mohan, Truong-Huu e Gurusamy [384] visa proteger contra o ataque da categoria *Interception* através de uma solução da categoria Caminho Seguro. Quando um *switch* está comprometido, em uma rede SDN de tráfego de controle in-band, esse *switch* pode excluir ou modificar mensagens de controle na rede, especialmente mensagens PACKET_IN, dificultando a detecção da origem do ataque. Para evitar isso, é proposto um algoritmo de definição de rota do plano de controle, onde cada *switch* envia as mensagens do plano de controle por dois caminhos disjuntos, através da definição de um conjunto de *switches* de cobertura. A medida que as mensagens que passam pelos *switches* intermediários e pelos *switches* de cobertura são analisadas, o número de *switches* suspeitos diminui, de forma a convergir para o *switch* comprometido. Para isso, é definido um conjunto de *switches* de cobertura primária, como o conjunto mínimo de *switches* tal que a intersecção dos seus planos de controle resultem em um único *switch* intermediário. Dessa forma, um *switch* malicioso irá ser detectado diretamente, baseado na inconsistência das mensagens PACKET_IN que chegarão nos *switches* do seu conjunto de cobertura primária. Para isso, foi proposta uma formulação de otimização que minimiza o total de nós intermediários nos caminhos do plano de controle, de forma a cada *switch* ser o nó intermediário de pelo menos dois caminhos, oriundos de dois *switches* diferentes, para o Controlador.

Outra solução para a vulnerabilidade organizada na categoria de Facilidade de Adulteração é proposta por Li e outros [325]. Este trabalho aborda a capacidade de um atacante possui de alterar os pacotes na rede, e como mecanismo de proteção propõe um mecanismo dinâmico de verificação de encaminhamento de pacotes, através da amostra de pacotes e coleta de estatísticas de fluxos, em intervalos irregulares. Esses intervalos são ajustados dinamicamente, baseado nos resultados das verificações anteriores. O mecanismo detecta casos suspeitos através da identificação de padrões de ataque, como se um pacote não pode ser verificado com sucesso ou quando as estatísticas de fluxos são inconsistentes com o comportamento esperado do encaminhamento de pacotes. Quando isso ocorre, o mecanismo aumenta a taxa de coleta de amostras de pacotes e de estatísticas de fluxos, de forma a confirmar um ataque mais rapidamente. Um exemplo de técnica de detecção do mecanismo ocorre quando um pacote ingressa na rede. O respectivo *switch* encaminha esse pacote para o Controlador, através de uma mensagem PACKET_IN. O mecanismo gera um MAC (*Message Authentication Code*) para todo o pacote, e armazena o MAC em uma tabela de *hash*, juntamente com um TTL (*Time To Live*). Se a mensagem PACKET_IN vier de um *switch* conectado ao destino, e o MAC do pacote estiver armazenado na tabela de *hash* com um TTL não expirado, o registro é removido da tabela. Isso garante que o pacote não sofreu um ataque que atrase o tráfego de pacotes. Outro exemplo de detecção ocorre caso um pacote venha percorrendo a rede, e consequentemente o MAC desse pacote venha sendo comparado na tabela de *hash* após chegar em cada *switch* da rota, e em um determinado ponto o MAC não corresponda a nenhum registro na tabela *hash*, o mecanismo conclui que o pacote foi adulterado durante o tráfego. Outro exemplo de detecção ocorre quando as estatísticas de um fluxo não apresente os mesmos valores em todos os *switches* de uma determinada rota, e com isso o mecanismo conclui que o *switch* de apresenta as estatísticas diferentes é um *switch* suspeito.

Na categoria de vulnerabilidades *Misconfiguration*, o trabalho de Morzhov e Nikitinskiy [390] se depara com o problema de gerir políticas de Firewall em uma SDN formada por controladores distribuídos, onde uma nova política pontual pode impactar nas políticas de outros controladores. Para solucionar o problema, é proposto um algoritmo de pré Firewall, que recebe uma política e compara com um conjunto de políticas em uso, onde tenta detectar anomalias na política. São possíveis dois tipos de anomalias: a anomalia de sombreamento, onde um campo da política corresponde ao campo de uma política que está no conjunto de políticas, de forma plena, inclusiva ou parcial, e a prioridade das políticas comparadas não são iguais; e a anomalia de correlação, que ocorre da mesma maneira que a anomalia de sombreamento, porém as prioridades das políticas são iguais. No caso de sombreamento, dependendo das ações das políticas (permitir ou negar) e das prioridades, a nova regra pode ser adicionada, parcialmente adicionada, ou eliminada. No caso de correlação, a nova política pode ser adicionada, enquanto a política antiga é eliminada ou

dividida, através de uma função de divisão de políticas, que retorna novas políticas com as partes disjuntas da comparação de correlação.

Já Freire e outros [179] resolvem o problema de *Misconfiguration* com uma solução de verificação de modelos. Este trabalho verifica se as configurações definidas na linguagem P4 atendem a um conjunto de propriedades. A solução proposta provê uma linguagem de assertivas que permite aos programadores especificar suas propriedades, realizando anotações nos programas escritos em P4. Essa linguagem proposta permite a especificação de invariantes, e uma vez que essas invariantes tenham sido anotadas, o programa é simbolicamente executado, sendo verificadas as assertivas enquanto todos os caminhos são percorridos. Para isso, o desenvolvedor primeiro anota as assertivas no código P4. Em seguida, o programa P4 é traduzido para um modelo baseado na linguagem C. Durante esse processo, algumas regras de encaminhamento podem ser inseridas no tradutor, de forma a restringir a verificação para uma dada configuração de rede. O modelo gerado é então verificado por um motor simbólico, que testa todos os caminhos de execução, procurando por falhas nas assertivas.

A categoria de *Misconfiguration* é abordada também por Kim e outros [288], que propõe uma solução de gerência de políticas, onde converte políticas de alto nível em políticas de baixo nível, através do protocolo netconf. Para isso, é proposto um gerador de políticas de alto nível, um conversor de informações de políticas, definidas pelo administrador da rede através de interface gráfica, para um formato XML, e um *parser* dessas regras para netconf.

No caso de vulnerabilidades envolvendo o controle de acesso ou a sua falta, O trabalho de Zou e outros [678], descreve o abuso de APIs do northbound em virtude da falta de um sistema de controle de acesso em uma arquitetura multi-tenant. Para isso, é proposto um sistema de controle de acesso, formado por uma API de autorização, responsável por proteger o compartilhamento de APIs do northbound em redes multi-tenant. O sistema é composto por um gerenciador de permissões e um intermediário para acessos em tempo de execução. O gerenciador de permissões abstrai as permissões em uma estrutura de três níveis, além de prover uma linguagem de descrição de permissões, com interfaces para configurações de permissões e verificação de conectividade de redes de usuários autorizados. Já o intermediário para acessos em tempo de execução intercepta todas as chamadas às APIs para garantir a validade e manter as informações da topologia de redes de usuários.

Para resolver problemas envolvendo autenticação, o trabalho de Allouzi e Khan [32] discute a falta de autenticação que existe no protocolo Openflow, de forma que tanto um Controlador quanto um *switch* possam confiar um no outro, toda vez que um dos dispositivos solicite acesso aos recursos sensíveis do outro dispositivo. Para isso, é proposta uma extensão do protocolo de *handshake* da especificação Openflow, através da inclusão de uma mensagem *Negotiation Request*, que indica quais recursos o dispositivo necessita de

acesso. Caso seja um recurso sensível, é realizada a troca de credenciais, caso contrário, o recurso é liberado instantaneamente. Para isso, os recursos são gerenciados através de uma política de controle de acesso, representada na forma normal disjuntiva (DNF - *Disjunctive Normal Form*), onde, por exemplo, para um recurso *Rc*1 de um Controlador ser acessado, primeiro ou o recurso do *switch Rs*2 ou o recurso *Rs*3 precisam ter sido autorizados para esse Controlador previamente.

Já na categoria Exposição de Informações, o trabalho de Conti, De Gaspari e Mancini [124] propõe uma solução de caminho seguro. Nesse sentido, o trabalho aborda o acesso a tabela de fluxos de um *switch*, através de diversas formas, como ao conectar em uma porta não protegida do *switch*, aproveitar de uma configuração pobre do *switch*, como uma senha fraca, ou mal configuração, usar a variação de RTT para inferir informações sobre as tabelas de fluxo, aproveitar-se da falta de TLS no canal de controle, explorar *back-doors* inseridos pelos fabricantes de dispositivos ou governos, ou comprometer o *switch*. De posse do acesso às tabelas de fluxos, o atacante pode desferir uma série de ataques, como infectar a rede com um *worm*, escanear a rede, realizar um DoS, obter informações sobre os mecanismos de detecção e prevenção de anomalias, através da injeção de pacotes na rede e análise das mudanças ocorridas nas tabelas de fluxos, além de correlacionar as configurações dos mecanismos de controle de acesso com as regras de fluxos, de forma a burlar esses mecanismos. Como mecanismo de segurança, é proposto o ofuscamento de alguns dados das regras de fluxo, durante o caminho dos pacotes até o *switch* de saída da rede, de forma que o atacante não consiga detectar que as regras de fluxos foram impactadas com a sua atuação, e com isso não conseguir obter informações relevantes das tabelas de fluxos para realizar os ataques.

Na situação categorizada como Falta de TLS, o trabalho de Zhang e Qiu [657] propõe resolver o ataque da categoria *Interception* com uma detecção de anomalias. Devido à falta de TLS, é possível que um atacante realize um ataque de MitM. Em virtude disso, este trabalho provê um IDS que organiza todos as regras de fluxos dos *switches* em uma tabela global, e monitora as informações relativas ao caminho global de um fluxo, o tempo de um pacote percorrer o caminho de um fluxo, o número de pacotes e bytes que cruzam o caminho de um fluxo, entre outras. Com base na comparação dessas informações, o IDS detecta a existência e a localização de um MitM, em virtude do tempo que o atacante utiliza para receber, processar e reenviar os pacotes pela rede.

Os resultados do SMS indicaram uma série de temas onde esforços significativos estão sendo investidos para solucioná-los (DoS, por exemplo) e vários outros tópicos estão em fase inicial de exploração, abrindo um longo campo de pesquisas. Este mapeamento, além de fornecer uma visão geral sobre as pesquisas envolvendo segurança em SDN, também auxiliou na identificação do problema de pesquisa, a qual será detalhada no Capítulo 4.

4. PROPOSTA DE SOLUÇÃO PARA O BALANCEAMENTO ENTRE SEGURANÇA E DESEMPENHO ENTRE CONTROLADOR E *SWITCH*

Através do mapeamento sistemático, apresentado no Capítulo 3, foi identificado que a falta de uso de uma comunicação segura no plano de controle de redes SDN é uma vulnerabilidade que possibilita uma série de ataques. Por outro lado, o uso de um canal seguro representa um aumento do uso de recursos dos dispositivos de rede. Nesse sentido, neste capítulo é proposta uma abordagem para realizar o balanceamento entre segurança e desempenho no plano de controle, de forma a prover segurança sem comprometer o desempenho da rede.

4.1 Modelo de ameaça

Um ambiente SDN pode ser comprometido por diversos motivos, como discutido anteriormente. Nesse sentido, foi definido o modelo de ameaça deste trabalho baseado no modelo de ameaça STRIDE [499] encontrado na literatura sobre SDN [426],[557],[295],[134].

Sendo assim, de acordo com o modelo STRIDE, possíveis ameaças são *spoofing*, *tampering*, *repudiation*, *information disclosure*, *DoS* e *elevation of privileges*. *Spoofing* significa que a identidade do atacante é mascarada para acessar recursos de forma ilícita. *Tampering* significa que os dados são manipulados maliciosamente (destruído ou adulterado). *Repudiation* significa que a autorização para efetuar uma ação pode ser negada, embora o agente da ação tenha autorização para realizá-la. *Information Disclosure* significa a exposição de informações para entidades não autorizadas. DoS, como explicado anteriormente, é a indisponibilidade de um serviço causado por um ataque. Por fim, *Elevation of Privilege* significa a expansão de uma permissão de um usuário [557].

A Open Networking Foundation [426] identificou várias vulnerabilidades na arquitetura SDN, utilizando o modelo STRIDE. Foi percebido que, no ambiente SDN, o *Spoofing* poderia ser explorado por um atacante para impersonar uma configuração Openflow visando modificar o conteúdo da configuração de um *switch* ou modificar as informações necessárias para estabelecer uma comunicação com o Controlador. Através da modificação de algumas informações de configuração, um atacante pode preparar outros tipos de ataques no futuro.

Além disso, depois que um atacante impersonar um Controlador ou executar um ataque MitM, ele pode realizar um *Tampering* nas configurações e dados vitais de um *switch*, adicionando ou modificando as mensagens Openflow. O software do Controlador ou os pacotes de atualizações podem ser modificados por uma entidade maliciosa (por exemplo, uma aplicação de gerenciamento) para realizar ataques. Por exemplo, um atacante

pode modificar as políticas de um Controlador e redirecionar o tráfego associado para um destino específico, a fins de interceptação do tráfego.

Consequentemente, um ataque usando *Repudiation* pode ser realizado por aplicações no Controlador, desabilitando algumas funções de rede, como um IDS ou um *Firewall*, por exemplo. Esse ataque também poderia ocorrer entre controladores, quando há uma hierarquia de controladores.

Adicionalmente, a *Information Disclosure* torna possível para um atacante preencher a tabela de fluxos com o objetivo de descobrir a sua capacidade. Isso é possível também para um atacante descobrir mais informações, realizando um ataque MitM. Por exemplo, o atacante pode fazer um *switch* demorar mais tempo para processar um pacote que pertence a um novo fluxo. Baseado nas informações descobertas, um atacante pode preparar outros tipos de ataques [428]. Por exemplo, ao realizar um ataque de inundação com um grande volume de novos fluxos em um *switch* com baixa capacidade de recursos, o atacante pode fazer o *switch* sofrer um DoS.

Quando a comunicação entre o *switch* e o Controlador não são encriptadas, um atacante pode monitorar o canal de comunicação para obter informações de configuração. O risco de sofrer um ataque de *Information Disclosure* inclui acesso não autorizado a dados sensíveis no Controlador, como backup de tabelas de fluxos, dados de configuração, dados de topologia, etc. O ataque envolvendo *Information Disclosure* pode ser causado também por recursos compartilhados, uma vez que diferentes aplicações provêm seus dados no mesmo Controlador. Um atacante pode acessar as portas abertas, os serviços em execução e a versão de softwares em um Controlador usando certas ferramentas de escaneamento.

Um atacante também pode modificar as entradas da tabela de fluxos para realizar um ataque de DoS. Se o atacante configurar um *switch* com vários fluxos com diferentes portas de entrada com destino a mesma porta de saída, isso pode fazer com que a porta exceda a sua capacidade. Quando as mensagens são transportadas através de um canal seguro, o processamento dessas mensagens atingem altos níveis de uso de CPU devido ao custo da criptografia. Como resultado, um atacante pode realizar um ataque de DoS em *switch* de baixa capacidade de recursos, enviando um vasto número de pacotes inconsistentes em um curto período. Em um ambiente SDN centralizado, o Controlador é responsável por o trabalho de gerenciamento de um vasto número de *switches* e aplicações. Então, eles podem escanear as portas abertas ou os serviços existentes no Controlador e trocar muitas mensagens concorrentemente com o Controlador, sobrecarregando o Controlador e atingindo com sucesso um ataque de DoS.

Por fim, depois de impersonar com sucesso um Controlador, um atacante pode ter a oportunidade de aumentar os seus privilégios no sistema (*elevation of privileges*). Por exemplo, o atacante pode adquirir a prioridade em certas filas, gerando ou modificando as políticas dos fluxos nas tabelas de fluxos e então garantir que os seus pacotes de dados

possam ser transferidos mais rápido pela rede. Uma aplicação maliciosa pode abusar da API do Northbound para aumentar os seus privilégios e não se submeter às regras administrativas ou às políticas de segurança. Essa situação pode ser realizada também ao criar uma política que conflite com as políticas administrativas ou com as aplicações de segurança.

Nesse sentido, é possível resumir as ameaças SDN em dez ameaças, de acordo com Jacquin, Shaw e Dalton [244]:

- T1 - o ataque MitM que intercepta e/ou altera os pacotes enviados pelos elementos da rede para o Controlador;
- T2 - o ataque MitM que intercepta e/ou altera os pacotes enviados pelo Controlador para os elementos de rede;
- T3 - *eavesdropping* passivo, como logar o comportamento do plano de controle;
- T4 - administração maliciosa ou acidental dos elementos de rede;
- T5 - *exploits* de dia zero das vulnerabilidades do *firmware* de elementos de rede;
- T6 - sobrescrita de regras realizada por uma aplicação SDN como, por exemplo, uma aplicação que quebra uma política de rede através da reconfiguração de um fluxo, usando o Controlador como meio;
- T7 - atualização de *firmware* de elementos de rede com softwares personalizados (incluindo softwares maliciosos, como *bootkits* persistentes;
- T8 - ataques físicos (substituição de chips, sondagens de barramento, etc.);
- T9 - Controlador *rogue* com permissão para alterar a configuração de elementos da rede;
- T10 - *downgrade* do firmware de elementos de rede para uma versão mais antiga (e potencialmente vulnerável).

Embora todas as vulnerabilidades supracitadas possam ser exploradas em um ambiente SDN, por questões de escopo, neste trabalho foi considerado que o gargalo de vulnerabilidades na arquitetura SDN pode ser representado pelo *switch*, em virtude do Controlador poder ser considerado uma entidade com hardware ilimitado (devido a facilidade de escalabilidade) comparado a um *switch*. Com isso, ao sobrecarregar o *switch* ou o canal de comunicação irá diretamente impactar nos recursos do *switch*, habilitando boa parte dos ataques que visam comprometer o *switch*. Sendo assim, neste trabalho foi considerado somente as ameaças T1 e T2 onde um atacante pode comprometer a rede através da realização de ataques MitM. Nesse sentido, o atacante está habilitado a ler, injetar e adulterar

mensagens Openflow no canal de comunicação do *Southbound*. Situações como ataques diretamente relacionados aos algoritmos criptográficos, ao Controlador, ao /textitswitch, ao *host* e ao plano de dados está fora do escopo deste trabalho.

4.2 ZONFlow

A segurança e o desempenho são requisitos que normalmente um deles é priorizado em detrimento do outro, porque quando os níveis de segurança são aumentados, o *overhead* imposto por essa decisão faz com que o desempenho diminua, e se o dispositivo necessita de alto desempenho, a segurança provavelmente será deixada em segundo plano. Nesse sentido, em um Southbound SDN padrão, como os protocolos apresentados no Capítulo 2, normalmente é sugerido o uso de, ou uma conexão segura usando TLS, ou uma conexão insegura padrão, de forma que ou todo o tráfego do plano de controle seja realizado de forma totalmente segura ou totalmente insegura. Se o *switch* estiver consumindo uma alta taxa de uso de CPU devido a carga de trabalho e a situação requer um tráfego seguro do plano de controle, os recursos do *switch* poderão ficar sobrecarregados.

Essa opção (desempenho *versus* segurança) impõe a necessidade de alternativas. Então, como poderia ser possível prover um nível de segurança e desempenho suficientes ao mesmo tempo? A resposta pode ser encontrada ao analisar cuidadosamente as mensagens do protocolo do plano de controle, uma vez que normalmente algumas mensagens são usadas para prover um ataque. Se somente essas mensagens cruzarem a rede através de um canal seguro e as outras mensagens do plano de controle forem transferidas através de um canal inseguro, poderia ser possível diminuir o *overhead* de uso de CPU imposto pela criptografia.

Entretanto, a taxa de mensagens sensíveis deve ser analisada, pois se esta taxa é alta, a economia de uso de CPU imposta por essa abordagem pode ser insignificante. Outro ponto a ser considerado é se o *overhead* imposto pelo gerenciamento das mensagens (necessário para decidir por qual canal cada mensagem deverá ser transferida) poderia produzir mais *overhead* que a transferência exclusivamente pelo canal seguro.

Nesse sentido, foi desenvolvido o ZONFlow, uma abordagem para prover segurança durante a comunicação do plano de controle e reduzir o *overhead* de uso de CPU no *switch* imposto pela criptografia. A metodologia da abordagem é apresentada na Figura 4.1. Inicialmente, deve-se explorar a literatura sobre vulnerabilidades nas mensagens do protocolo definido para a comunicação no plano de controle. Em seguida, categorizar as mensagens em dois conjuntos, o conjunto das mensagens sensíveis, ou seja, aquelas mensagens que apresentam alguma vulnerabilidade conhecida, e o conjunto das mensagens inofensivas, ou seja, aquelas mensagens que não se conhece nenhuma vulnerabilidade envolvida, de forma a explorar em algum ataque. Por fim, as mensagens do conjunto inofensivo serão

transmitidas, entre o Controlador e o *switch* por um canal de comunicação não seguro, e as mensagens do conjunto sensível serão transmitidas por um canal de comunicação seguro (criptografado e com autenticação mútua).

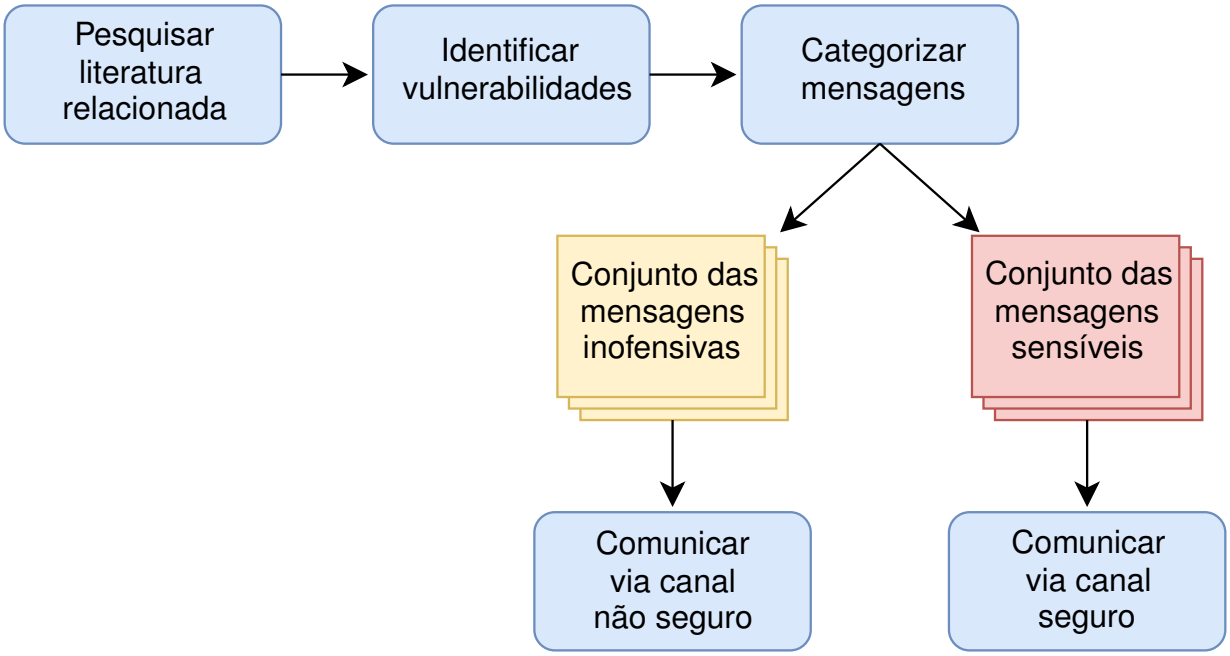

Figura 4.1 – Metodologia ZONFlow.

Para atingir esse objetivo, a solução é composta por dois módulos de gerenciamento de mensagens, ambos conectados um ao outro através de dois canais TCP (seguro e não seguro). Um módulo é executado no hardware do Controlador e conectado ao software (POX, Opendaylight, etc.) e o segundo módulo é executado no hardware do *switch* e conectado ao software de gerenciamento do *switch* (Open vSwitch, por exemplo), como ilustrado na Figura 4.2. O canal seguro é provido por uma conexão TLS com autenticação mútua.

Os módulos podem ser expressos baseado em uma modelagem formal. Considerando M como o conjunto de todas as mensagens do protocolo do *Southbound*, A como o conjunto de mensagens identificadas na literatura que podem ser usadas para realizar um ataque, e S como o conjunto de mensagens que ainda não são identificadas como vetores de ataque, pode-se expressar M como na Equação 4.1:

$$M = A \cup S, \text{onde } A \cap S = \emptyset \tag{4.1}$$

Considerando esses conjuntos como mensagens que serão trafegadas na comunicação do Southbound, o processo de receber uma mensagem de uma origem (Controlador ou *switch*) e enviar para o módulo ZONFlow de destino pode ser expresso pelo predicado *receberDaOrigem*(m, T), onde $m \in M$, $T = A \veebar S$. O predicado *receberDaOrigem*(m, T) é responsável por monitorar e receber uma mensagem oriunda da origem. Além disso, o

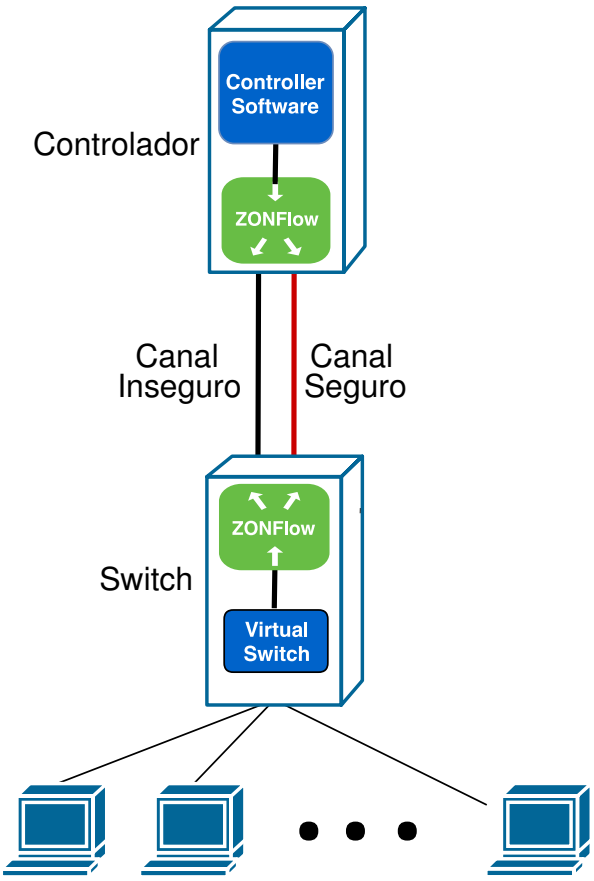

Figura 4.2 – Arquitetura ZONFlow.

predicado *enviar*(m, h) é responsável por enviar uma mensagem da origem para o módulo ZONFlow de destino, onde h é um canal de comunicação seguro (c) ou não seguro (p). Esses predicados fundamentam as Equação 4.2 e Equação 4.3, onde I é o conjunto de mensagens oriundas da origem, sendo $I = M$.

$$\forall i \in I : \textit{receberDaOrigem}(i, A) \rightarrow \textit{enviar}(i, c) \tag{4.2}$$

$$\forall i \in I : \textit{receberDaOrigem}(i, S) \rightarrow \textit{enviar}(i, p) \tag{4.3}$$

A Equação 4.2 expressa que, quando uma mensagem i advenha do dispositivo de origem (conjunto das mensagens de origem I), caso essa mensagem seja classificada como uma mensagem sensível (conjunto A), essa mensagem deverá ser encaminhada, ao módulo ZONFlow de destino, via canal de comunicação seguro (c). Seguindo a mesma lógica, a Equação 4.3 expressa que, caso a mensagem i advenha do dispositivo de origem e seja classificada como uma mensagem inofensiva (conjunto S), essa mensagem deverá ser encaminhada pelo canal de comunicação não seguro (p). As Equação 4.2 e Equação 4.3 podem também ser representadas através do Pseudocódigo 4.1.

Pseudocódigo 4.1 – Enviar mensagens do dispositivo de origem e encaminhar para o módulo ZONFlow de destino

```
1: Função enviarParaZONFlowDestino( )
2:     Se receberDaOrigem(i) Então
3:         Se i.pertence(A) Então
4:             enviarPeloCanalSeguro(i)
5:         Senão
6:             enviarPeloCanalNaoSeguro(i)
```

O Pseudocódigo 4.1 define a situação quando uma origem (Controlador ou *switch*) envia uma mensagem Openflow para um destino. O módulo ZONFlow está instalado no dispositivo de origem e se conecta com o software de origem através de uma conexão TCP insegura (*localhost*). Quando o módulo ZONFlow do dispositivo de origem recebe uma mensagem do software de origem, o módulo verifica se a mensagem pertence ao conjunto A (se é uma mensagem sensível). Se essa condição for verdadeira, então o módulo ZONFlow da origem envia a mensagem para o módulo ZONFlow do destino, através do canal seguro. Caso contrário, o módulo ZONFlow da origem envia a mensagem, através do canal não seguro, para o módulo ZONFlow do destino.

Consequentemente, o processo de receber uma mensagem do módulo ZONFlow de origem e enviar para o destino, pode ser expresso pelas Equação 4.4, Equação 4.5 e Equação 4.6, onde $O = M$ é o conjunto de mensagens que chegam ao módulo ZONFlow de destino, o predicado $receber(o, h, [T])$ representa o monitoramento e recebimento da mensagem o pelo canal de comunicação h, podendo ser um canal de comunicação seguro (c) ou não seguro (p), e $T = A \trianglelefteq S$, sendo uma informação opcional. O predicado $enviarParaDestino(o)$ representa o processo de envio da mensagem o, que chega do módulo ZONFlow de origem, para o destino (Controlador ou *switch*). Por fim, o predicado $eliminar(o)$ representa o descarte de uma mensagem o. A Figura 4.3 ilustra a representação do tráfego das mensagens.

$$\forall o \in O : receber(o, c) \rightarrow enviarParaDestino(o) \tag{4.4}$$

$$\forall o \in O : receber(m, p, S) \rightarrow enviarParaDestino(o) \tag{4.5}$$

$$\forall o \in O : receber(m, p, A) \rightarrow eliminar(o) \tag{4.6}$$

A Equação 4.4 expressa que, caso o módulo ZONFlow de destino receba uma mensagem o, e essa mensagem tenha sido recebida através do canal seguro (c), então essa mensagem deverá ser encaminhada ao destino. Da mesma forma, a Equação 4.5 expressa que, caso o módulo ZONFlow de destino receba uma mensagem o, e essa men-

sagem tenha sido recebida através do canal não seguro (p), essa mensagem deverá ser uma mensagem inofensiva (conjunto S), e assim, encaminhada ao destino. Por fim, a Equação 4.6 expressa que, caso o módulo ZONFlow de destino receba uma mensagem o, e essa mensagem tenha sido recebida pelo canal de comunicação não seguro, se essa mensagem seja classificada como sensível, então essa mensagem deverá ser descartada.

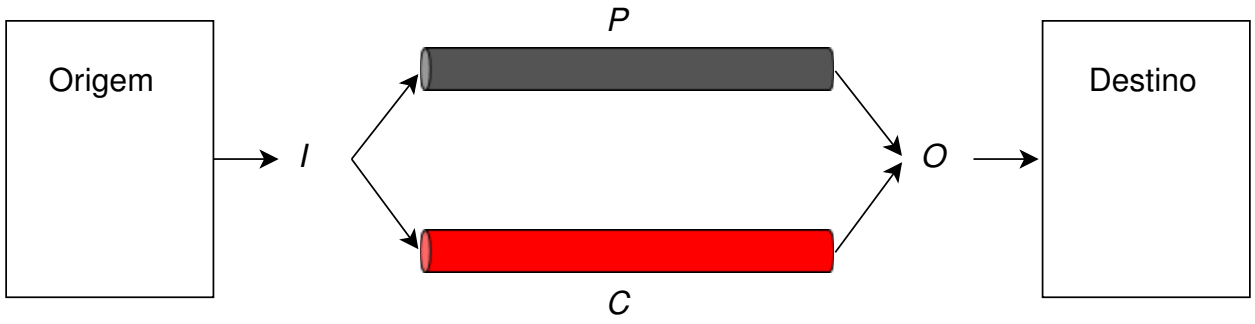

Figura 4.3 – Tráfego de mensagens.

Pseudocódigo 4.2 – Receber mensagens do módulo ZONFlow de origem e encaminhar para o dispositivo de destino.

```
1: Função receberDeZONFlowOrigem( )
2:     Se receberPeloCanalSeguro(o) Então
3:         enviarParaDestino(o)
4:     Se receberPeloCanalNaoSeguro(o) Então
5:         Se o.pertence(A) Então
6:             eliminar(o)
7:         Senão
8:             enviarParaDestino(o)
```

O Pseudocódigo 4.2 apresenta a situação quando um módulo ZONFlow de destino recebe uma mensagem a partir de um canal seguro ou não seguro. Quando a mensagem chegar ao módulo pelo canal seguro, então o módulo envia a mensagem para o destino (software do *switch* ou do Controlador). Quando a mensagem chegar ao módulo através do canal não seguro, se a mensagem é membro do conjunto A (mensagem sensível), então o módulo elimina a mensagem. Caso contrário, a mensagem é enviada ao destino. Se uma mensagem, que é membro de A chegar pelo canal não seguro, isso significa que a mensagem não foi enviada pelo módulo ZONFlow do dispositivo de origem, pois o módulo ZONFlow de origem encaminha as mensagens sensíveis exclusivamente pelo canal seguro. Ou seja, provavelmente a mensagem foi injetada no canal não seguro por um atacante. Em virtude disso, a mensagem é excluída.

Para verificar que a abordagem ZONFlow é imune às consequências de um ataque MitM, no qual intercepta e/ou altera os pacotes no plano de controle, enviados pelos elementos da rede para o Controlador (ameaça T1), e ataque MitM que intercepta e altera

os pacotes do plano de controle, enviados pelo Controlador para os elementos de rede (ameaça T2), pode-se realizar a seguinte prova:

Lema 4.2.1. *Toda mensagem $m \in A$, recebida no destino é oriunda da origem.*

Demonstração. Para provar o Lema 4.2.1 por contradição, assume-se a hipótese de que o Lema 4.2.1 é falso. Para que a mensagem *m* seja entregue ao destino, é necessário que ocorra o processo de recebimento de mensagens pelo módulo ZONFlow de destino (Equações 4.4, 4.5 ou 4.6). Como a Equação 4.4 não analisa se a mensagem é obrigatoriamente pertencente ao conjunto *A*, embora a Equação 4.3 direciona as mensagens do conjunto *S* para o canal de comunicação não seguro, um atacante poderia injetar essa mensagem no canal seguro, através de um ataque MitM. Porém, a comunicação entre os módulos ZONFlow ocorre através de TLS com autenticação mútua. A Tabela 4.1 apresenta as etapas que ocorrem durante a comunicação (considere um módulo ZONFlow como *Zin* e o outro módulo ZONFlow como *Zout*):

Tabela 4.1 – TLS com autenticação mútua [141]

Zin		*Zout*
Hello	$\longrightarrow$	
		Hello
		Certificate
		KeyExchange
		CertificateRequest
	$\longleftarrow$	HelloDone
Certificate		
KeyExchange		
CertificateVerify		
ChangeCipherSpec		
Finished	$\longrightarrow$	
		ChangeCipherSpec
	$\longleftarrow$	Finished
Message	$\longrightarrow$	

Como a troca de mensagens envolve a troca e a validação de certificados, e a comunicação é criptografada, a única forma possível de um atacante inserir uma mensagem no canal de comunicação ocorre quando a chave privada de *Zin* ou de *Zout* é comprometida. Como essa situação está fora do escopo deste trabalho, considera-se que as chaves privadas de ambos os módulos estão protegidas contra essa situação. Sendo assim, todas as mensagens que cruzam o canal de comunicação *c* são oriundas de *i*. Logo, a Equação 4.2 contradiz parcialmente a hipótese.

Porém, é possível que um atacante realize um ataque MitM no canal de comunicação não seguro (*p*), e injete mensagens oriundas do conjunto *A*. Mas a Equação 4.6 descarta qualquer mensagem de *A*, que chegue ao destino *o*, pelo canal de comunicação *p*. Logo, a Equação 4.6 também contradiz a hipótese. Dessa forma, o Lema 4.2.1 é

verdadeiro. Mas ainda é possível que o atacante injete mensagens do conjunto *S* no canal de comunicação *p*. Entretanto, como as mensagens dessa categoria são inofensivas, o atacante não poderá explorar essas mensagens para realizar um ataque, de forma que esse ataque não consegue causar maiores consequências. Contudo, mesmo com o ataque ocorrendo, o ambiente segue seguro. □

Para verificar que a abordagem ZONFlow traz melhorias no desempenho dos dispositivos, em termos de CPU, pode-se considerar o o custo de CPU α para gerenciar uma conexão TCP não segura como:

$$\alpha = \beta + \gamma + \sigma + \theta + \zeta \tag{4.7}$$

Onde β representa o consumo de CPU para realizar o *three way handshake*, γ é o consumo de CPU para empacotar e enviar uma mensagem, σ é o consumo de CPU para receber e desempacotar uma mensagem, θ é o custo, em termos de CPU para encerrar a conexão e ζ é o custo, em termos de CPU da execução da implementação ZONFlow. O consumo de CPU ζ dependerá de uma série de fatores, como a linguagem de programação utilizada, o tamanho do conjunto de mensagens sensíveis e o conjunto das mensagens inofensivas, porém, como os conjuntos de mensagens serão relativamente pequenos, o custo de CPU para percorrer os elementos dos conjuntos será desprezível, ficando o impacto de ζ a cargo da implementação dos módulos. Caso a implementação seja nativa, esse custo será aproximadamente o mesmo que os dispositivos já produzem para gerenciar as conexões, de forma que a diferença de custo de CPU entre a implementação dos códigos executados pelos dispositivos para gerenciar conexões e a implementação do módulo ZONFlow será desprezível. Porém, se a implementação dos módulos ZONFlow não forem nativas aos equipamentos, esse custo de CPU pode ser significativo. Ao considerar um tempo relativamente longo de conexão, o consumo de CPU β e θ acabam sendo insignificantes. Sendo assim, pode-se considerar α basicamente como:

$$\alpha = \sum \gamma + \sum \sigma + \zeta \tag{4.8}$$

Já em uma conexão segura, o custo de CPU α pode ser representado como:

$$\alpha = \beta + \gamma + \omega + \phi + \sigma + \theta + \zeta \tag{4.9}$$

Onde β compõe o consumo de CPU para realizar o processo de definição do algoritmo criptográfico, trocas de certificados e autenticações e definição de chave. Já ω e ϕ representam o consumo de CPU para criptografar e descriptografar as mensagens, respectivamente. Considerando um tempo de conexão relativamente longo, o custo β nesse tipo de conexão também acaba sendo desprezível, de forma que pode-se considerar α como:

$$\alpha = \sum(\gamma + \omega) + \sum(\phi + \sigma) + \zeta \tag{4.10}$$

Como o algoritmo ZONFlow utiliza duas conexões para trafegar as mensagens (uma segura e uma não segura), e o tráfego ocorrerá durante um período de tempo relativamente longo, o custo de criar as conexões acabam sendo desprezíveis, como representado nas equações 4.7, 4.8, 4.9 e 4.10. Sendo assim, o custo de CPU para gerenciar as conexões se resume ao processo de envio e recebimento das mensagens. Nesse sentido, considere a seguinte prova:

Lema 4.2.2. *Dada uma série de mensagens, as mensagens trafegadas exclusivamente por um canal não seguro sempre consumirão menos recursos de CPU que as mensagens trafegadas exclusivamente por um canal seguro.*

Demonstração. Para provar isso por contradição, considere que a afirmação é falsa. Como a comunicação segura possui duas etapas a mais que a comunicação não segura (ω e ϕ), o consumo de CPU ω e ϕ precisaria ser negativo ou igual a zero, o que acaba sendo uma contradição. □

Dessa forma, quanto mais mensagens trafegarem pelo canal não seguro, menor será o consumo de CPU total da comunicação, em comparação a uma comunicação totalmente via um canal seguro. Logo, a menos que todas as mensagens sejam sensíveis, a abordagem ZONFlow apresentará ganhos em termos de consumo de CPU, pois as mensagens inofensivas serão transmitidas via um canal não seguro.

Lema 4.2.3. *Dada uma série de mensagens $a \in A$ e $s \in S$, sendo $\sum(a)$ inversamente proporcional a $\sum(s)$, $(\sum a + \sum s) \times \alpha_c >= (\sum a + \sum s) \times \alpha_{ZONFlow}$. Onde α_c representa o custo de CPU para realizar uma comunicação no canal seguro $c \in C$ e $\alpha_{ZONFlow}$ representa o custo de CPU para realizar uma comunicação utilizando a abordagem ZONFlow.*

Demonstração. Para provar isso por contradição, considere que a afirmação é falsa. Como a abordagem ZONFlow só terá os valores de α iguais a conexão estritamente segura (pertencente ao conjunto *C*) quando 100% das mensagens serem sensíveis (pertencentes ao conjunto *A*), nos demais casos o tráfego das mensagens provenientes do conjunto *S* (inofensivas) ocorrerá pelo canal não seguro (pertencente ao conjunto *P*), fazendo com que o α da abordagem ZONFlow seja igual a:

$$\alpha_{ZONFlow} = \frac{[\sum(a) \times \alpha_c] + [(\sum(s)) \times \alpha_p]}{\sum(a) + \sum(s)} \tag{4.11}$$

Sendo assim, sempre que que $\sum s > 0$, $\alpha_c > \alpha_{ZONFlow}$. Somente se $\sum s = 0$, $\alpha_c = \alpha_{ZONFlow}$, o que acaba sendo uma contradição. □

4.3 Trabalhos relacionados

O balanceamento entre segurança e desempenho, no contexto de redes, é pouco explorado na literatura. Não foram identificados trabalhos que explorem o paradigma SDN nesse contexto, mas foram identificados trabalhos como o de Tesfamicael et al. [564], que propõe um modelo para balancear segurança e desempenho para ambientes de nuvem que provêm o serviço de *trading communication system - TCS*. Para isso, este estudo categorizou níveis de segurança baseados nas opções de configurações de segurança disponíveis de Rede privada virtual (*Virtual Private Network* - VPN) baseada em IPSec, os quais podem ser o algoritmo AES com tamanhos de chave de 128, 192 ou 256 bits, para o processo de criptografia dos dados e SHA1 ou SHA256 para autenticação, e analisou o impacto do desempenho desses níveis em termos de *jitter*, latência, perda de pacotes e *Mean Opinion Score (MOS)*, de acordo com os padrões definidos para esse tipo de serviço pelo ITU-T G.114 [563]. Com base nisso, foi proposto um modelo de configurações para prover um desempenho dentro dos padrões recomendados pela norma, para cada nível de segurança desejado. O modelo foi obtido através de simulações de diferentes ambientes que apresentam a necessidade desse tipo de serviço.

O trabalho desenvolvido por Silva et al. [532] propõe um esquema adaptativo e *cross-layer* para balancear desempenho e segurança no roteamento de dados em Wireless Mesh Networks (WMN), mesmo em cenários onde a rede apresenta ações maliciosas realizadas por nós comprometidos. Para isso, o esquema utiliza um método de tomada de decisão multi-critério para oferecer uma métrica de roteamento, de forma a auxiliar na seleção de caminhos de protocolos de roteamento. O nível de segurança dos caminhos é estimado, levando em consideração os critérios usados pelos nós para definir os mecanismos de segurança preventivos, reativos e tolerantes. As defesas preventivas compreendem os mecanismos de segurança que tentam prevenir ataques, como o uso de criptografia, Firewall e técnicas de controle de acesso. As defesas reativas tentam detectar e reagir contra intrusões, como sistemas de reputação e IDS. As defesas tolerantes buscam mitigar os danos causados por ataques ou intrusões, e recuperar os serviços comprometidos.

O esquema é composto por quatro funções: o nível de segurança do caminho, o custo de desempenho do caminho, cálculo das métricas de roteamento e informações trocadas entre os nós. Devido a infraestrutura descentralizada de uma WMN, cada nodo calcula localmente o nível de segurança e o custo de desempenho baseado nas informações trocadas com os outros nós. Para isso, o custo de segurança é calculado através da probabilidade de um caminho prevenir ações de nós maliciosos em dados criptografados (por exemplo, a probabilidade de garantir integridade, confiabilidade e autenticidade dos dados), a probabilidade dos nós de um caminho trabalharem corretamente no encaminhamento de pacotes de dados e a probabilidade de suportar redundância em casos de falhas

ou ataques na rota. Já o custo de desempenho do caminho é calculado através do número de saltos entre a origem e o destino, e do custo de desempenho de cada nó do caminho. O custo de desempenho do nó é calculado com base no número de canais disponíveis e no custo de cada canal, mensurado através do nível de interferência. Sendo assim, quanto menor for o custo de desempenho de um nó, maior é a probabilidade desse nó escolher um bom canal, uma vez que altos níveis de interferência resultam em mais colisões, atraso nos pacotes ou retransmissões. A métrica de roteamento é definida, para cada protocolo de roteamento, através no nível de segurança dos caminhos e do custo de desempenho dos caminhos. Por fim, esses dados são trocados entre os nós para compor o esquema. Esse esquema foi avaliado através de simulações, considerando a taxa de entrega dos pacotes (PDR - Packet Delivery Ratio), o atraso fim a fim (E2E - End-to-End Delay) e a quantidade de pacotes de dados que foram excluídos devido a ataques (APD - Amount of data Packets Dropped due to attacks). Para isso, foram realizados alguns tipos de ataques e verificado o desempenho dos algoritmos de roteamento com e sem o esquema. Os resultados obtidos indicaram que os algoritmos utilizando o esquema obtiveram um desempenho melhor do que quando executados isoladamente, mantendo os níveis de segurança necessários para proteger a rede contra os ataques realizados.

No trabalho realizado por Youssef et al. [649], foi desenvolvida a arquitetura de um sistema de segurança dinâmico para comunicação wireless 4G LTE, de forma a considerar a relação entre recursos, custos e riscos para o refinamento do perfil de segurança da rede, de forma a otimizar o desempenho de redes de comunicação. Para isso, foram categorizados níveis de segurança, com base em uma série de critérios, tais como o algoritmo de criptografia utilizado e o tamanho da chave de criptografia. Essa categorização considera também do desempenho da rede, como o volume de tráfego, condições do rádio (taxa média de erros nos bits comunicados), entre outros. O conceito da arquitetura é baseado em segmentar uma taxa de ligação recebida em grupos, de acordo com a aplicação utilizada. Em seguida, é aplicado um modelo de máquina de estados finita para cada cenário de ligação, de acordo com a classificação realizada. Dessa forma, um usuário pode utilizar duas portadoras, com dois modos diferentes ao mesmo tempo, um modo configurado com um nível de segurança crítico na portadora de um rádio dedicado, e o segundo modo configurado com um nível de segurança baixo na portadora do rádio padrão, dependendo da aplicação. A arquitetura foi validada em um ambiente de simulação, representando uma rede wireless simples, com diferentes níveis de tráfego e perfis de rede. Os resultados obtidos indicaram que o uso da arquitetura faz com que os recursos da rede sejam melhores aproveitados do que utilizando uma configuração estática, aumentando assim a qualidade da experiência no uso da rede.

De posse das informações referentes aos trabalhos relacionados, se percebe a dificuldade de comparação com a abordagem proposta neste estudo, devido ao contexto de SDN não ter sido explorado ainda do ponto de vista da segurança vs desempenho.

Sendo assim, ao analisar a abordagem ZONFlow, percebe-se que o tamanho do conjunto de mensagens sensíveis é uma limitação da abordagem, pois se o tamanho do conjunto de mensagens sensíveis for muito próximo ao total de mensagens do protocolo de comunicação, provavelmente o esforço de implantar a abordagem não será conveniente. Da mesma maneira, dependendo da forma que a abordagem for implantada no ambiente SDN, o custo de CPU gerado pela implementação da abordagem, somado ao custo de CPU dos tráfegos das mensagens, poderão ser superiores ao custo de uma infraestrutura tradicional, formada por um único canal de comunicação seguro, sendo essa também uma limitação da abordagem proposta neste trabalho.

Como as provas formais representam um ambiente ideal, para verificar a validade da abordagem proposta é necessário aplicá-la em um ambiente de experimentação. Sendo assim, no Capítulo 5 será abordado uma série de experimentos e uma discussão sobre os resultados obtidos.

5. ESTUDO DE CASO - OPENFLOW

Como as especificações Openflow oferecem uma vasta gama de diferentes mensagens, elas trazem um dilema que pode ser descrito como uma evolução das arquiteturas de rede versus um aumento perigoso no escopo das ameaças [258]. Sendo assim, é necessário compreender o impacto deste novo paradigma de rede frente a requisitos não funcionais, como segurança e desempenho.

5.1 Impactos do protocolo Openflow na segurança

A segurança é uma das maiores ameaças em SDN [198], e se pode considerar os vetores de ameaças baseados em três perspectivas: aquelas originadas pelo Controlador, originadas pelo *switch* (incluindo hosts) e originados do canal de comunicação entre Controlador e *switch*. Os ataques que são oriundos do Controlador podem ser causados por várias vulnerabilidades, como má configuração do Controlador [369], aplicações maliciosas [319], problemas relacionados a Autenticação, Autorização e Auditoria [529] ou *bugs* de software (loops persistentes, falhas de sincronização, estados inconsistentes, omissão de responsabilidade, etc.) [667]. Uma vez que o Controlador esteja comprometido, um atacante pode alterar as regras nos dispositivos e negar um acesso de um usuário legítimo aos recursos disponíveis [7].

Os ataques que são realizados a partir do *switch* podem ser causados por um processo de atualização da rede, por exemplo, onde algumas políticas do *switch* podem ficar obsoletas por um tempo [346]. Desta forma, os ataques podem ocorrer quando o *switch* está comprometido, uma vez que o atacante estará habilitado a modificar as entradas de fluxos do *switch* devido ao comportamento malicioso ou poderá se conectar a um Controlador malicioso e manipular o tráfego de controle da forma que o atacante desejar [110].

Por fim, os ataques que são iniciados a partir do canal de comunicação podem ser causados pela falta de suporte a TLS da implementação da especificação Openflow, por exemplo. Alguns estudos [377] [500] [557] [295] [362] [190] [322] [10] [406] indicam que a falta de suporte a TLS pode comprometer a segurança, e esta vulnerabilidade pode permitir que atacantes realizem *spoofing*, adulteração de dados que estão circulando no canal de comunicação, repúdio e ataques de MitM [87][657][136].

Além disso, a própria implementação da interface do Southbound pode oferecer vulnerabilidades. Os estudos encontrados na literatura indicam que o protocolo Openflow pode ser um vetor de várias ameaças. Foi desenvolvida uma análise de segurança da especificação do Openflow [614], e de acordo com a especificação Openflow [174]: "*The switch and controller **may** communicate through a TLS connection...*", ou seja, o Openflow

inclui uma funcionalidade de segurança opcional, que permite o uso de TLS [19] em um canal de controle. Este mecanismo tem como objetivo prevenir os ataques relacionados a impersonalização de um *switch* ou de um Controlador na rede [489], através de um processo de autenticação (caso os certificados tenham sido verificados apropriadamente em ambas as direções). O TLS permite também o uso de criptografia nas informações trafegadas no canal de controle, com o objetivo de prevenir ataques que visam obter ou adulterar os dados que trafegam na rede [142]. A especificação menciona também que conexões auxiliares podem usar UDP com DTLS, mas não foram realizadas discussões de como efetivar essa funcionalidade.

A análise supracitada concluiu que a segurança em Openflow foi minimamente especificada, ao ponto que a diferença de implementação entre os desenvolvedores pode causar complexidade operacional, problemas relacionados à interoperabilidade e vulnerabilidades inesperadas. Por exemplo, a especificação indica a seguinte situação: "*when using plain TCP, it is recommended to use alternative security measures to prevent eavesdropping, controller impersonation or other attacks on the OpenFlow channel.*". Entretanto, a especificação não apresenta um indicativo de que tipo de medidas alternativas de segurança são necessárias para prevenir esses ataques.

Além disso, a especificação não aborda, caso seja definido o uso de TLS, quais tipos de certificados podem ser utilizados (por exemplo, RSA [301], Diffie-Hellman [247], Diffie–Hellman com curva elíptica [375], Diffie–Hellman com curva elíptica efêmera [193], chave pré-compartilhada [159], senha remota segura [562], em combinação com os algoritmos de *hash* MD5 [578], SHA1 [157], SHA224 [229], SHA256 [577], SHA384 [577] ou SHA512[577], e com os algoritmos de criptografia simétrica apresentados no Capítulo 1), nem ao menos quais campos dos certificados podem ser usados. Esse problema traz complicadores para a operacionalidade, como falhas de interoperabilidade, pois um determinado fabricante não consegue presumir quais tipos de certificados serão disponibilizados por outro fabricante, na hora de prover o suporte a uma conexão segura. Outro problema levantado é a falta de definição de qual versão do TLS utilizar, podendo ocorrer situações que diferentes dispositivos utilizem diferentes versões de TLS. Como o TLS versão 1.1 corrige várias falhas de segurança do TLS 1.0, vários tipos e ataques podem ocorrer ao utilizar a versão mais antiga. Por fim, a fragilidade da especificação sobre a utilização de múltiplos Controladores compromete a questão de utilizar, na camada de aplicação, muitos aplicativos, por falta de suporte para autorização e de um controle de acesso granular.

Considerando as mensagens disponíveis nas especificações Openflow (Tabela 2.1), a mensagem OFPT_FEATURES_REPLY pode ser usada para realizar o ataque *Switch Table Flooding*, que ocorre quando um *switch* falso é conectado na rede e envia várias mensagens OFPT_FEATURES_REPLY ao Controlador com diferentes PIDs. Entretanto, para poder alterar o PID é necessário que o *switch* falso se reconecte na rede [150]. Um outro ataque que pode ser realizado é o denominado *Switch Identification Spoofing Attack*, que

ocorre quando um atacante conecta um *switch* falso na rede com o mesmo PID e nome de um *switch* atualmente em uso. Quando o Controlador detecta essas inconsistências, ele desconecta o primeiro *switch*, ou seja, o *switch* original. Depois de alguns segundos, o *switch* original irá reconectar na rede, e o *switch* falso irá ser desconectado e em seguida tentará a reconexão. Esse ciclo irá gradualmente degradar o desempenho da rede a medida que as entradas das tabelas de fluxos do *switch* legítimo expiram [149].

A mensagem OFPT_PORT_STATUS também pode ser usada para realizar um ataque Switch Table Flooding [150]. Para realizar este ataque, considere uma conexão entre um Controlador e um *switch*. Após o *handshake* entre eles ser completado, o atacante envia uma mensagem falsa OFPT_FEATURES_REPLY para o Controlador, com um PID diferente do *switch* original. Quando o Controlador receber a mensagem OFPT_FEATURES_REPLY adulterada, ele irá alterar o registro do PID do *switch* original para o PID adulterado, embora o Controlador não altere o registro da porta física nem a entrada do *switch* seja removida da tabela de *switches* do Controlador. Então, se o atacante enviar várias mensagens OFPT_FEATURES_REPLY falsas, os recursos do Controlador irão ser drenados lentamente. De forma a acelerar o ataque, é possível enviar várias mensagens PORT_STATUS para o Controlador, adicionando assim novas portas ao registro do *switch* falso, com o objetivo de aumentar a quantidade de dados armazenados na memória do Controlador. Se considerar que o atacante não tem controle do *switch* original, então o vetor de ameaça será somente o canal de comunicação. Sendo assim, o uso de um canal seguro poderá desabilitar a injeção de mensagens falsas OFPT_FEATURES_REPLY e OFPT_PORT_STATUS no canal, tornando este ataque inviável.

Já a mensagem OFPT_PACKET_IN pode ser enviada pelo *switch* para o Controlador com um pacote capturado, como resultado de uma correspondência que requisita o envio desse tipo de mensagem, ou como resultado de um *miss* nos campos de correspondência, ou como um resultado de erro de *Time To Live* (TTL). Esse comportamento pode ser usado para prover um ataque de DoS no Controlador, uma vez que um atacante pode forjar ou alterar os pacotes para forçar os *switch* a enviar exaustivamente mensagens OFPT_PACKET_IN. Um canal seguro pode proteger os pacotes contra a leitura ou modificação, embora não proteja contra dois tipos de ataques DoS, uma vez que os pacotes são criados no plano de dados: i) aquele ataque em que o Controlador responde à mensagem OFPT_PACKET_IN com um OFPT_FLOW_MOD (*Data-To-Control Plane Saturation Attack*), consumindo assim os recursos do *switch* [228][515][223]; ou ii) aquele ataque em que o Controlador responde com mensagens OFPT_PACKET_OUT , causando assim um *Packet Injection Attack* [138].

Outro vetor de ameaças é a mensagem OFPT_PACKET_OUT, que é enviada pelo Controlador para o *switch*, que pode ou carregar um *raw packet* para ser injetado no *switch*, ou indicar um *buffer* no *switch* contendo um *raw packet* para ser lançado. Esta habilidade é usada pelo Controlador para injetar pacotes no plano de dados de um *switch* em particular.

Quando um pacote padrão chega no *switch*, os campos de cabeçalho do pacote são comparados com as regras de fluxo para verificar se alguma regra de fluxo é correspondente, executando assim a ação da regra de fluxo; entretanto, como o pacote encapsulado em uma mensagem OFPT_PACKET_OUT tem o seu próprio campo de ação, o pacote salta para o estágio de aplicação da ação no *pipeline* padrão do processamento de pacotes. Esse detalhe pode ser explorado para realizar um ataque de DoS no *switch*, uma vez que várias mensagens OFPT_PACKET_OUT podem ser enviadas para o *switch* gerenciar a execução dos campos de ações, sobrecarregando os recursos do *switch* [227][138].

Por fim, a mensagem OFPT_FLOW_MOD, que é responsável por inserir, remover ou atualizar as regras de fluxos de um *switch*, pode ser usada para realizar um ataque de DoS no *switch*, uma vez que ela consome recursos significativos de memória e processamento [228]. Alguns exemplos de ataques de DoS que exploram a mensagem OFPT_FLOW_MOD são o *Flow Table Overflow* [662], *New Flow* [630], e *Flow Modification Suppression* [579].

Considerando todas os tipos de mensagens disponíveis na especificação Openflow (Tabela 2.1, somente essas mensagens foram identificadas como vetores de ataque na literatura. Não foram encontrados estudos que exploram outras mensagens das especificações Openflow em temas relacionados à segurança. Sendo assim, neste trabalho as outras mensagens das especificações não serão consideradas como vetores de ataque, embora elas possam ser consideradas como vetores de ataque no futuro e deverá ser necessário analisar a inclusão delas neste estudo posteriormente.

5.2 Impactos do protocolo Openflow no desempenho

Várias das preocupações relacionadas à segurança que foram supracitadas poderiam ser solucionadas utilizando uma conexão segura no plano de controle. A especificação Openflow [173] sugere o uso de TLS para proteger o plano de controle.

Entretanto, somente 50% dos fabricantes de equipamentos Openflow provêm suporte a TLS [489]. Uma causa desse baixo índice pode ser devido ao custo de desempenho imposto por uma conexão segura [154]. Essa preocupação com desempenho pode ser resolvida, do ponto de vista do Controlador, aumentando as características de hardware ou usando Controladores distribuídos [204]. Nesse sentido, o *switch* acaba sendo o gargalo, uma vez que o *switch* possui um hardware limitado e, de acordo com os princípios da SDN, o *switch* deve ser considerado um simples encaminhador de pacotes, deixando as atividades que demandam uma maior complexidade para o Controlador gerenciar. Devido a isso, os fabricantes de *switch* devem escolher cuidadosamente as funcionalidades que consomem altos níveis de recursos quando projetar um novo *switch* Openflow que irá suportar comunicações seguras.

Por exemplo, o processo de adicionar várias entradas de fluxos em uma tabela de fluxos representará um atraso significativo [78]. NO trabalho proposto por Aliyu, Bull e Abdallah [29], foi calculado o custo de inserir um novo fluxo baseado no atraso em questão de tempo como $\eta = \alpha + \beta + \rho + \lambda + \theta + \tau + \gamma$, onde α é o tempo consumido pela memória do *switch* no processo de localizar a regra de fluxo, β é o tempo consumido para um fluxo, que não foi localizado na tabela de fluxos, ser encaminhado para o Agente Openflow (OFA - OpenFlow Agent) no *switch*, ρ é o tempo que o Agente Openflow leva para encapsular o pacote e gerar uma mensagem OFPT_PACKET_IN, λ é o atraso causado ao invocar o canal de comunicação do plano de controle, θ é o tempo despendido pelo Controlador para determinar como os pacotes deverão ser manipulados baseado nas configurações das políticas estabelecidas ou o estado global da rede, τ é o atraso incorrido pela ação de enviar a regra de fluxo para o *switch*, com o intuito de atualizar a tabela de fluxos, e γ é o tempo gasto pelo *switch* para encaminhar os pacotes baseado na regra de fluxo. Entretanto, essa pesquisa não levou em consideração o atraso imposto pelo consumo de recursos no *switch* ao gerenciar a conexão segura.

No estudo desenvolvido por Sattar e Matrawy [497], foi proposto um modelo empírico do Open vSwitch para caracterizar diferentes atrasos de processamento (atrasos não relacionados com a rede) que afetam o desempenho do Open vSwitch. A metodologia para calcular o modelo empírico consiste de dados de tempo de processamento coletados de cada experimento realizado e então calcular as suas frequências. Em seguida, essas frequências são convertidas em frequências relativas e as frequências relativas acumuladas são calculadas para produzir o a Função de Distribuição Cumulativa Empírica (ECDF - *Empirical Cumulative Distribution Function*). Entretanto, o estudo não considerou os recursos consumidos pelo Open vSwitch para gerenciar as conexões seguras. Se isso fosse levado em consideração, os dados de tempos de processamento poderiam ser substancialmente diferentes do observado. Além disso, vários estudos [549],[480] [307] [358] [215] [216] [334] também têm investigado o desempenho de *switches* SDN, embora nenhum dos estudos levou em consideração as implicações da segurança.

5.3 Experimentos realizados

De forma a validar a proposta deste trabalho, foi desenvolvida uma série de experimentações, as quais são descritas, analisadas e discutidas no decorrer deste capítulo.

5.3.1 Ambiente de experimentação

O ambiente de hardware utilizado para realizar os experimentos foi composto por um Intel® Core™ i7-6500U CPU @2.50GHz x 4, com 16GB de RAM e Ubuntu 17.04 64 bits. Nesse hardware foram elaborados três cenários, onde todos eles foram compostos por uma máquina virtual de 1GHz x 1 CPU, 2GB de RAM e Ubuntu 17.04 64 bits, representando o dispositivo Controlador e executando o software Controlador POX versão dart[458]. O *switch* do primeiro cenário foi simulado por uma máquina virtual de 1GHz x 1 CPU, 2GB de RAM e Ubuntu 17.04 64 bits, executando o software Open vSwitch 2.9.0. Essa especificação visou ser similar aos hardwares do *switch* PICA8 P-3297 [452][554] e do *switch* Edge-Core AS4600-54T [410][396]. O *switch* do segundo cenário foi simulado por uma máquina virtual de 625 MHz x 3 CPU, 512 MB de RAM e Ubuntu 17.04 64 bits, rodando o software Open vSwitch 2.9.0. Essa especificação visou ser similar ao hardware do *switch* HP 2920-24G [220]. O *switch* do terceiro cenário foi simulado por uma máquina virtual de 1,5GHz x 2 CPU, 8 GB de RAM e Ubuntu 17.04 64 bits, executando o software Open vSwitch 2.9.0. Essa especificação visou ser similar ao hardware do *switch* Juniper QFX5100 [259]. Nos três cenários, as máquinas virtuais foram conectadas por uma conexão Ethernet virtual com um controle de tráfego de 1 Gbps. Para esses experimentos, os protótipos dos módulos ZONFlow foram implementados na linguagem Python versão 3.6.4.

O primeiro experimento simulou um ataque DoS, disparado por um ataque MitM, no canal de comunicação entre o Controlador e o *switch*. Para esse experimento, foi utilizado o primeiro cenário do ambiente de experimentação supracitado. Neste experimento foram elaboradas duas situações. A primeira situação representa um ataque MitM realizado em um ambiente SDN clássico (Figura 5.1(A)), e a segunda situação representa um ataque MitM realizado no ambiente SDN onde é aplicada a abordagem ZONFlow (Figura 5.1(B)). Para isso, em ambas situações foram injetadas no canal de comunicação pelo atacante 10.000 mensagens OFPT_FLOW_MOD, em direção ao *switch*. Para cada mensagem enviada, os parâmetros da mensagem foram alterados com o objetivo de popular a tabela de fluxos do *switch* com fluxos com diferentes endereços de origem, destino e ações.

5.3.2 Descrição dos experimentos

Para iniciar o experimento, primeiro foi enviado uma mensagem OFPT_STATS_REQUEST a partir do Controlador, com o objetivo de solicitar as regras de fluxos registradas no *switch*. Depois disso, as regras de fluxos existentes no *switch* foram contadas, através da mensagem de resposta do *switch*. Em seguida, foi iniciado o ataque MitM, injetando as mensagens OFPT_FLOW_MOD. Por fim, foi enviado novamente a mensagem

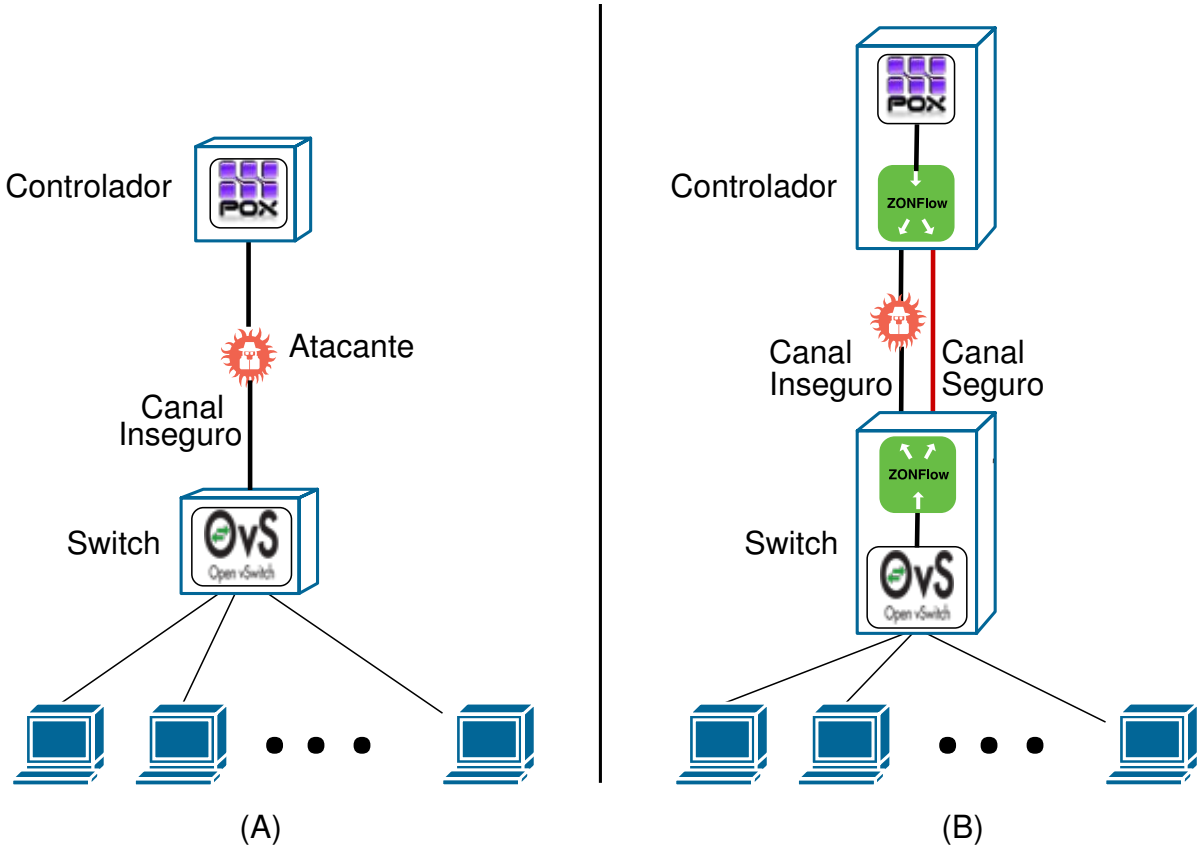

Figura 5.1 – Ambientes do primeiro experimento.

OFPT_STATS_ REQUEST e foram contados novamente o número de regras de fluxos retornados pelo *switch*. Por fim, o total de regras de fluxos retornados pela segunda requisição é subtraída do total de regras de fluxos retornados na primeira requisição, de forma a mensurar quantas regras de fluxos foram aceitas pelo *switch*. Esse experimento foi repetido 100 vezes para cada situação, e os resultados obtidos são apresentados na Tabela 5.1.

Tabela 5.1 – Mensagens injetadas através do ataque MitM

Mensagens injetadas	**Fluxos armazenados**
Comunicação totalmente insegura	
10.000	10.000
Comunicação via ZONFlow	
10.000	0

O segundo experimento foi conduzido de forma a simular um ataque de DoS no *switch*, oriundo de um atacante com acesso ao Controlador (de posse de uma aplicação maliciosa, por exemplo). Para isso, foram enviadas 1.000 mensagens OFPT_FLOW_MOD por segundo ao *switch*, utilizando o primeiro cenário de infraestrutura. A Figura 5.2 ilustra o ambiente do experimento. Nesse sentido, foi monitorado o comportamento da CPU do *switch*, através do monitoramento do uso de CPU consumido pelo Open vSwitch durante 60 segundos. Inicialmente, o experimento foi realizado através de um canal de comunicação totalmente inseguro (Figura 5.2 (A)), e em seguida o experimento foi realizado novamente,

agora através de um canal de comunicação seguro. (Figura 5.2 (B)). Cada experimento foi repetido 100 vezes, e foi calculada a média de consumo de CPU, com um intervalo de confiança de 95%. A Figura 5.3 apresenta os resultados alcançados. A margem de erro foi de $\pm$ 2,8% de uso de CPU.

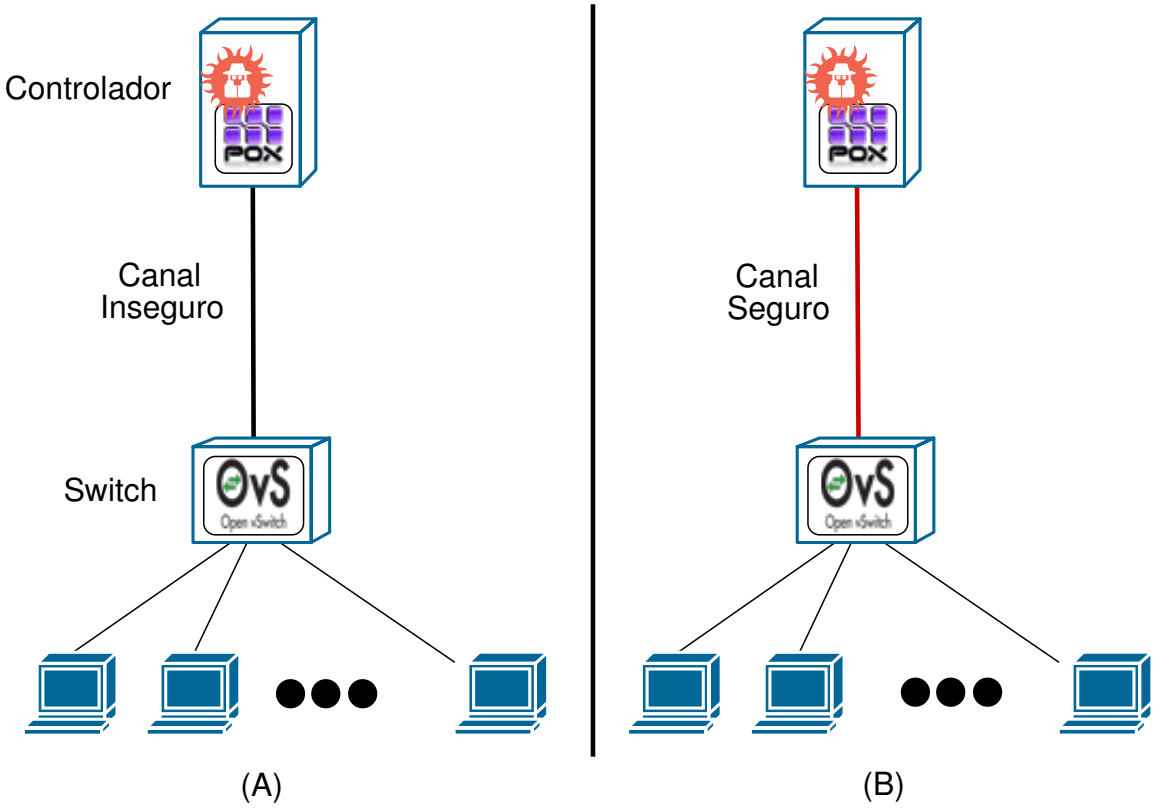

Figura 5.2 – Ambientes do segundo experimento.

O terceiro experimento desenvolvido buscou identificar o consumo de CPU pelo *switch* causado pela comunicação com diferentes cargas de processamento, visando analisar o balanceamento entre segurança e desempenho. Para isso, o experimento foi composto por três ambientes, como ilustrado na Figura 5.4: um formado por uma conexão insegura entre o Controlador e o *switch* (Figura 5.4 (A)); um formado por uma conexão segura (Figura 5.4 (B)); e uma formada por uma conexão utilizando ZONFlow (Figura 5.4 (C)). Nesses ambientes, foram enviadas 1.000 mensagens por segundo, durante 60 segundos, do Controlador para o *switch*. Foram selecionados dois tipos de mensagens para serem enviadas OFPT_FLOW_MOD e OFPT_PORT_STATUS. O primeiro tipo de mensagem foi selecionado devido a ser uma mensagem que pode ser usada para realizar um ataque no *switch*, pois uma vez que o *switch* necessita realizar leituras e escritas nas tabelas de fluxos para processar essa mensagem, ela consome altos níveis de uso de CPU. O segundo tipo de mensagem foi escolhido devido ao fato que, esta mensagem pode ser considerada uma mensagem inofensiva, pois não foram relatados ataques que utilizem essa mensagem até este momento, além de consumir baixíssimos níveis de uso de CPU para processá-la. A porcentagem de envio de cada uma dessas mensagens variou de 0 a 100% (por exemplo,

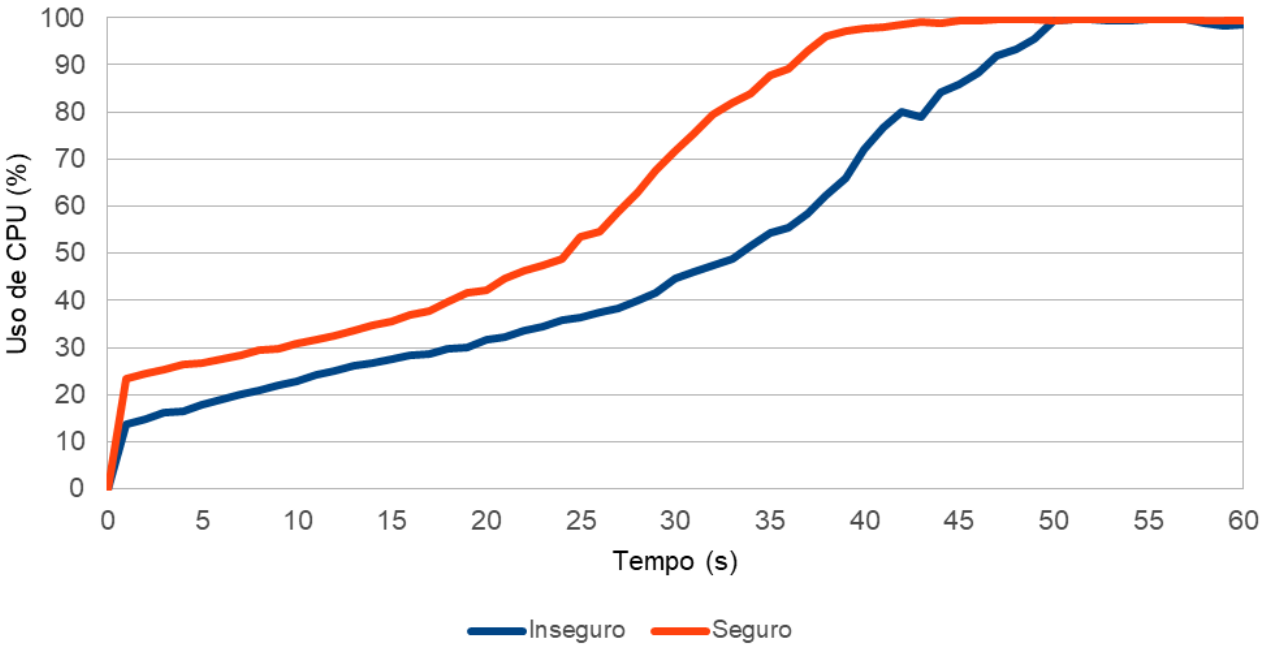

Figura 5.3 – Uso de CPU na comunicação de mensagens OFPT_FLOW_MOD através de canais seguro e inseguro.

0% de OFPT_FLOW_MOD e 100% de OFPT_PORT_STATUS, 1% de OFPT_FLOW_MOD e 99% de OFPT_PORT_STATUS, etc.). Nos dois primeiros ambientes (comunicação totalmente segura e comunicação totalmente insegura) foi monitorado o consumo de CPU pelo Open vSwitch. Já no terceiro ambiente (comunicação com ZONFlow) foi monitorado os consumos de CPU do Open vSwitch e do módulo do ZONFlow no *switch* somados.

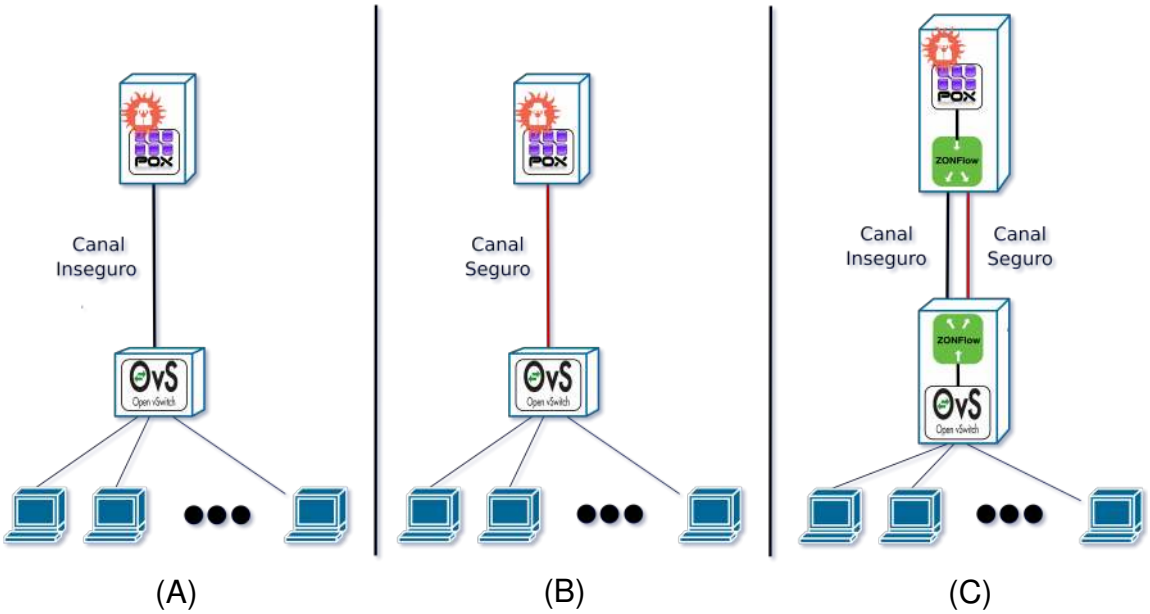

Figura 5.4 – Ambientes do terceiro experimento.

O terceiro experimento foi repetido nos três cenários de infraestrutura de hardware. Para tal, foi calculada a média de consumo de CPU, com um intervalo de confiança de 95%. A Figura 5.5 representa os resultados obtidos para o primeiro cenário (*switch* com 1GHz x 1 CPU, 2GB de RAM). A margem de erro foi de ± 0,5% de uso de CPU.

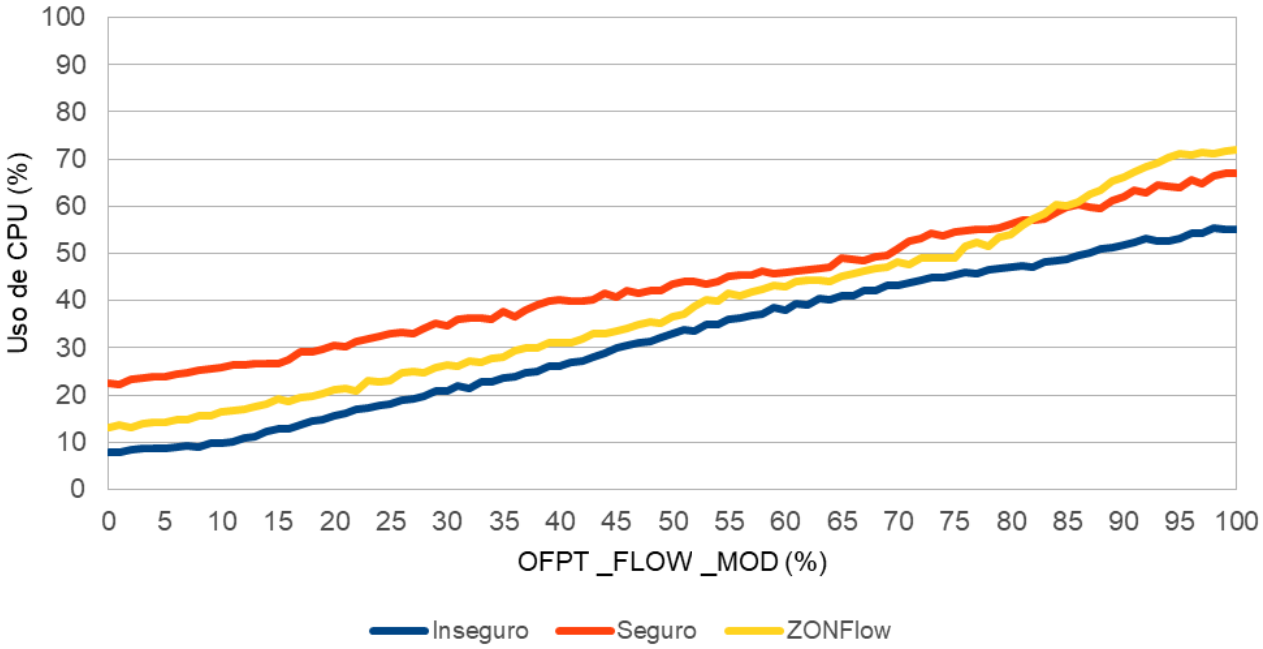

Figura 5.5 – *Overhead* imposto ao *switch* em virtude da comunicação de mensagens Openflow (Infraestrutura 1).

A Figura 5.6 apresenta os resultados obtidos no segundo cenário (*switch* com 625 MHz x 3 CPU, 512 MB de RAM). A margem de erro foi de ± 0,4% de uso de CPU.

A Figura 5.7 apresenta os resultados obtidos no terceiro cenário (*switch* com 1.5GHz x 2 CPU, 8 GB de RAM). A margem de erro foi de ± 0,1% de uso de CPU.

5.3.3 Discussão

Os resultados obtidos no primeiro experimento (Tabela 5.1 indicaram que, quando o Controlador e o *switch* estão diretamente conectados por uma conexão insegura, o ataque MitM pode ser realizado com sucesso, em virtude de todas as mensagens OFPT_FLOW _MOD são recebidas e processadas pelo *switch*, populando assim as tabelas de fluxos, e podendo concluir com sucesso um ataque de *Flow Table Overflow* [662]; entretanto, quando o ZONFlow foi inserido no contexto do experimento, o atacante pode somente realizar um ataque no canal inseguro, pois a conexão segura do ZONFlow é realizada com TLS e autenticação mútua, impedindo assim um ataque MitM. Quando o atacante injetou as mensagens OFPT_FLOW_MOD no canal inseguro do ambiente com ZONFlow, o módulo ZONFlow do

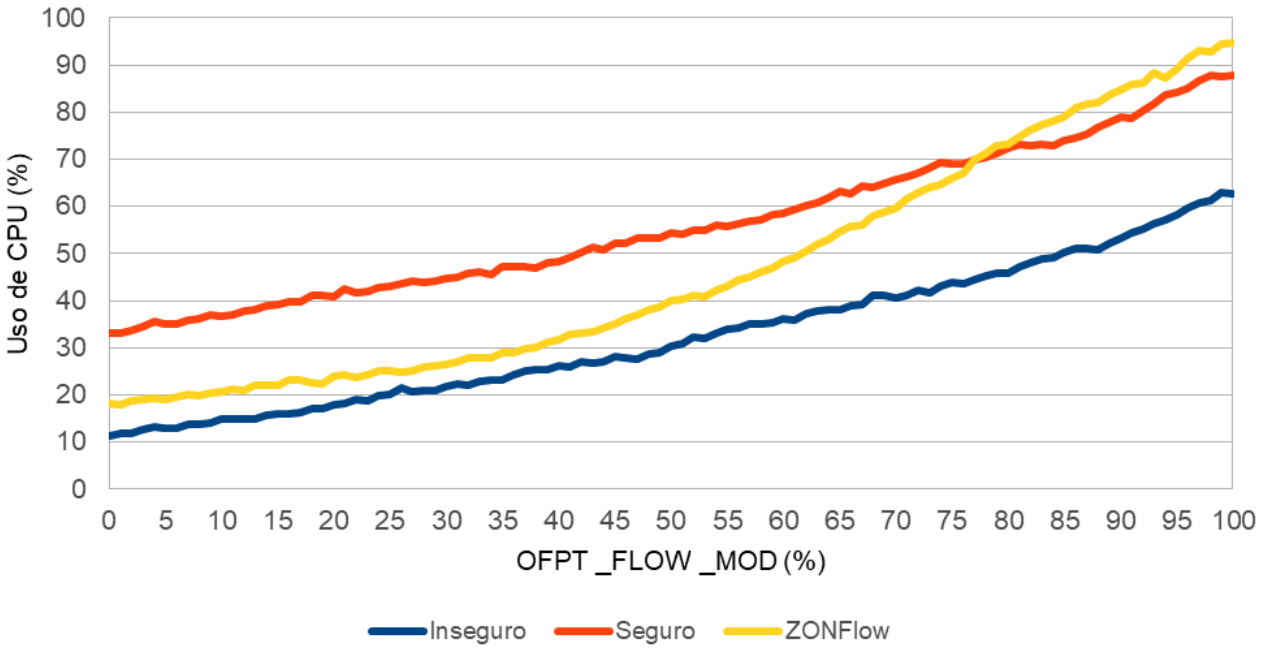

Figura 5.6 – *Overhead* imposto ao *switch* em virtude da comunicação de mensagens Openflow (Infraestrutura 2).

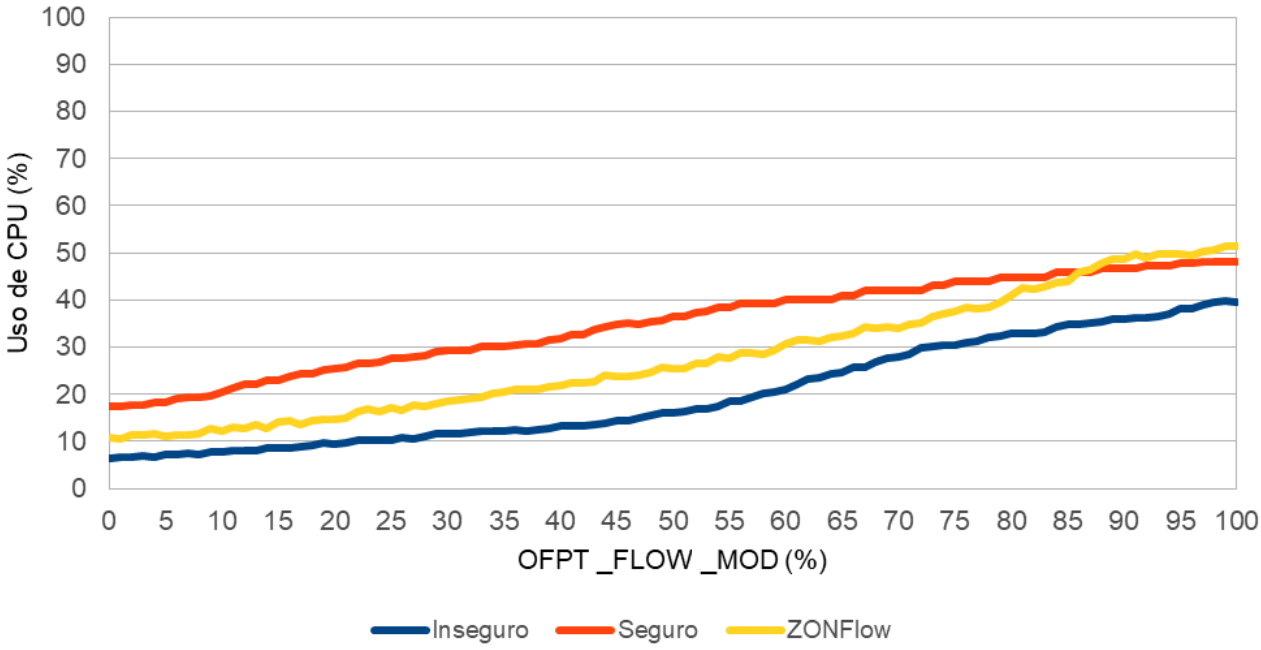

Figura 5.7 – *Overhead* imposto ao *switch* em virtude da comunicação de mensagens Openflow (Infraestrutura 3).

switch verificou que foram enviadas mensagens sensíveis pelo canal inseguro e, como o módulo possui uma regra de excluir mensagens sensíveis que cheguem pelo canal inseguro, as mensagens não chegaram até o Open vSwitch, e com isso, não foram inseridos novos fluxos, fazendo com que o ataque não tenha tido sucesso. Esse comportamento faz com que o objetivo de garantir um nível de segurança, para o atual contexto de possíveis

ataques através da API do *Southbound*, seja garantido, pois todas as mensagens sensíveis possuem a garantia que serão trafegadas exclusivamente por um canal seguro.

No caso do segundo experimento, o qual foi realizado visando verificar o impacto do canal seguro no desempenho do *switch*, em termos de consumo de CPU, comparado a um canal não seguro, os resultados obtidos (Figura 5.3) comprovam que o consumo de CPU por um canal seguro, inicialmente já apresenta um valor elevado, em relação ao canal não seguro ($\pm$ 23% do canal seguro contra $\pm$ 13% do canal não seguro). A medida que o alto tráfego de mensagens vai ocorrendo durante o tempo, o consumo de CPU para gerenciar o canal seguro vai sendo associado ao consumo de CPU necessário ao Open vSwitch para gerenciar as tabelas de fluxos, aumentando assim o consumo de CPU total. Isso faz com que o *switch*, gerenciando o Open vSwitch e o canal seguro, chegue próximo a 100% de uso de CPU mais rapidamente que quando o *switch* precisa gerenciar o Open vSwitch e o canal não seguro (em torno de 17s mais rápido). Esses resultados demonstram a necessidade de uma alternativa, como o ZONFlow, para aliar segurança e desempenho. Em termos dos experimentos seguintes, esses resultados demonstram também que, neste contexto, 1 minuto de tráfego é mais que suficiente para realizar um ataque de DoS. Em virtude disso, foi optado utilizar, nos experimentos seguintes, 1 minuto como o tempo de tráfego de mensagens.

O Controlador POX suporta, como canal seguro no plano de controle, somente conexões SSL. Sendo assim, para fins de comparação de resultados, o canal seguro do ZONFlow foi configurado para, durante a execução dos experimentos que objetivam coletar dados relacionados ao desempenho, não utilizar TLS, e sim SSL.

Para o terceiro experimento, foram escolhidas as três configurações de *switches* para representar um ambiente de rede de uso corporativo ou residencial (sem foco em alto desempenho, como *switch* de rack): um *switch* com uma especificação extremamente limitada de hardware (HP 2920-24G), para simular um ambiente com dificuldades críticas em prover uma estrutura segura e associada com desempenho, um *switch* intermediário (PICA8 P-3297 ou Edge-Core AS4600-54T) e um *switch* mais atualizado (Juniper QFX5100). Os resultados ilustrados nas Figuras 5.5, 5.6 e 5.7 demonstram o impacto no desempenho do *switch* ao gerenciar uma conexão segura, em relação a uma conexão não segura, à medida que o número de mensagens sensíveis vai aumentando. Embora, em alguns casos observados, essa diferença pareça pouca, como o dado apresentado representa a média de consumo de CPU durante 1 minuto, o custo da comunicação segura em um determinado instante será bem superior a essa diferença (vide resultados instantâneos do segundo experimento - Figura 5.3). Esse fenômeno ocorre porque, a medida que as memórias TCAM do *switch* são populadas, mais e mais processamento é necessário para gerenciar as mensagens OFPT_FLOW_MOD durante um curto espaço de tempo. Nos experimentos realizados, foram usadas máquinas virtuais para simular *switches* Openflow comerciais, e embora tenham sido configurados com a mesma frequência de CPU dos *switches* reais, o

processador de uso geral utilizado apresenta características normalmente superiores aos processadores embarcados nos *switches*.

Sendo assim, os impactos da comunicação segura em um ambiente real poderiam ser piores do que os identificados nos experimentos realizados, reforçando a necessidade de preocupação do consumo de CPU no *switch*, durante o gerenciamento de uma conexão segura. Quando a porcentagem de mensagens sensíveis é baixa, o ZONFlow apresenta um pouco mais de uso de CPU que uma comunicação não segura. Esse consumo extra de CPU existe devido ao custo de processamento do módulo ZONFlow. Uma possível solução para diminuir esse consumo pode ser obtida através da implementação dos algoritmos do ZONFlow nativa no software do Controlador e no software do *switch*. Porém, essa solução trará um alto acoplamento, deixando de lado a independência que a atual abordagem ZONFlow traz, em termos de escolha de softwares controladores e softwares de virtualização de *switches*.

A medida que o tráfego de mensagens OFPT_FLOW_MOD aumenta, a abordagem ZONFlow necessita gerenciar mais dados no canal seguro, aumentando assim o consumo de CPU. Nos três cenários de hardware onde os experimentos foram realizados, constatou-se que, em altos níveis de tráfego de mensagens sensíveis, a abordagem ZONFlow começa a consumir mais CPU que ao utilizar somente um canal seguro para todo o tráfego de mensagens. O motivo para esse fenômeno não decorre do algoritmo em si, mas da forma de implementação do protótipo, pois esse *overhead* pode ser definido em virtude do *switch* gerenciar duas aplicações concomitantemente, o Open vSwitch e o módulo ZONFlow. Caso os algoritmos ZONFlow sejam implementados de forma nativa aos softwares de gerenciamento do *switch* e Controlador, ou implementado diretamente nos hardwares dos dispositivos, como os algoritmos ZONFlow apresentam complexidade assintótica $O(1)$, o custo de processamento fica insignificante, fazendo com que o consumo de CPU total tenda a ficar menor ou igual ao consumo estritamente via canal seguro. Porém, essa escolha trará inevitavelmente impacto em uso de memória, em virtude de ser necessários dois *buffers* extras (armazenar os pacotes a serem enviados e os recebidos), durante o gerenciamento dos canais.

Entretanto, um tráfego tão alto de mensagens sensíveis durante um tempo considerável pode, provavelmente, ser um indício de um ataque em curso. Como a abordagem ZONFlow foca em proteger o canal de comunicação do plano de controle, embora já tenha sido constatado nos resultados do primeiro experimento, que o atacante não consegue desenvolver com sucesso algum ataque que dependa da injeção ou modificação de mensagens sensíveis no canal de comunicação, o atacante ainda pode realizar o ataque, como um DoS, partindo do software Controlador. Uma maneira de impedir a realização desse ataque poderia ser alcançada ao inserir, nos módulos ZONFlow, um IPS que considere alguns dados para decidir se um ataque está em curso, como por exemplo, o consumo de CPU e memória no *switch* e a quantidade de fluxos sendo gerenciadas pelo *switch*. Caso

esses parâmetros variem bruscamente, ou tenham um incremento gradual, as mensagens sensíveis poderão ser descartadas até que os parâmetros voltem a normalidade e, evitando assim, que o consumo de CPU do *switch* com ZONFlow seja próximo ao utilizar apenas um canal seguro. Essa alternativa de proteção poderá produzir falsos positivos, porém, para garantir que a rede não seja comprometida em termos de eficácia, esse IPS pode notificar o administrador da rede dos fatos detectados, possibilitando assim que os fatos sejam analisados e ações pontuais sejam tomadas.

Sendo assim, de acordo com os resultados dos experimentos, em situações normais de uso da rede a abordagem ZONFlow apresenta ganhos em termos de uso de CPU, comparado ao modelo de comunicação totalmente via canal seguro. Porém, em casos onde o volume de tráfego de mensagens sensíveis seja extremamente alto, como no caso de um DoS, a abordagem ZONFlow se mostrou mais custosa, em termos de uso de CPU, do que uma comunicação exclusivamente via canal seguro. Essa limitação pode ser atribuída ao consumo de CPU do interpretador Python, tanto para o seu funcionamento, quanto para gerenciar os canais de comunicação com alto tráfego. Uma solução pode ser provida com a implementação dos módulos ZONFlow nativamente, tanto no Open vSwitch, quanto no Controlador POX.

6. CONCLUSÃO

Desde o início do desenvolvimento das redes de computadores, foram surgindo vários novos protocolos, especificações, metodologias, técnicas, abordagens e tecnologias para a rede solucionar as crescentes demandas da sociedade. Porém, o aumento da complexidade dessas soluções e o crescimento acelerado de novas soluções fez com que as redes ficassem ossificadas, aumentando cada vez mais o desafio de propor soluções inovadoras, em virtude da necessidade de manter a compatibilidade com o que já foi posto em operação no passado, que impacta na resistência a uma disrupção.

Como uma alternativa ao status quo, foi proposto o paradigma SDN, que possibilita o desenvolvimento de soluções de forma mais desacopladas, através da separação dos planos da rede (gerenciamento, controle e dados), ou seja, a quebra da integração vertical e aplicando o princípio da separação de responsabilidades, fazendo com que algumas novas funcionalidades, como centralização lógica da rede e a programabilidade da rede, estejam disponíveis. Isso permite que o desenvolvimento de novas soluções e funcionalidades para a rede surjam sem comprometer o funcionamento da rede.

Porém, como toda inovação disruptiva, o paradigma SDN entrou na fase de adoção da curva de Gartner [306], atraindo a atenção da academia e da indústria. Como SDN é um paradigma que ainda não atingiu a maturidade, o foco da sociedade esteve durante muito tempo direcionado ao desenvolvimento das principais características do paradigma, de forma que possa ser aplicado de forma funcional em ambientes reais. Isso impactou que, requisitos como desempenho fossem priorizados em detrimento de outros, como segurança, por exemplo. A medida que o paradigma foi evoluindo, a segurança começou a ser um ponto crítico. Foram sendo descobertas várias vulnerabilidades, e consequentemente exploradas para prover diversos tipos de ataques, necessitando assim de novos mecanismos de defesa.

Neste trabalho, foram categorizadas as principais vulnerabilidades, tipos de ataque e mecanismos de defesa, envolvendo SDN, que foram descritas na literatura, como apresentado no Capítulo 3. Porém, todo esse processo (vulnerabilidade, ataque, defesa) impacta diretamente no desempenho. Um dos pontos identificados no Capítulo 3 foi a constatação dos problemas relacionados à falta de incentivos a uma comunicação segura no plano de controle da arquitetura SDN. A hipótese utilizada neste trabalho foi que isso ocorre em virtude do impacto que uma comunicação segura traz no desempenho. Para validar essa hipótese, foram realizados experimentos e constatado que o impacto, em termos de uso de CPU, nos *switches* é considerável, como apresentado no Capítulo 5.

Com o objetivo de fornecer a possibilidade de prover segurança, sem necessariamente comprometer o desempenho, foi proposta neste trabalho uma abordagem que faz o balanceamento entre segurança e desempenho, durante a comunicação do plano de

controle em SDN. Como apresentado no Capítulo 4, foi realizado um estudo de caso, envolvendo a especificação Openflow. As mensagens oriundas da especificação Openflow foram analisadas e categorizadas, de acordo com os ataques descritos na literatura, a fim de definir quais mensagens necessitam de um cuidado maior, durante o tráfego do plano de controle, em termos de segurança e quais mensagens podem ser consideradas atualmente como inofensivas, podendo o seu tráfego ser focado em desempenho. Para isso, foi proposto um algoritmo que considera dois canais de comunicação concomitantes, um seguro e outro não seguro, para o tráfego das mensagens do plano de controle. Esse algoritmo analisa se a mensagem faz parte do conjunto de mensagens sensíveis à segurança da rede e encaminha os pacotes pelo canal seguro. Caso a mensagem faz parte do conjunto de mensagens consideradas inofensivas, a encaminha pelo canal não seguro. O algoritmo foi validado, em termos de segurança e desempenho, através de provas formais e experimentações em ambientes simulados. Esses ambientes consideraram *switches* com diferentes níveis de recursos de hardware, a fim de verificar se a abordagem obtinha ganhos de desempenho, independente da infraestrutura física. Como constatado no Capítulo 5, a abordagem proposta consegue obter ganhos de desempenho, sem comprometer a segurança, comparado ao uso exclusivo de uma comunicação segura, durante a comunicação do plano de controle.

6.1 Validação da hipótese

O desenvolvimento deste estudo validou a hipótese que é possível balancear segurança e desempenho na comunicação do plano de controle de um ambiente SDN, entre Controladores e *switches*. A validação foi conduzida através do estudo do estado da arte em segurança em SDN, assim como através do desenvolvimento de uma abordagem que distribui o tráfego do plano de controle em dois canais de comunicação: um seguro e um não seguro. Essa distribuição de tráfego é baseada nas vulnerabilidades identificadas na literatura sobre as mensagens do protocolo do Southbound, de forma a direcionar as mensagens, consideradas sensíveis para a realização de um ataque, através do canal seguro e direcionar as mensagens consideradas, até hoje, inofensivas, para trafegarem pelo canal não seguro, e como isso atingir um melhor desempenho do trafegar todas as mensagens exclusivamente por um canal seguro.

Para isso, três questões de pesquisa foram respondidas, de forma a suportar a hipótese:

- Quais são as principais vulnerabilidades, ataques e soluções existentes para prover segurança em ambientes SDN? Foram descritas na literatura várias vulnerabilidades e ataques, desde o surgimento da SDN. De forma a combater essas adversidades, foram propostos vários mecanismos de defesa. Esse levantamento foi descrito no Ca-

pítulo 3, apresentando assim uma visão geral do que foi produzido pela comunidade até hoje, envolvendo segurança em SDN. O grande volume de trabalhos nessa área mostrou que segurança em SDN é um tema atual e que esse requisito é ainda extremamente relevante no contexto de redes. Ao focar no tema de segurança no canal de comunicação do plano de controle, este estudo mostrou que a falta de estímulos ao uso de uma comunicação segura, traz uma série de preocupações em termos de possíveis ameaças que podem prejudicar a rede como um todo.

- Quais são os impactos em termos de desempenho, ao prover segurança na comunicação entre controlador e *switch*, em um ambiente SDN? Além das preocupações, envolvendo esse tema, identificadas na literatura e apresentadas no Capítulo 4, foi realizado um estudo de caso com o protocolo Openflow e, através dos experimentos relatados no Capítulo 5, foi possível demonstrar que os *switches* apresentam uma queda de desempenho significativa, ao aderir a uma comunicação segura no plano de controle, comparado ao uso de uma comunicação não segura. Quanto maior o tráfego de mensagens sensíveis e menor a quantidade de recursos de hardware que o *switch* tem disponibilidade, maior é a queda de desempenho.

- Como prover uma solução para balancear desempenho e segurança, na comunicação entre controlador e *switch*, em um ambiente SDN? Uma vez compreendida a importância da segurança e do desempenho na comunicação do plano de controle de uma SDN, foi possível desenvolver uma abordagem que, utilizando a categorização das mensagens do protocolo do Southbound, em termos de oportunidades relatadas na literatura para realizar um ataque, realiza o tráfego dessas mensagens por um canal seguro, priorizando assim a segurança. Da mesma forma, foi desenvolvido o estudo de caso do protocolo Openflow, onde foi feita uma categorização das mensagens que atualmente não apresentam instrumentos para a efetivação de um ataque, consideradas assim inofensivas, de forma a trafegá-las por um canal não seguro, focando assim em melhorias em termos de desempenho, comparado ao uso exclusivo de um canal de comunicação seguro. Dessa forma, essa abordagem provê a comunicação segura do plano de controle de uma SDN, apresentando um desempenho da rede melhor do que se utilizada uma abordagem formada exclusivamente pelo uso de um canal seguro de comunicação, como demonstrado no Capítulo 4. Essa pergunta foi respondida através da aplicação desse balanceamento em uma série de experimentos, como apresentado no Capítulo 5, onde foram identificados melhorias no desempenho em todos os experimentos, quando o tráfego de mensagens sensíveis é moderado. Porém, os experimentos mostraram que, quando o tráfego de mensagens sensíveis é extremamente alto, depois de um determinado momento o atual protótipo da abordagem desenvolvida apresenta um desempenho pior do que o uso exclusivo de uma comunicação segura. Entretanto, a causa desse fenômeno pode ser solucionada ao implementar a abordagem nativamente, nos softwares dos Controladores e dos *swit-*

ches, ou no hardware dos *switches*. Porém, essa solução faz com que o Controlador e o *switch* adquiram um alto acoplamento, dificultando assim uma característica da SDN que é não dependência dos dispositivos da rede, em termos de fabricantes e modelos. Sendo assim, outra alternativa pode ser alcançada através do uso de um IPS, de forma a considerar o alto tráfego de mensagens sensíveis, durante um período de tempo considerável, como provavelmente sendo consequência da realização de um ataque. Ao identificar esse fenômeno, o IPS deve bloquear o tráfego dessas mensagens e comunicar o administrador da rede da ocorrência verificada de um possível ataque em curso, a fim de a situação ser melhor analisada e que sejam tomadas ações para impedir que o ataque volte a ocorrer. Com isso, a abordagem mantém um ganho de desempenho significativo, comparado ao uso exclusivo de um canal seguro de comunicação no plano de controle.

6.2 Contribuições da Tese

Este trabalho tem como contribuições uma abordagem para proporcionar o balanceamento entre segurança e desempenho na comunicação entre Controladores e *switches* de ambientes SDN; uma revisão da literatura sobre segurança em SDN; a análise do impacto, em termos de desempenho, de prover segurança na comunicação do plano de controle de ambientes SDN.

Além disso, o processo de desenvolvimento deste trabalho produziu artigos científicos. O trabalho *Security Analysis of Forwarding Strategies in Network Time Measurements Using Openflow* [428] foi apresentado um conjunto de experimentos para verificar o impacto de cada algoritmo de encaminhamento de pacotes na medição de *Round-Trip Time* (RTT) em redes Openflow. Os resultados obtidos indicaram que uma má escolha do algoritmo de encaminhamento de pacotes pode afetar significativamente a medição da rede. Além disso, essa decisão pode possibilitar que ataques de *delay* [299] possam ser mascarados, e a contribuição que SDN traz na melhoria de confiabilidade das medições de rede pode ficar comprometida. Os resultados também indicaram que ataques que confiam na medida de tempo da rede para poderem ser efetuados podem ser detectados e, em algumas circunstâncias, mitigados através da escolha adequada do algoritmo de encaminhamento.

Além disso, outro artigo produzido foi *An approach for detecting encrypted insider attacks on OpenFlow SDN Networks* [412], que descreveu uma abordagem para identificar ataques criptografados de *insiders* em redes Openflow, através de informações estatísticas requisitadas aos *switches* compatíveis com Openflow, e com isso prover um sistema de detecção de intrusão leve.

Por fim, foi publicado o trabalho *Lightweight IPS for Port Scan in Openflow SDN networks* [411], que descreveu uma solução não intrusiva para detectar e prevenir *port scan*

em ambientes Openflow. Os fluxos (*flows*) oriundos do *port scans* são detectados através do uso de estatísticas de fluxos coletadas dos *switches* e então as regras de roteamento dos fluxos são atualizadas. Sendo assim, toda vez que um atacante realiza um *port scan*, a fim de preparar um outro ataque, o sistema de prevenção de intrusão (IPS - *Intrusion Prevention System*) proposto bloqueia a origem dos pacotes, impedindo a realização de ataques subsequentes. Um possível ataque subsequente seria um ataque de negação de serviços (DoS - *Denial of Service*) de forma simples ou distribuída (DDoS- *Distributed Denial of Service*) usando pacotes criptografados. Essa solução foi considerada leve, pois consume poucos recursos, como largura de banda, uso de memória e processamento, para realizar a proteção do ambiente.

6.3 Trabalhos futuros

A segurança ainda é uma demanda crescente em todos os ambientes computacionais, pois cada vez mais as pessoas mal intencionadas utilizam de artifícios de computação para obterem vantagens ilícitas ou para causarem prejuízos a terceiros. Sendo assim, SDN não é diferente. Nesse sentido, este estudo pode identificar alguns pontos que podem ser explorados em trabalhos futuros. Um ponto é a exploração de vulnerabilidades que ainda não foram identificadas nas mensagens da especificação Openflow. Esse assunto tem sido pouco explorado pela comunidade científica, embora possa trazer grandes impactos na sociedade, devido ao crescente interesse da indústria em desenvolver equipamentos compatíveis com SDN.

Com a evolução da Internet das coisas (IoT - Internet of Things), cada vez mais dispositivos serão incluídos nas redes, de forma a aumentar o tráfego significativamente, tanto no plano de dados quanto no plano de controle das SDNs. Poucos são os trabalhos que exploram desempenho de SDN, e como esses dispositivos apresentam severas vulnerabilidades, a relação entre segurança e desempenho pode ser uma oportunidade de trabalhos futuros importantes.

Embora Openflow seja a tecnologia *de facto* de implementação do *Southbound* da arquitetura SDN, existem outras propostas de implementação (como apresentado no Capítulo 2), de forma que é importante desenvolver uma análise detalhada das vulnerabilidades existentes nessas tecnologias, assim como a relação delas com o desempenho da rede.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Abaid, Z.; Rezvani, M.; Jha, S. “Malwaremonitor: An SDN-based framework for securing large networks”. In: CoNEXT on Student Workshop, 2014, pp. 40–42.

[2] Abbasi, A. G.; Khan, Z. “Veidblock: Verifiable identity using blockchain and ledger in a software defined network”. In: 10th International Conference on Utility and Cloud Computing, 2017, pp. 173–179.

[3] Abdulhassan, A.; Ahmadi, M. “Parallel many fields packet classification technique using r-tree”. In: Annual Conference on New Trends in Information Communications Technology Applications (NTICT), 2017, pp. 274–279.

[4] Abdullaziz, O. I.; Chen, Y. J.; Wang, L. C. “Lightweight authentication mechanism for software defined network using information hiding”. In: IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6.

[5] Abdulqadder, I. H.; Zou, D.; Aziz, I. T.; Yuan, B. “Modeling software defined security using multi-level security mechanism for SDN environment”. In: IEEE 17th International Conference on Communication Technology (ICCT), 2017, pp. 1342–1346.

[6] Abdulqadder, I. H.; Zou, D.; Aziz, I. T.; Yuan, B.; Li, W. “SecSDN-cloud: Defeating vulnerable attacks through secure software-defined networks”, *IEEE Access*, vol. 6, 2018, pp. 8292–8301.

[7] Abubakar, A.; Pranggono, B. “Machine learning based intrusion detection system for software defined networks”. In: 7th International Conference on Emerging Security Technologies (EST), 2017, pp. 138–143.

[8] Achleitner, S.; La Porta, T.; Jaeger, T.; McDaniel, P. “Adversarial network forensics in software defined networking”. In: Symposium on SDN Research, 2017, pp. 8–20.

[9] Adam, I.; Ahola, T.; Sailio, M.; Vallivaara, V.; von Eye, F. “Adaptive monitoring and management of security events with SDN”. In: NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 817–820.

[10] Agborubere, B.; Sanchez-Velazquez, E. “Openflow communications and tls security in software-defined networks”. In: IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017, pp. 560–566.

[11] Aguado, A.; Lopez, V.; Martinez-Mateo, J.; Szyrkowiec, T.; Autenrieth, A.; Peev, M.; Lopez, D.; Martin, V. “Hybrid conventional and quantum security for software defined and virtualized networks”, *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9–10, Oct 2017, pp. 819–825.

[12] Ahmad, I.; Liyanage, M.; Ylianttila, M.; Gurtov, A. “Analysis of deployment challenges of host identity protocol”. In: European Conference on Networks and Communications (EuCNC), 2017, pp. 1–6.

[13] Ahmad, I.; Namal, S.; Ylianttila, M.; Gurtov, A. “Security in software defined networks: A survey”, *IEEE Communications Surveys Tutorials*, vol. 17–4, Fourthquarter 2015, pp. 2317–2346.

[14] Ahmed, M. E.; Kim, H. “Ddos attack mitigation in internet of things using software defined networking”. In: IEEE 3rd International Conference on Big Data Computing Service and Applications (BigDataService), 2017, pp. 271–276.

[15] Ahmed, M. E.; Kim, H.; Park, M. “Mitigating dns query-based ddos attacks with machine learning on software-defined networking”. In: IEEE Military Communications Conference (MILCOM), 2017, pp. 11–16.

[16] Aizuddin, A. A.; Atan, M.; Norulazmi, M.; Noor, M. M.; Akimi, S.; Abidin, Z. “Dns amplification attack detection and mitigation via sflow with security-centric SDN”. In: 11th International Conference on Ubiquitous Information Management and Communication, 2017, pp. 3:1–3:7.

[17] Akhunzada, A.; Ahmed, E.; Gani, A.; Khan, M. K.; Imran, M.; Guizani, S. “Securing software defined networks: taxonomy, requirements, and open issues”, *IEEE Communications Magazine*, vol. 53–4, April 2015, pp. 36–44.

[18] Akhunzada, A.; Khan, M. K. “Toward secure software defined vehicular networks: Taxonomy, requirements, and open issues”, *IEEE Communications Magazine*, vol. 55–7, 2017, pp. 110–118.

[19] Al Fardan, N.; Paterson, K. “Lucky thirteen: Breaking the tls and dtls record protocols”. In: IEEE Symposium on Security and Privacy (SP), 2013, pp. 526–540.

[20] Al-Haj, S.; Tolone, W. J. “Flowtable pipeline misconfigurations in software defined networks”. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 247–252.

[21] Al-Shaer, E.; Al-Haj, S. “Flowchecker: Configuration analysis and verification of federated openflow infrastructures”. In: 3rd ACM Workshop on Assurable and Usable Security Configuration, 2010, pp. 37–44.

[22] Al-Zewairi, M.; Suleiman, D.; Almajali, S. “An experimental software defined security controller for software defined network”. In: 4th International Conference on Software Defined Systems (SDS), 2017, pp. 32–36.

[23] Alasadi, E.; Al-Raweshidy, H. S. “Ssed: Servers under software-defined network architectures to eliminate discovery messages”, *IEEE/ACM Transactions on Networking*, vol. 26–1, Feb 2018, pp. 104–117.

[24] Alcorn, J.; Chow, C. “A framework for large-scale modeling and simulation of attacks on an openflow network”. In: 23rd International Conference on Computer Communication and Networks (ICCCN), 2014, pp. 1–6.

[25] Aleroud, A.; Alsmadi, I. “Identifying dos attacks on software defined networks: A relation context approach”. In: NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 853–857.

[26] Alharbi, T.; Durando, D.; Pakzad, F.; Portmann, M. “Securing arp in software defined networks”. In: IEEE 41st Conference on Local Computer Networks (LCN), 2016, pp. 523–526.

[27] Ali, S.; Wu, K.; Khan, H. “Traffic anomaly detection in the presence of p2p traffic”. In: 39th Conference on Local Computer Networks (LCN), 2014, pp. 482–485.

[28] Ali, S. T.; Sivaraman, V.; Radford, A.; Jha, S. “A survey of securing networks using software defined networking”, *IEEE Transactions on Reliability*, vol. 64–3, Sept 2015, pp. 1086–1097.

[29] Aliyu, A. L.; Bull, P.; Abdallah, A. “Performance implication and analysis of the Openflow SDN protocol”. In: 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2017, pp. 391–396.

[30] Aliyu, A. L.; Bull, P.; Abdallah, A. “A trust management framework for network applications within an SDN environment”. In: 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2017, pp. 93–98.

[31] Allen, C.; Dierks, T. “The TLS protocol version 1.0”, 1999. Disponível em http://tools.ietf.org/html/rfc2246. Acesso em 01/06/2018.

[32] Allouzi, M.; Khan, J. “Safeflow: Authentication protocol for software defined networks”. In: IEEE 12th International Conference on Semantic Computing (ICSC), 2018, pp. 374–376.

[33] Almutairi, L. M.; Shetty, S. “Generalized stochastic petri net model based security risk assessment of software defined networks”. In: IEEE Military Communications Conference (MILCOM), 2017, pp. 545–550.

[34] Alparslan, O.; Gunes, O.; Hanay, Y. S.; Arakawa, S.; Murata, M. “Improving resiliency against ddos attacks by SDN and multipath orchestration of vnf services”. In: IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), 2017, pp. 1–3.

[35] Alsaleh, M. N.; Al-Shaer, E. “Towards automated verification of active cyber defense strategies on software defined networks”. In: ACM Workshop on Automated Decision Making for Active Cyber Defense, 2016, pp. 23–29.

[36] Alshamrani, A.; Chowdhary, A.; Pisharody, S.; Lu, D.; Huang, D. “A defense system for defeating ddos attacks in SDN based networks”. In: 15th ACM International Symposium on Mobility Management and Wireless Access, 2017, pp. 83–92.

[37] Alsmadi, I.; Xu, D. “Security of software defined networks: A survey”, *Computers & Security*, vol. 53, 2015, pp. 79 – 108.

[38] Alwabel, A.; Yu, M.; Zhang, Y.; Mirkovic, J. “Senss: Observe and control your own traffic in the internet”. In: ACM Conference on SIGCOMM, 2014, pp. 349–350.

[39] Amann, J.; Sommer, R. “Providing dynamic control to passive network security monitoring”. In: Research in Attacks, Intrusions, and Defenses, 2015, pp. 133–152.

[40] Ambrosin, M.; Conti, M.; De Gaspari, F.; Poovendran, R. “Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks”. In: 10th ACM Symposium on Information, Computer and Communications Security, 2015, pp. 639–644.

[41] Ambrosin, M.; Conti, M.; Gaspari, F. D.; Devarajan, N. “Amplified distributed denial of service attack in software defined networking”. In: 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2016, pp. 1–4.

[42] Ambrosin, M.; Conti, M.; Gaspari, F. D.; Poovendran, R. “Lineswitch: Tackling control plane saturation attacks in software-defined networking”, *IEEE/ACM Transactions on Networking*, vol. 25–2, April 2017, pp. 1206–1219.

[43] Amin, R.; Shah, N.; Shah, B.; Alfandi, O. “Auto-configuration of acl policy in case of topology change in hybrid SDN”, *IEEE Access*, vol. 4, 2016, pp. 9437–9450.

[44] Ammar, M.; Rizk, M.; Abdel-Hamid, A.; Aboul-Seoud, A. K. “A framework for security enhancement in SDN-based datacenters”. In: 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2016, pp. 1–4.

[45] Anderson, R.; Hall, C. “Collaborating with the enemy on network management”. In: *Security Protocols XXII*, Springer International Publishing, 2014, *Lecture Notes in Computer Science*, vol. 8809, pp. 163–171.

[46] Aneci, M. N.; Gheorghe, L.; Carabas, M.; Soriga, S.; Somesan, R. A. “SDN-based security mechanism”. In: 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER), 2015, pp. 12–17.

[47] Antonioli, D.; Tippenhauer, N. O. “Minicps: A toolkit for security research on cps networks”. In: 1st ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy, 2015, pp. 91–100.

[48] Arbettu, R. K.; Khondoker, R.; Bayarou, K.; Weber, F. “Security analysis of opendaylight, onos, rosemary and ryu SDN controllers”. In: 17th International Telecommunications Network Strategy and Planning Symposium (Networks), 2016, pp. 37–44.

[49] Arins, A. “Firewall as a service in SDN openflow network”. In: IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), 2015, pp. 1–5.

[50] Armando, A.; Ranise, S.; Traverso, R.; Wrona, K. “Compiling nato authorization policies for enforcement in the cloud and SDNs”. In: IEEE Conference on Communications and Network Security (CNS), 2015, pp. 741–742.

[51] Aschoff, R.; Rosendo, D.; Machado, M.; Santos, A.; Sadok, D. “A network access control solution combining orbac and SDN”. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 483–489.

[52] Aseeri, A.; Netjinda, N.; Hewett, R. “Alleviating eavesdropping attacks in software-defined networking data plane”. In: 12th Annual Conference on Cyber and Information Security Research, 2017, pp. 1–8.

[53] Ashraf, J.; Latif, S. “Handling intrusion and ddos attacks in software defined networks using machine learning techniques”. In: Software Engineering Conference (NSEC), National, 2014, pp. 55–60.

[54] Aslan, M.; Matrawy, A. “Could network view inconsistency affect virtualized network security functions?” In: IEEE Conference on Communications and Network Security (CNS), 2017, pp. 510–512.

[55] Assis, M. V. O. D.; Hamamoto, A. H.; Abrão, T.; Proença, M. L. “A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for dos/ddos mitigation on SDN networks”, *IEEE Access*, vol. 5, 2017, pp. 9485–9496.

[56] Aydeger, A.; Saputro, N.; Akkaya, K.; Rahman, M. “Mitigating crossfire attacks using SDN-based moving target defense”. In: IEEE 41st Conference on Local Computer Networks (LCN), 2016, pp. 627–630.

[57] Azab, M.; Fortes, J. A. B. “Towards proactive SDN-controller attack and failure resilience”. In: International Conference on Computing, Networking and Communications (ICNC), 2017, pp. 442–448.

[58] Azzouni, A.; Braham, O.; Nguyen, T. M. T.; Pujolle, G.; Boutaba, R. “Fingerprinting openflow controllers: The first step to attack an SDN control plane”. In: IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6.

[59] Bailey, J.; Budgen, D.; Turner, M.; Kitchenham, B.; Brereton, P.; Linkman, S. “Evidence relating to object-oriented software design: A survey”. In: 1st International Symposium on Empirical Software Engineering and Measurement, 2007, pp. 482–484.

[60] Bakker, J. N.; Welch, I.; Seah, W. K. G. “Network-wide virtual firewall using SDN/openflow”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016, pp. 62–68.

[61] Balagopal, D.; Rani, X. A. K. “Netwatch: Empowering software-defined network switches for packet filtering”. In: International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2015, pp. 837–840.

[62] Bannour, F.; Souihi, S.; Mellouk, A. “Distributed SDN control: Survey, taxonomy, and challenges”, *IEEE Communications Surveys Tutorials*, vol. 20–1, Firstquarter 2018, pp. 333–354.

[63] Banse, C.; Rangarajan, S. “A secure northbound interface for SDN applications”. In: IEEE Trustcom/BigDataSE/ISPA, 2015, pp. 834–839.

[64] Banse, C.; Schuette, J. “A taxonomy-based approach for security in software-defined networking”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–6.

[65] Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Warfield, A. “Xen and the art of virtualization”, *SIGOPS Oper. Syst. Rev.*, vol. 37–5, Oct 2003, pp. 164–177.

[66] Barki, L.; Shidling, A.; Meti, N.; Narayan, D. G.; Mulla, M. M. “Detection of distributed denial of service attacks in software defined networks”. In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016, pp. 2576–2581.

[67] Basnet, S. R.; Shakya, S. “Bss: Blockchain security over software defined network”. In: International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 720–725.

[68] Beigi-Mohammadi, N.; Barna, C.; Shtern, M.; Khazaei, H.; Litoiu, M. “Caamp: Completely automated ddos attack mitigation platform in hybrid clouds”. In: 12th International Conference on Network and Service Management (CNSM), 2016, pp. 136–143.

[69] Belyaev, M.; Gaivoronski, S. “Towards load balancing in SDN-networks during ddos-attacks”. In: International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014, pp. 1–6.

[70] Ben-Itzhak, Y.; Barabash, K.; Cohen, R.; Levin, A.; Raichstein, E. “EnforSDN: Network policies enforcement with SDN”. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 80–88.

[71] Benabbou, J.; Elbaamrani, K.; Idboufker, N.; Elassali, R. “Software-defined networks, security aspects analysis”. In: 11th International Conference on Information Assurance and Security (IAS), 2015, pp. 79–84.

[72] Benson, T.; Akella, A.; Maltz, D. “Unraveling the complexity of network management”. In: 6th USENIX Symposium on Networked Systems Design and Implementation, 2009, pp. 335–348.

[73] Bera, P.; Saha, A.; Setua, S. K. “Denial of service attack in software defined network”. In: 5th International Conference on Computer Science and Network Technology (ICCSNT), 2016, pp. 497–501.

[74] Bernardo, D. V.; Chua, B. B. “Introduction and analysis of SDN and nfv security architecture (sn-seca)”. In: IEEE 29th International Conference on Advanced Information Networking and Applications, 2015, pp. 796–801.

[75] Betge-Brezetz, S.; Kamga, G.-B.; El Amrani Joutei, A.; Maalmi, O. “Control of sensitive traffic in the cloud based on openflow”. In: IEEE 3rd International Conference on Cloud Networking (CloudNet), 2014, pp. 266–268.

[76] Betgé-Brezetz, S.; Kamga, G. B.; Balla, M. N.; Criton, T.; Jebalia, H. “SDN-based trusted path in a multi-domain network”. In: IEEE International Conference on Cloud Engineering Workshop (IC2EW), 2016, pp. 19–24.

[77] Bhunia, S. S.; Gurusamy, M. “Dynamic attack detection and mitigation in iot using SDN”. In: 27th International Telecommunication Networks and Applications Conference (ITNAC), 2017, pp. 1–6.

[78] Bianco, A.; Birke, R.; Giraudo, L.; Palacin, M. “Openflow switching: Data plane performance”. In: IEEE International Conference on Communications, 2010, pp. 1–5.

[79] Bilal, T.; Faiz, Z.; Shah, M. A. “Software defined networks: An analysis on robust security practices”. In: 23rd International Conference on Automation and Computing (ICAC), 2017, pp. 1–5.

[80] Boero, L.; Marchese, M.; Zappatore, S. “Support vector machine meets software defined networking in ids domain”. In: 29th International Teletraffic Congress (ITC 29), 2017, pp. 25–30.

[81] Boite, J.; Nardin, P. A.; Rebecchi, F.; Bouet, M.; Conan, V. “Statesec: Stateful monitoring for ddos protection in software defined networks”. In: IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–9.

[82] Bosshart, P.; Daly, D.; Gibb, G.; Izzard, M.; McKeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; Walker, D. “P4: Programming protocol-independent packet processors”, *SIGCOMM Comput. Commun. Rev.*, vol. 44–3, Jul 2014, pp. 87–95.

[83] Bouacida, N.; Alghadhban, A.; Alalmaei, S.; Mohammed, H.; Shihada, B. “Failure mitigation in software defined networking employing load type prediction”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–7.

[84] Bouet, M.; Leguay, J.; Conan, V. “Cost-based placement of virtualized deep packet inspection functions in SDN”. In: Military Communications Conference, 2013, pp. 992–997.

[85] Braga, R.; Mota, E.; Passito, A. “Lightweight ddos flooding attack detection using nox/openflow”. In: IEEE 35th Conference on Local Computer Networks (LCN), 2010, pp. 408–415.

[86] Bremler-Barr, A.; Harchol, Y.; Hay, D.; Koral, Y. “Deep packet inspection as a service”. In: 10th ACM International on Conference on Emerging Networking Experiments and Technologies, 2014, pp. 271–282.

[87] Brooks, M.; Yang, B. “A man-in-the-middle attack against opendaylight SDN controller”. In: 4th Annual ACM Conference on Research in Information Technology, 2015, pp. 45–49.

[88] Bull, P.; Austin, R.; Popov, E.; Sharma, M.; Watson, R. “Flow based security for IoT devices using an SDN gateway”. In: IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), 2016, pp. 157–163.

[89] Buragohain, C.; Medhi, N. “Flowtrapp: An SDN based architecture for ddos attack detection and mitigation in data centers”. In: 3rd International Conference on Signal Processing and Integrated Networks (SPIN), 2016, pp. 519–524.

[90] Carvalho, L. F.; Fernandes, G.; Rodrigues, J. J. P. C.; Mendes, L. S.; Proença, M. L. “A novel anomaly detection system to assist network management in SDN environment”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–6.

[91] Ceron, J. M.; Margi, C. B.; Granville, L. Z. “MARS: An SDN-based malware analysis solution”. In: IEEE Symposium on Computers and Communication (ISCC), 2016, pp. 525–530.

[92] Chandrasekaran, B.; Tschaen, B.; Benson, T. “Isolating and tolerating SDN application failures with legoSDN”. In: Symposium on SDN Research, 2016, pp. 1–12.

[93] Chang, S. Y.; Park, Y.; Muralidharan, A. “Fast address hopping at the switches: Securing access for packet forwarding in SDN”. In: NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 454–460.

[94] Chang, Y.; Lin, T. “Cloud-clustered firewall with distributed SDN devices”. In: IEEE Wireless Communications and Networking Conference (WCNC), 2018, pp. 1–5.

[95] Chao, T. W.; Ke, Y. M.; Chen, B. H.; Chen, J. L.; Hsieh, C. J.; Lee, S. C.; Hsiao, H. C. “Securing data planes in software-defined networks”. In: IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 465–470.

[96] Charmet, F.; Waldinger, R.; Blanc, G.; Kiennert, C.; Toumi, K. “Preserving confidentiality during the migration of virtual SDN topologies: A formal approach”. In: IEEE 16th International Symposium on Network Computing and Applications (NCA), 2017, pp. 1–5.

[97] Chen, C. C.; Chen, Y. R.; Lu, W. C.; Tsai, S. C.; Yang, M. C. “Detecting amplification attacks with software defined networking”. In: IEEE Conference on Dependable and Secure Computing, 2017, pp. 195–201.

[98] Chen, G.; Hu, G.; Jiang, Y.; Zhang, C. “Savsh: Ip source address validation for SDN hybrid networks”. In: IEEE Symposium on Computers and Communication (ISCC), 2016, pp. 409–414.

[99] Chen, M.-H.; Ciou, J.-Y.; Chung, I.-H.; Chou, C.-F. “Flexprotect: A SDN-based ddos attack protection architecture for multi-tenant data centers”. In: International Conference on High Performance Computing in Asia-Pacific Region, 2018, pp. 202–209.

[100] Chen, P. J.; Chen, Y. W. “Implementation of SDN based network intrusion detection and prevention system”. In: International Carnahan Conference on Security Technology (ICCST), 2015, pp. 141–146.

[101] Chen, Q.; Qian, C.; Zhong, S. “Privacy-preserving cross-domain routing optimization - a cryptographic approach”. In: IEEE 23rd International Conference on Network Protocols (ICNP), 2015, pp. 356–365.

[102] Chen, Z.; Jiang, F.; Cheng, Y.; Gu, X.; Liu, W.; Peng, J. “Xgboost classifier for ddos attack detection and analysis in SDN-based cloud”. In: IEEE International Conference on Big Data and Smart Computing (BigComp), 2018, pp. 251–256.

[103] Cheng, F.; Qiu, X. “Network anomaly detection based on frequent sub-graph mining approach and association analysis”. In: IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), 2016, pp. 12–16.

[104] Chi, P.-W.; Kuo, C.-T.; Guo, J.-W.; Lei, C.-L. “How to detect a compromised SDN switch”. In: 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–6.

[105] Chi, P.-W.; Kuo, C.-T.; Ruan, H.-M.; Chen, S.-J.; Lei, C.-L. “An ami threat detection mechanism based on SDN networks”. In: 8th International Conference on Emerging Security Information, Systems and Technologies, 2014, pp. 208–211.

[106] Chi, Y.; Jiang, T.; Li, X.; Gao, C. “Design and implementation of cloud platform intrusion prevention system based on SDN”. In: IEEE 2nd International Conference on Big Data Analysis (ICBDA), 2017, pp. 847–852.

[107] Chin, T.; Mountrouidou, X.; Li, X.; Xiong, K. “An SDN-supported collaborative approach for ddos flooding detection and containment”. In: IEEE Military Communications Conference, 2015, pp. 659–664.

[108] Chin, T.; Mountrouidou, X.; Li, X.; Xiong, K. “Selective packet inspection to detect dos flooding using software defined networking (SDN)”. In: IEEE 35th International Conference on Distributed Computing Systems Workshops, 2015, pp. 95–99.

[109] Chippalkatti, O.; Nimbhorkar, S. U. “An approach for detection of attacks in software defined networks”. In: International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017, pp. 1–3.

[110] Chiu, Y. C.; Lin, P. C. “Rapid detection of disobedient forwarding on compromised openflow switches”. In: International Conference on Computing, Networking and Communications (ICNC), 2017, pp. 672–677.

[111] Choi, T.; Kang, S.; Yoon, S.; Yang, S.; Song, S.; Park, H. “Suvmf: Software-defined unified virtual monitoring function for SDN-based large-scale networks”. In: 9th International Conference on Future Internet Technologies, 2014, pp. 1–6.

[112] Choi, T.; Lee, B.; Kang, S.; Song, S.; Park, H.; Yoon, S.; Yang, S. “Iris-coman: Scalable and reliable control and management architecture for SDN-enabled large-scale networks”, *Journal of Network and Systems Management*, vol. 23–2, Apr 2015, pp. 252–279.

[113] Chou, L. D.; Tseng, C. W.; Huang, Y. K.; Chen, K. C.; Ou, T. F.; Yen, C. K. “A security service on-demand architecture in SDN”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2016, pp. 287–291.

[114] Chowdhary, A.; Alshamrani, A.; Huang, D.; Liang, H. “Mtd analysis and evaluation framework in software defined network (mason)”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2018, pp. 43–48.

[115] Chowdhary, A.; Pisharody, S.; Alshamrani, A.; Huang, D. “Dynamic game based security framework in SDN-enabled cloud networking environments”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2017, pp. 53–58.

[116] Chowdhary, A.; Pisharody, S.; Huang, D. “SDN based scalable mtd solution in cloud network”. In: ACM Workshop on Moving Target Defense, 2016, pp. 27–36.

[117] Chu, Y.-H.; Chen, Y.-T.; Chou, Y.-C.; Tseng, M.-C. “A simplified cloud computing network architecture using future internet technologies”. In: Network Operations and Management Symposium (APNOMS), 13th Asia-Pacific, 2011, pp. 1–4.

[118] Chung, C.-J.; Khatkar, P.; Xing, T.; Lee, J.; Huang, D. “Nice: Network intrusion detection and countermeasure selection in virtual network systems”, *IEEE Trans. Dependable Secur. Comput.*, vol. 10–4, Jul 2013, pp. 198–211.

[119] Cisco. “Cisco application centric infrastructure”, 2016. Disponível em http://www.cisco.com/c/en/us/solutions/data-center-virtualization/application-centric-infrastructure/index.html. Acesso em 01/06/2018.

[120] Cisco. “Cisco open SDN controller”, 2016. Disponível em http://www.cisco.com/c/enlus/products/cloud-systems-management/open-{SDN}-controller/index.html. Acesso em 01/06/2018.

[121] Collings, J.; Liu, J. “An openflow-based prototype of SDN-oriented stateful hardware firewalls”. In: IEEE 22nd International Conference on Network Protocols (ICNP), 2014, pp. 525–528.

[122] Conti, M.; Gangwal, A. “Blocking intrusions at border using software defined-internet exchange point (sd-ixp)”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 1–6.

[123] Conti, M.; Gangwal, A.; Gaur, M. S. “A comprehensive and effective mechanism for ddos detection in SDN”. In: IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2017, pp. 1–8.

[124] Conti, M.; Gaspari, F. D.; Mancini, L. V. “A novel stealthy attack to gather SDN configuration-information”, *IEEE Transactions on Emerging Topics in Computing*, 2018, pp. 1–12.

[125] Cox, J. H.; Clark, R.; Owen, H. “Leveraging SDN and webrtc for rogue access point security”, *IEEE Transactions on Network and Service Management*, vol. 14–3, Sept 2017, pp. 756–770.

[126] Cox, J. H.; Clark, R. J.; Owen, H. L. “Leveraging SDN for arp security”. In: SoutheastCon 2016, 2016, pp. 1–8.

[127] Cox, J. H.; Clark, R. J.; Owen, H. L. “Security policy transition framework for software defined networks”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016, pp. 56–61.

[128] Cox, Jr., J. H.; Clark, R. J.; Owen, III, H. L. “Leveraging SDN to improve the security of dhcp”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 35–38.

[129] Cui, H.; Chen, Z.; Yu, L.; Xie, K.; Xia, Z. “Authentication mechanism for network applications in SDN environments”. In: 20th International Symposium on Wireless Personal Multimedia Communications (WPMC), 2017, pp. 1–5.

[130] Cui, H.; Karame, G. O.; Klaedtke, F.; Bifulco, R. “On the fingerprinting of software-defined networks”, *IEEE Transactions on Information Forensics and Security*, vol. 11–10, 2016, pp. 2160–2173.

[131] da Silva, A. S.; Wickboldt, J. A.; Granville, L. Z.; Schaeffer-Filho, A. “Atlantic: A framework for anomaly traffic detection, classification, and mitigation in SDN”. In: NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 27–35.

[132] Dao, N.-N.; Park, J.; Park, M.; Cho, S. “A feasible method to combat against ddos attack in SDN network”. In: International Conference on Information Networking (ICOIN), 2015, pp. 309–311.

[133] Dargahi, T.; Caponi, A.; Ambrosin, M.; Bianchi, G.; Conti, M. “A survey on the security of stateful SDN data planes”, *IEEE Communications Surveys Tutorials*, vol. 19–3, thirdquarter 2017, pp. 1701–1725.

[134] Dauer, P.; Khondoker, R.; Marx, R.; Bayarou, K. “Security analysis of software defined networking applications for monitoring and measurement: Sflow and bigtap”. In: The 10th International Conference on Future Internet, 2015, pp. 51–56.

[135] Dayal, N.; Srivastava, S. “Analyzing behavior of ddos attacks to identify ddos detection features in SDN”. In: 9th International Conference on Communication Systems and Networks (COMSNETS), 2017, pp. 274–281.

[136] de la Hoz, E.; Cochrane, G.; Moreira-Lemus, J. M.; Paez-Reyes, R.; Marsa-Maestre, I.; Alarcos, B. “Detecting and defeating advanced man-in-the-middle attacks against tls”. In: 6th International Conference On Cyber Conflict (CyCon 2014), 2014, pp. 209–221.

[137] Deng, S.; Gao, X.; Lu, Z.; Gao, X. “Packet injection attack and its defense in software-defined networks”, *IEEE Transactions on Information Forensics and Security*, vol. 13–3, March 2018, pp. 695–705.

[138] Deng, S.; Gao, X.; Lu, Z.; Gao, X. “Packet injection attack and its defense in software-defined networks”, *IEEE Transactions on Information Forensics and Security*, vol. 13–3, 2018, pp. 695–705.

[139] Dharma, N. I. G.; Muthohar, M. F.; Prayuda, J. D. A.; Priagung, K.; Choi, D. “Time-based ddos detection and mitigation for SDN controller”. In: 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2015, pp. 550–553.

[140] Diekmann, C.; Korsten, A.; Carle, G. “Demonstrating topos: Theorem-prover-based synthesis of secure network configurations”. In: 11th International Conference on Network and Service Management (CNSM), 2015, pp. 366–371.

[141] Dierks, T.; Rescorla, E. “The Transport Layer Security (TLS) Protocol Version 1.2”, 2008. Disponível em https://tools.ietf.org/html/rfc5246. Acesso em 01/06/2018.

[142] Diffie, W.; Hellman, M. “New directions in cryptography”, *IEEE Transactions on Information Theory*, vol. 22–6, Nov 1976, pp. 644–654.

[143] Ding, A. Y.; Crowcroft, J.; Tarkoma, S.; Flinck, H. “Software defined networking for security enhancement in wireless mobile networks”, *Computer Networks*, vol. 66, 2014, pp. 94–101, Leonard Kleinrock Tribute Issue: A Collection of Papers by his Students.

[144] Ding, K.; Wang, X.; Zhang, G.; Wang, Z.; Chen, M. “A flow-based authentication handover mechanism for multi-domain SDN mobility environment”, *China Communications*, vol. 14–9, 2017, pp. 127–143.

[145] Diovu, R. C.; Agee, J. T. “A cloud-based openflow firewall for mitigation against ddos attacks in smart grid ami networks”. In: IEEE PES PowerAfrica, 2017, pp. 28–33.

[146] Dolev, S.; David, S. “SDN-based private interconnection”. In: IEEE 13th International Symposium on Network Computing and Applications (NCA), 2014, pp. 129–136.

[147] Dong, P.; Du, X.; Zhang, H.; Xu, T. “A detection method for a novel ddos attack against SDN controllers by vast new low-traffic flows”. In: IEEE International Conference on Communications (ICC), 2016, pp. 1–6.

[148] Dotcenko, S.; Vladyko, A.; Letenko, I. “A fuzzy logic-based information security management for software-defined networks”. In: 16th International Conference on Advanced Communication Technology (ICACT), 2014, pp. 167–171.

[149] Dover, J. “A denial of service attack against the open floodlight SDN controller”, 2013. Tech. rep., Dover Networks LLC. Disponível em http://dovernetworks.com/wp-content/uploads/2013/12/OpenFloodlight-12302013.pdf. Acesso em 01/06/2018.

[150] Dover, J. “A switch table vulnerability in the open floodlight SDN controller”, 2014. Tech. rep., Dover Networks LLC. Disponível em http://dovernetworks.com/wp-content/uploads/2014/03/OpenFloodlight-03052014.pdf. Acesso em 01/06/2018.

[151] Duan, Q.; Al-Shaer, E.; Jafarian, H. “Efficient random route mutation considering flow and network constraints”. In: IEEE Conference on Communications and Network Security (CNS), 2013, pp. 260–268.

[152] Duan, X.; Wang, X. “Authentication handover and privacy protection in 5g hetnets using software-defined networking”, *IEEE Communications Magazine*, vol. 53–4, 2015, pp. 28–35.

[153] Duan, X.; Wang, X. “Fast authentication in 5g hetnet through SDN enabled weighted secure-context-information transfer”. In: IEEE International Conference on Communications (ICC), 2016, pp. 1–6.

[154] Durner, R.; Kellerer, W. “The cost of security in the SDN control plane”. In: ACM International Conference on emerging Networking EXperiments and Technologies Student Workshop, 2015, pp. 1–3.

[155] Durner, R.; Lorenz, C.; Wiedemann, M.; Kellerer, W. “Detecting and mitigating denial of service attacks against the data plane in software defined networks”. In: IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–6.

[156] Duttaluri, S. M.; Karimi, B. “Security solutions and design scenarios for software defined data centers”. In: IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017, pp. 341–351.

[157] Eastlake, D.; Jones, P. “US Secure Hash Algorithm 1 (SHA1)”, 2001. Disponível em https://tools.ietf.org/html/rfc3174. Acesso em 01/06/2018.

[158] Eggert, T.; Khondoker, R. “Security analysis of approaches to integrate middleboxes into software defined networks”. In: 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), 2016, pp. 1–7.

[159] Eronen, P.; Tschofenig, H. “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)”, 2005. Disponível em https://tools.ietf.org/html/rfc4279. Acesso em 01/06/2018.

[160] Etaiwi, W.; Biltawi, M.; Almajali, S. “Securing distributed SDN controllers against dos attacks”. In: International Conference on New Trends in Computing Sciences (ICTCS), 2017, pp. 203–206.

[161] Ezekiel, S.; Divakaran, D. M.; Gurusamy, M. “Dynamic attack mitigation using SDN”. In: 27th International Telecommunication Networks and Applications Conference (ITNAC), 2017, pp. 1–6.

[162] Fan, W.; Fernandez, D. “A novel SDN based stealthy tcp connection handover mechanism for hybrid honeypot systems”. In: IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–9.

[163] Fan, X.; Lu, Z.; Ju, L.; Mu, D. “The research on security SDN south interface based on otr protocol”. In: 16th International Symposium on Communications and Information Technologies (ISCIT), 2016, pp. 629–633.

[164] Fan, Z.; Liu, R. “Investigation of machine learning based network traffic classification”. In: International Symposium on Wireless Communication Systems (ISWCS), 2017, pp. 1–6.

[165] Fayazbakhsh, S. K.; Chiang, L.; Sekar, V.; Yu, M.; Mogul, J. C. “Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags”. In: 11th USENIX Conference on Networked Systems Design and Implementation, 2014, pp. 533–546.

[166] Fayazbakhsh, S. K.; Sekar, V.; Yu, M.; Mogul, J. C. “Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions”. In: 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 19–24.

[167] Feghali, A.; Kilany, R.; Chamoun, M. “SDN security problems and solutions analysis”. In: International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), 2015, pp. 1–5.

[168] Feghali, A.; Kilany, R.; Chamoun, M. “SDN security problems and solutions analysis”. In: International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), 2015, pp. 1–5.

[169] Feldmann, A.; Heyder, P.; Kreutzer, M.; Schmid, S.; Seifert, J. P.; Shulman, H.; Thimmaraju, K.; Waidner, M.; Sieberg, J. “Netco: Reliable routing with unreliable routers”. In: 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), 2016, pp. 128–135.

[170] Ferguson, A. D.; Guha, A.; Liang, C.; Fonseca, R.; Krishnamurthi, S. “Participatory networking: An api for application control of SDNs”, *SIGCOMM Comput. Commun. Rev.*, vol. 43–4, 2013, pp. 327–338.

[171] Foster, N.; Guha, A.; Reitblatt, M.; Story, A.; Freedman, M. J.; Katta, N. P.; Monsanto, C.; Reich, J.; Rexford, J.; Schlesinger, C.; Walker, D.; Harrison, R. “Languages for software-defined networks”, *IEEE Communications Magazine*, vol. 51–2, 2013, pp. 128–134.

[172] Foster, N.; Harrison, R.; Freedman, M. J.; Monsanto, C.; Rexford, J.; Story, A.; Walker, D. “Frenetic: A network programming language”. In: 16th ACM SIGPLAN International Conference on Functional Programming, 2011, pp. 279–291.

[173] Foundation, O. N. “Openflow switch specification, version 1.0.0”, 2009. Disponível em https://www.opennetworking.org/images/stories/downloads/{SDN}-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf. Acesso em 01/06/2018.

[174] Foundation, O. N. “Openflow switch specification, version 1.3.0”, 2012. Disponível em https://www.opennetworking.org/images/stories/downloads/{SDN}-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf. Acesso em 01/06/2018.

[175] Foundation, O. N. “Openflow switch specification, version 1.5.1”, 2015. Disponível em https://www.opennetworking.org/images/stories/downloads/{SDN}-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf. Acesso em 01/06/2018.

[176] François, J.; Dolberg, L.; Festor, O.; Engel, T. “Network security through software defined networking: A survey”. In: Conference on Principles, Systems and Applications of IP Telecommunications, 2014, pp. 1–8.

[177] Francois, J.; Festor, O. “Anomaly traceback using software defined networking”. In: IEEE International Workshop on Information Forensics and Security (WIFS), 2014, pp. 203–208.

[178] Frankel, S.; Glenn, R.; Kelly, S. “The AES-CBC Cipher Algorithm and Its Use with IPsec”, 2003. Disponível em https://tools.ietf.org/html/rfc3602. Acesso em 01/06/2018.

[179] Freire, L.; Neves, M.; Leal, L.; Levchenko, K.; Schaeffer-Filho, A.; Barcellos, M. “Uncovering bugs in p4 programs with assertion-based verification”. In: Symposium on SDN Research, 2018, pp. 1–7.

[180] Furukawa, M.; Kuroda, K.; Ogawa, T.; Miyaho, N. “Highly secure communication service architecture using SDN switch”. In: 10th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), 2015, pp. 1–3.

[181] Fysarakis, K.; Petroulakis, N. E.; Roos, A.; Abbasi, K.; Vizarreta, P.; Petropoulos, G.; Sakic, E.; Spanoudakis, G.; Askoxylakis, I. “A reactive security framework for operational wind parks using service function chaining”. In: IEEE Symposium on Computers and Communications (ISCC), 2017, pp. 663–668.

[182] Gao, J.; Xia, C.; Wang, S.; Zhang, H. “A SDN-based deployment framework for computer network defense policy”. In: 4th International Conference on Computer Science and Network Technology (ICCSNT), 2015, pp. 1253–1258.

[183] Gao, S.; Li, Z.; Xiao, B.; Wei, G. “Security threats in the data plane of software-defined networks”, *IEEE Network*, 2018, pp. 1–6.

[184] Gao, S.; Li, Z.; Yao, Y.; Xiao, B.; Guo, S.; Yang, Y. “Software-defined firewall: Enabling malware traffic detection and programmable security control”. In: Asia Conference on Computer and Communications Security, 2018, pp. 413–424.

[185] Gao, W.; Li, X.; Zhou, B.; Wu, C. “Load balancing fat-tree on long-lived flows: Avoiding congestion in a data center network”, *ZTE Communications*, vol. 12–2, 2014, pp. 57–62.

[186] Garg, G.; Garg, R. “A comparative study for accuracy of anomaly detection methods of adaptive flow counting in SDN”. In: 2nd International Conference on Recent Advances in Engineering Computational Sciences (RAECS), 2015, pp. 1–4.

[187] Garg, G.; Garg, R. “Detecting anomalies efficiently in SDN using adaptive mechanism”. In: 5th International Conference on Advanced Computing Communication Technologies, 2015, pp. 367–370.

[188] Gebert, S.; Zinner, T.; Gray, N.; Durner, R.; Lorenz, C.; Lange, S. “Demonstrating a personalized secure-by-default bring your own device solution based on software defined networking”. In: 28th International Teletraffic Congress (ITC 28), 2016, pp. 197–200.

[189] Gelberger, A.; Yemini, N.; Giladi, R. “Performance analysis of software-defined networking (SDN)”. In: IEEE 21st International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013, pp. 389–393.

[190] Gember, A.; Dragga, C.; Akella, A. “Ecos: Leveraging software-defined networks to support mobile application offloading”. In: 8th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2012, pp. 199–210.

[191] Ghodsi, A.; Shenker, S.; Koponen, T.; Singla, A.; Raghavan, B.; Wilcox, J. “Intelligent design enables architectural evolution”. In: 10th ACM Workshop on Hot Topics in Networks, 2011, pp. 1–6.

[192] Ghosh, U.; Chatterjee, P.; Shetty, S. “A security framework for SDN-enabled smart power grids”. In: IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017, pp. 113–118.

[193] Gillmor, D. “Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)”, 2016. Disponível em https://tools.ietf.org/html/rfc7919. Acesso em 01/06/2018.

[194] Giotis, K.; Androulidakis, G.; Maglaris, V. “Leveraging SDN for efficient anomaly detection and mitigation on legacy networks”. In: 3rd European Workshop on Software Defined Networks, 2014, pp. 85–90.

[195] Giotis, K.; Apostolaki, M.; Maglaris, V. “A reputation-based collaborative schema for the mitigation of distributed attacks in SDN domains”. In: NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 495–501.

[196] Giotis, K.; Argyropoulos, C.; Androulidakis, G.; Kalogeras, D.; Maglaris, V. “Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments”, *Comput. Netw.*, vol. 62, Apr 2014, pp. 122–136.

[197] Gonzalez, C.; Charfadine, S. M.; Flauzac, O.; Nolot, F. “SDN-based security framework for the iot in distributed grid”. In: International Multidisciplinary Conference on Computer and Energy Science (SpliTech), 2016, pp. 1–5.

[198] Govindarajan, K.; Meng, K. C.; Ong, H. “A literature review on software-defined networking (SDN) research topics, challenges and solutions”. In: 5th International Conference on Advanced Computing (ICoAC), 2013, pp. 293–299.

[199] Gray, N.; Zinner, T.; Tran-Gia, P. “Enhancing SDN security by device fingerprinting”. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 879–880.

[200] Grover, N.; Agarwal, N.; Kataoka, K. “liteflow: Lightweight and distributed flow monitoring platform for SDN”. In: 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–9.

[201] Grusho, A.; Grusho, N.; Timonina, E.; Piskovski, V. “Five SDN-oriented directions in information security”. In: International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014, pp. 1–4.

[202] Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. “Nox: Towards an operating system for networks”, *SIGCOMM Comput. Commun. Rev.*, vol. 38–3, 2008, pp. 105–110.

[203] Guodong, T.; Xi, Q.; Chaowen, C. “A SDN security control forwarding mechanism based on cipher identification”. In: IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 1419–1425.

[204] Görkemli, B.; Parlakışık, A. M.; Civanlar, S.; Ulaş, A.; Tekalp, A. M. “Dynamic management of control plane performance in software-defined networks”. In: IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 68–72.

[205] Ha, T.; Yoon, S.; Risdianto, A. C.; Kim, J.; Lim, H. “Suspicious flow forwarding for multiple intrusion detection systems on software-defined networks”, *IEEE Network*, vol. 30–6, 2016, pp. 22–27.

[206] Hadi, F.; Imran, M.; Durad, M. H.; Waris, M. “A simple security policy enforcement system for an institution using SDN controller”. In: 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 2018, pp. 489–494.

[207] Hakiri, A.; Gokhale, A.; Berthou, P.; Schmidt, D. C.; Gayraud, T. “Software-defined networking: Challenges and research opportunities for future internet”, *Computer Networks*, vol. 75, Part A, 2014, pp. 453 – 471.

[208] Halder, B.; Barik, M. S.; Mazumdar, C. “A graph based formalism for detecting flow conflicts in software defined network”. In: IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2017, pp. 1–6.

[209] Haleplidis, E.; Hadi Salim, J.; Denazis, S.; Koufopavlou, O. “Towards a network abstraction model for SDN”, *Journal of Network and Systems Management*, vol. 23–2, 2015, pp. 309–327.

[210] Hameed, S.; Khan, H. A. “Leveraging SDN for collaborative ddos mitigation”. In: International Conference on Networked Systems (NetSys), 2017, pp. 1–6.

[211] Han, W.; Zhao, Z.; Doupé, A.; Ahn, G.-J. “Honeymix: Toward SDN-based intelligent honeynet”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 1–6.

[212] Hao, W.; Jiang, Y.; Gao, J. “Detection mechanisms of rule conflicts in SDN based on a path-tree model”. In: 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 336–339.

[213] Haque, M. R.; Ali, S.; Tan, S. C.; Yusoff, Z.; Kwang, L. C.; Kaspin, I. R.; Ziri, S. R. “Motivation of ddos attack-aware in software defined networking controller placement”. In: International Conference on Computer and Applications (ICCA), 2017, pp. 36–42.

[214] He, D.; Chan, S.; Ni, X.; Guizani, M. “Software-defined-networking-enabled traffic anomaly detection and mitigation”, *IEEE Internet of Things Journal*, vol. 4–6, Dec 2017, pp. 1890–1898.

[215] He, K.; Khalid, J.; Das, S.; Gember-Jacobson, A.; Prakash, C.; Akella, A.; Li, L. E.; Thottan, M. “Latency in software defined networks: Measurements and mitigation techniques”. In: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2015, pp. 435–436.

[216] He, K.; Khalid, J.; Gember-Jacobson, A.; Das, S.; Prakash, C.; Akella, A.; Li, L. E.; Thottan, M. “Measuring control plane latency in SDN-enabled switches”. In: 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 1–6.

[217] He, L.; Xu, C.; Luo, Y. “vtc: Machine learning based traffic classification as a virtual network function”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 53–56.

[218] He, Q.; Wang, Y.; Li, W.; Qiu, X. “Traffic steering of middlebox policy chain based on SDN”. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 754–759.

[219] Hegr, T.; Bohac, L.; Uhlir, V.; Chlumsky, P. “Openflow deployment and concept analysis”, *Advances in Electrical and Electronic Engineering*, vol. 11–5, 2013, pp. 327–335.

[220] Hewlett Packard Enterprise. “Aruba 2920 24g switch datasheet”, 2018. Disponível em https://h20195.www2.hpe.com/v2/GetDocument.aspx?docname=c04111401. Acesso em 01/06/2018.

[221] Ho, B.; Pham-Quoc, C.; Thinh, T. N.; Thoai, N. “A secured openflow-based switch architecture”. In: International Conference on Advanced Computing and Applications (ACOMP), 2016, pp. 83–89.

[222] Hoffman, P. “Algorithms for Internet Key Exchange version 1 (IKEv1)”, 2005. Disponível em https://tools.ietf.org/html/rfc4109. Acesso em 01/06/2018.

[223] Hommes, S.; State, R.; Engel, T. “Implications and detection of dos attacks in openflow-based networks”. In: IEEE Global Communications Conference, 2014, pp. 537–543.

[224] Hong, D.; Kim, J.; Hyun, D.; Jeong, J. P. “A monitoring-based load balancing scheme for network security functions”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2017, pp. 668–672.

[225] Hong, J. B.; Yoon, S.; Lim, H.; Kim, D. S. “Optimal network reconfiguration for software defined networks using shuffle-based online mtd”. In: IEEE 36th Symposium on Reliable Distributed Systems (SRDS), 2017, pp. 234–243.

[226] Hong, K.; Kim, Y.; Choi, H.; Park, J. “SDN-assisted slow http ddos attack defense method”, *IEEE Communications Letters*, vol. 22–4, April 2018, pp. 688–691.

[227] Hong, S.; Xu, L.; Wang, H.; Gu, G. “Poisoning network visibility in software-defined networks: New attacks and countermeasures”. In: Network and Distributed System Security, 2015, pp. 1–15.

[228] Hori, Y.; Mizoguchi, S.; Miyazaki, R.; Yamada, A.; Feng, Y.; Kubota, A.; Sakurai, K. “A comprehensive security analysis checksheet for openflow networks”. In: Advances on Broad-Band Wireless Computing, Communication and Applications, 2017, pp. 231–242.

[229] Housley, R. “A 224-bit One-way Hash Function: SHA-224”, 2004. Disponível em https://tools.ietf.org/html/rfc3874. Acesso em 01/06/2018.

[230] Hu, D.; Hong, P.; Chen, Y. “Fadm: Ddos flooding attack detection and mitigation system in software-defined networking”. In: IEEE Global Communications Conference, 2017, pp. 1–7.

[231] Hu, F.; Hao, Q.; Bao, K. “A survey on software-defined network and openflow: From concept to implementation”, *Communications Surveys Tutorials, IEEE*, vol. 16–4, Fourthquarter 2014, pp. 2181–2206.

[232] Hu, H.; Han, W.; Ahn, G.-J.; Zhao, Z. “Flowguard: Building robust firewalls for software-defined networks”. In: 3rd Workshop on Hot Topics in Software Defined Networking, 2014, pp. 97–102.

[233] Hu, Y.-L.; Su, W.-B.; Wu, L.-Y.; Huang, Y.; Kuo, S.-Y. “Design of event-based intrusion detection system on openflow network”. In: 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2013, pp. 1–2.

[234] Hu, Z.; Wang, M.; Yan, X.; Yin, Y.; Luo, Z. “A comprehensive security architecture for SDN”. In: 18th International Conference on Intelligence in Next Generation Networks, 2015, pp. 30–37.

[235] 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 International Conference on Communication Software and Networks (ICCSN), 2015, pp. 85–92.

[236] Huang, X.; Du, X.; Song, B. “An effective ddos defense scheme for SDN”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–6.

[237] Huong, T. T.; Thanh, N. H. “Software defined networking-based one-packet ddos mitigation architecture”. In: 11th International Conference on Ubiquitous Information Management and Communication, 2017, pp. 1–7.

[238] Hurley, T.; Perdomo, J. E.; Perez-Pons, A. “Hmm-based intrusion detection system for software defined networking”. In: 15th IEEE International Conference on Machine Learning and Applications (ICMLA), 2016, pp. 617–621.

[239] Hussein, A.; Elhajj, I. H.; Chehab, A.; Kayssi, A. “SDN security plane: An architecture for resilient security services”. In: IEEE International Conference on Cloud Engineering Workshop (IC2EW), 2016, pp. 54–59.

[240] Hussein, A.; Elhajj, I. H.; Chehab, A.; Kayssi, A. “SDN verification plane for consistency establishment”. In: IEEE Symposium on Computers and Communication (ISCC), 2016, pp. 519–524.

[241] Hyun, D.; Kim, J.; Hong, D.; Jeong, J. P. “SDN-based network security functions for effective ddos attack mitigation”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2017, pp. 834–839.

[242] Hyun, D.; Kim, J.; Jeong, J. P.; Kim, H.; Park, J.; Ahn, T. “SDN-based network security functions for voip and volte services”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2016, pp. 298–302.

[243] Isong, B.; Kgogo, T.; Lugayizi, F.; Kankuzi, B. “Trust establishment framework between SDN controller and applications”. In: 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017, pp. 101–107.

[244] Jacquin, L.; Shaw, A. L.; Dalton, C. “Towards trusted software-defined networks using a hardware-based integrity measurement architecture”. In: 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–6.

[245] Jafarian, J. H.; Al-Shaer, E.; Duan, Q. “Openflow random host mutation: Transparent moving target defense using software defined networking”. In: 1st Workshop on Hot Topics in Software Defined Networks, 2012, pp. 127–132.

[246] Jäger, B.; Röpke, C.; Adam, I.; Holz, T. “Multi-layer access control for SDN-based telco clouds”. In: Secure IT Systems, 2015, pp. 197–204.

[247] Jager, T.; Kohlar, F.; Schäge, S.; Schwenk, J. “On the security of tls-dhe in the standard model”. In: Advances in Cryptology – CRYPTO 2012, 2012, pp. 273–293.

[248] Jakaria, A.; Rashidi, B.; Rahman, M. A.; Fung, C.; Yang, W. “Dynamic ddos defense resource allocation using network function virtualization”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2017, pp. 37–42.

[249] Jamjoom, H.; Williams, D.; Sharma, U. “Don’t call them middleboxes, call them middlepipes”. In: 3rd Workshop on Hot Topics in Software Defined Networking, 2014, pp. 19–24.

[250] Jasson Casey, C.; Sutton, A.; Dos Reis, G.; Sprintson, A. “Eliminating network protocol vulnerabilities through abstraction and systems language design”. In: 21st IEEE International Conference on Network Protocols (ICNP), 2013, pp. 1–6.

[251] Jeong, C.; Ha, T.; Narantuya, J.; Lim, H.; Kim, J. “Scalable network intrusion detection on virtual SDN environment”. In: IEEE 3rd International Conference on Cloud Networking (CloudNet), 2014, pp. 264–265.

[252] Jeong, J.; Seo, J.; Cho, G.; Kim, H.; Park, J. S. “A framework for security services based on software-defined networking”. In: IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, 2015, pp. 150–153.

[253] Jevtic, S.; Lotfalizadeh, H.; Kim, D. S. “Toward network-based ddos detection in software-defined networks”. In: 12th International Conference on Ubiquitous Information Management and Communication, 2018, pp. 40:1–40:8.

[254] Jiang, H.; Bouabdallah, A.; Aflatoonian, A.; Bonnin, J.-M.; Guillouard, K. “A secure multi-tenant framework for SDN”. In: 9th International Conference on Security of Information and Networks, 2016, pp. 40–44.

[255] Jin, R.; Wang, B. “Malware detection for mobile devices using software-defined networking”. In: 2nd GENI Research and Educational Experiment Workshop, 2013, pp. 81–88.

[256] Jonker, M.; Sperotto, A. “Mitigating ddos attacks using openflow-based software defined networking”. In: Intelligent Mechanisms for Network Configuration and Security, 2015, pp. 129–133.

[257] Juba, Y.; Huang, H.-H.; Kawagoe, K. “Dynamic isolation of network devices using openflow for keeping lan secure from intra-lan attack”, *Procedia Computer Science*, vol. 22, 2013, pp. 810 – 819, 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems.

[258] Juhola, A.; Ahola, T.; Ahola, K. “Adaptive risk management with ontology linked evidential statistics and SDN”. In: European Conference on Software Architecture Workshops, 2014, pp. 1–7.

[259] Juniper Networks. “Juniper qfx5100 ethernet switch”, 2018. Disponível em https://www.juniper.net/assets/us/en/local/pdf/datasheets/1000480-en.pdf. Acesso em 01/06/2018.

[260] Kalkan, K.; Gur, G.; Alagoz, F. “Defense mechanisms against ddos attacks in SDN environment”, *IEEE Communications Magazine*, vol. 55–9, 2017, pp. 175–179.

[261] Kalkan, K.; Gür, G.; Alagöz, F. “SDNscore: A statistical defense mechanism against ddos attacks in SDN environment”. In: IEEE Symposium on Computers and Communications (ISCC), 2017, pp. 669–675.

[262] Kalliola, A.; Lee, K.; Lee, H.; Aura, T. “Flooding ddos mitigation and traffic management with software defined networking”. In: IEEE 4th International Conference on Cloud Networking (CloudNet), 2015, pp. 248–254.

[263] Kamath, A. V.; S, S.; Kataoka, K.; Vijayvergiya, N.; Reddy, G. B.; Phatale, S. “Safe: Software-defined authentication framework”. In: 12th Asian Internet Engineering Conference, 2016, pp. 57–63.

[264] Kamisiński, A.; Fung, C. “Flowmon: Detecting malicious switches in software-defined networks”. In: Workshop on Automated Decision Making for Active Cyber Defense, 2015, pp. 39–45.

[265] Kampanakis, P.; Perros, H.; Beyene, T. “SDN-based solutions for moving target defense network protection”. In: IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014, pp. 1–6.

[266] Kandoi, R.; Antikainen, M. “Denial-of-service attacks in openflow SDN networks”. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 1322–1326.

[267] Kang, J. W.; Park, S. H.; You, J. “Mynah: Enabling lightweight data plane authentication for SDN controllers”. In: 24th International Conference on Computer Communication and Networks (ICCCN), 2015, pp. 1–6.

[268] Kantor, M.; State, R.; Engel, T.; Ormazabal, G. “A policy-based per-flow mobility management system design”. In: Principles, Systems and Applications on IP Telecommunications, 2015, pp. 35–42.

[269] Karame, G. “Towards trustworthy network measurements”. In: *Trust and Trustworthy Computing*, Springer Berlin Heidelberg, 2013, *Lecture Notes in Computer Science*, vol. 7904, pp. 83–91.

[270] Karmakar, K. K.; Varadharajan, V.; Tupakula, U. “On the design and implementation of a security architecture for software defined networks”. In: IEEE 18th International

Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016, pp. 671–678.

[271] Karmakar, K. K.; Varadharajan, V.; Tupakula, U. “Mitigating attacks in software defined network (SDN)”. In: 4th International Conference on Software Defined Systems (SDS), 2017, pp. 112–117.

[272] Karmakar, K. K.; Varadharajan, V.; Tupakula, U.; Hitchens, M. “Policy based security architecture for software defined networks”. In: 31st Annual ACM Symposium on Applied Computing, 2016, pp. 658–663.

[273] Kaur, K.; Kaur, S.; Gupta, V. “Software defined networking based routing firewall”. In: International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), 2016, pp. 267–269.

[274] Kaur, N.; Singh, A. K.; Kumar, N.; Srivastava, S. “Performance impact of topology poisoning attack in SDN and its countermeasure”. In: 10th International Conference on Security of Information and Networks, 2017, pp. 179–184.

[275] Kaur, R.; Singh, A.; Singh, S.; Sharma, S. “Security of software defined networks: Taxonomic modeling, key components and open research area”. In: International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pp. 2832–2839.

[276] Kaur, S.; Kaur, K.; Gupta, V. “Implementing openflow based distributed firewall”. In: International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds, 2016, pp. 172–175.

[277] Kaur, S.; Kaur, K.; Gupta, V. “Implementing openflow based firewall”. In: 3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016), 2016, pp. 1–3.

[278] Kawai, H.; Omino, T.; Shibahara, H.; Sakamoto, D.; Kanamori, I.; Sonoda, K. “"access authentication solutions"- providing flexible and secure network access”, *NEC Technical Journal*, vol. 8–2, 2014, pp. 25 – 28.

[279] Kent, S. “IP Encapsulating Security Payload (ESP)”, 2005. Disponível em https://www.ietf.org/rfc/rfc4303.txt. Acesso em 01/06/2018.

[280] Kent, S.; Seo, K. “Security Architecture for the Internet Protocol”, 2005. Disponível em https://tools.ietf.org/html/rfc4301. Acesso em 01/06/2018.

[281] Kerdsri, J.; Wipusitwarkun, K. “Data-wise routing in virtualization environment (drive) with multiple level of security for tactical network”. In: IEEE/SICE International Symposium on System Integration (SII), 2012, pp. 933–938.

[282] Kernen, T.; Asghar, J. “Optimised ip multicast architectures for real-time digital workflows”. In: Annual Technical Conference Exhibition, SMPTE 2012, 2012, pp. 1–15.

[283] Kgogo, T.; Isong, B.; Abu-Mahfouz, A. M. “Software defined wireless sensor networks security challenges”. In: IEEE AFRICON, 2017, pp. 1508–1513.

[284] Khan, S.; Gani, A.; Wahab, A. W. A.; Abdelaziz, A.; Bagiwa, M. A. “Fml: A novel forensics management layer for software defined networks”. In: 6th International Conference - Cloud System and Big Data Engineering (Confluence), 2016, pp. 619–623.

[285] Khan, S.; Gani, A.; Wahab, A. W. A.; Guizani, M.; Khan, M. K. “Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art”, *IEEE Communications Surveys Tutorials*, vol. 19–1, Firstquarter 2017, pp. 303–324.

[286] Khimabhai, Y. A.; Rohokale, V. “SDN control plane security in cloud computing against ddos attack”. In: International Conference on Advances in Information Communication Technology & Computing, 2016, pp. 1–5.

[287] Kietkaroon, P.; Watanabe, Y.; Murayama, J.; Hamada, T.; Igarashi, Y. “A dynamic ips allocation scheme using openflow for economical secure networking”. In: 10th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), 2015, pp. 1–3.

[288] Kim, E.; Kim, K.; Lee, S.; Jeong, J. P.; Kim, H. “A framework for managing user-defined security policies to support network security functions”. In: 12th International Conference on Ubiquitous Information Management and Communication, 2018, pp. 1–8.

[289] Kim, G.; An, J.; Kim, K. “A study on authentication mechanism in SEaaS for SDN”. In: 11th International Conference on Ubiquitous Information Management and Communication, 2017, pp. 1–6.

[290] Kim, H.; Feamster, N. “Improving network management with software defined networking”, *Communications Magazine, IEEE*, vol. 51–2, February 2013, pp. 114–119.

[291] Kinoshita, S.; Watanabe, T.; Yamato, J.; Goto, H.; Sone, H. “Implementation and evaluation of an openflow-based access control system for wireless lan roaming”.

In: IEEE 36th Annual Computer Software and Applications Conference Workshops (COMPSACW), 2012, pp. 82–87.

[292] Kitchenham, B.; Charters, S. “Guidelines for performing systematic literature reviews in software engineering”, Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University, 2007.

[293] Klaedtke, F.; Karame, G. O.; Bifulco, R.; Cui, H. “Access control for SDN controllers”. In: 3rd Workshop on Hot Topics in Software Defined Networking, 2014, pp. 219–220.

[294] Klaedtke, F.; Karame, G. O.; Bifulco, R.; Cui, H. “Towards an access control scheme for accessing flows in SDN”. In: 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–6.

[295] Klingel, D.; Khondoker, R.; Marx, R.; Bayarou, K. “Security analysis of software defined networking architectures: Pce, 4d and sane”. In: AINTEC on Asian Internet Engineering Conference, 2014, pp. 15–22.

[296] Kloti, R.; Kotronis, V.; Smith, P. “Openflow: A security analysis”. In: 21st IEEE International Conference on Network Protocols (ICNP), 2013, pp. 1–6.

[297] Koerner, M.; Kao, O. “Oftables: A distributed packet filter”. In: 6th International Conference on Communication Systems and Networks (COMSNETS), 2014, pp. 1–4.

[298] Koning, R.; de Graaff, B.; de Laat, C.; Meijer, R.; Grosso, P. “Interactive analysis of SDN-driven defence against distributed denial of service attacks”. In: IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 483–488.

[299] Korkmaz, E.; Dolgikh, A.; Davis, M.; Skormin, V. “Ics security testbed with delay attack case study”. In: IEEE Military Communications Conference, 2016, pp. 283–288.

[300] Kotani, D.; Okabe, Y. “A packet-in message filtering mechanism for protection of control plane in openflow networks”. In: 10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, 2014, pp. 29–40.

[301] Krawczyk, H.; Paterson, K. G.; Wee, H. “On the security of the tls protocol: A systematic analysis”. In: Advances in Cryptology – CRYPTO 2013, 2013, pp. 429–448.

[302] Kreutz, D.; Ramos, F.; Esteves Verissimo, P.; Esteve Rothenberg, C.; Azodolmolky, S.; Uhlig, S. “Software-defined networking: A comprehensive survey”, *Proceedings of the IEEE*, vol. 103–1, Jan 2015, pp. 14–76.

[303] Kreutz, D.; Ramos, F. M.; Verissimo, P. “Towards secure and dependable software-defined networks”. In: 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 55–60.

[304] Krishnan, R.; Krishnaswamy, D.; Mcdysan, D. “Behavioral security threat detection strategies for data center switches and routers”. In: IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2014, pp. 82–87.

[305] Kuerban, M.; Tian, Y.; Yang, Q.; Jia, Y.; Huebert, B.; Poss, D. “Flowsec: Dos attack mitigation strategy on SDN controller”. In: IEEE International Conference on Networking, Architecture and Storage (NAS), 2016, pp. 1–2.

[306] Kundrát, J.; Vojtěch, J.; Škoda, P.; Vohnout, R.; Radil, J.; Havlis, O. “Yang/netconf roadm: Evolving open dwdm towards SDN applications”, *Journal of Lightwave Technology*, 2018, pp. 1–9.

[307] Kuźniar, M.; Perešíni, P.; Kostić, D. “What you need to know about SDN flow tables”. In: Passive and Active Measurement, 2015, pp. 347–359.

[308] Kyung, S.; Han, W.; Tiwari, N.; Dixit, V. H.; Srinivas, L.; Zhao, Z.; Doupé, A.; Ahn, G. J. “Honeyproxy: Design and implementation of next-generation honeynet via SDN”. In: IEEE Conference on Communications and Network Security (CNS), 2017, pp. 1–9.

[309] Lai, S. F.; Su, H. K.; Hsiao, W. H.; Chen, K. J. “Design and implementation of cloud security defense system with software defined networking technologies”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2016, pp. 292–297.

[310] Lam, J.; Lee, S.-G.; Andrianto, V. C. “Secure switch migration protocol with openflow”. In: International Conference on Information Technology, 2017, pp. 171–174.

[311] Lam, J.-H.; Lee, S.-G.; Lee, H.-J.; Oktian, Y. E. “Securing distributed SDN with ibc”. In: 7th International Conference on Ubiquitous and Future Networks, 2015, pp. 921–925.

[312] Lara, A.; Kolasani, A.; Ramamurthy, B. “Network innovation using openflow: A survey”, *Communications Surveys Tutorials, IEEE*, vol. 16–1, First 2014, pp. 493–512.

[313] Lara, A.; Ramamurthy, B. “Opensec: A framework for implementing security policies using openflow”. In: IEEE Global Communications Conference (GLOBECOM), 2014, pp. 781–786.

[314] Lara, A.; Ramamurthy, B. “Opensec: Policy-based security using software-defined networking”, *IEEE Transactions on Network and Service Management*, vol. 13–1, March 2016, pp. 30–42.

[315] Le, A.; Dinh, P.; Le, H.; Tran, N. C. “Flexible network-based intrusion detection and prevention system on software-defined networks”. In: International Conference on Advanced Computing and Applications (ACOMP), 2015, pp. 106–111.

[316] Lee, C.; Shin, S. “Shield: An automated framework for static analysis of SDN applications”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 29–34.

[317] Lee, J.; Park, M.-w.; Chung, T.-m. “Path information based packet verification for authentication of SDN network manager”. In: Computer Science and its Applications, 2015, pp. 861–866.

[318] Lee, S.; Kim, J.; Shin, S.; Porras, P.; Yegneswaran, V. “Athena: A framework for scalable anomaly detection in software-defined networks”. In: 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2017, pp. 249–260.

[319] Lee, S.; Yoon, C.; Shin, S. “The smaller, the shrewder: A simple malicious application can kill an entire SDN environment”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 23–28.

[320] Lee, Y.; Kim, Y.; Lee, Y. “Untraceable blind packet forwarding using centralized path control”. In: IEEE Military Communications Conference (MILCOM), 2014, pp. 268–273.

[321] Lévai, T.; Pelle, I.; Németh, F.; Gulyás, A. “Epoxide: A modular prototype for SDN troubleshooting”. In: ACM Conference on Special Interest Group on Data Communication, 2015, pp. 359–360.

[322] Li, C.; Qin, Z.; Novak, E.; Li, Q. “Securing SDN infrastructure of iot-fog networks from mitm attacks”, *IEEE Internet of Things Journal*, vol. 4–5, 2017, pp. 1156–1164.

[323] Li, J.; Berg, S.; Zhang, M.; Reiher, P.; Wei, T. “Drawbridge: Software-defined ddos-resistant traffic engineering”. In: ACM Conference on SIGCOMM, 2014, pp. 591–592.

[324] Li, J.; Wolf, T. “A one-way proof-of-work protocol to protect controllers in software-defined networks”. In: ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2016, pp. 123–124.

[325] Li, Q.; Zou, X.; Huang, Q.; Zheng, J.; Lee, P. P. C. “Dynamic packet forwarding verification in SDN”, *IEEE Transactions on Dependable and Secure Computing*, 2018, pp. 1–16.

[326] Li, S.; Doh, I.; Chae, K. “Key management mechanism in alto/SDN based cdni architecture”. In: International Conference on Information Networking (ICOIN), 2015, pp. 110–115.

[327] Li, Y.; Mao, J. “SDN-based access authentication and automatic configuration for ipsec”. In: 4th International Conference on Computer Science and Network Technology (ICCSNT), 2015, pp. 996–999.

[328] Liang, X.; Qiu, X. “A software defined security architecture for SDN-based 5g network”. In: IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), 2016, pp. 17–21.

[329] Lim, S.; Ha, J.; Kim, H.; Kim, Y.; Yang, S. “A SDN-oriented ddos blocking scheme for botnet-based attacks”. In: 6th International Conf on Ubiquitous and Future Networks (ICUFN), 2014, pp. 63–68.

[330] Lim, S.; Yang, S.; Kim, Y.; Yang, S.; Kim, H. “Controller scheduling for continued SDN operation under ddos attacks”, *Electronics Letters*, vol. 51–16, 2015, pp. 1259–1261.

[331] Lin, C.; Wu, C.; Tian, Y.; Wen, Z.; Ji, S. “Pbuf: Sharing buffer to mitigate flooding attacks”. In: IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS), 2017, pp. 392–399.

[332] Lin, P. C.; Li, P. C.; Nguyen, V. L. “Inferring openflow rules by active probing in software-defined networks”. In: 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 415–420.

[333] Lin, P. C.; Liu, J. C.; Chiou, P. R. “An event-based SDN architecture for network security analysis”. In: International Carnahan Conference on Security Technology (ICCST), 2015, pp. 159–164.

[334] Lin, Y. D.; Lai, Y. K.; Wang, C. Y.; Lai, Y. C. “Ofbench: Performance test suite on openflow switches”, *IEEE Systems Journal*, vol. PP–99, 2017, pp. 1–11.

[335] Lin, Y. H.; Kuo, J. J.; Yang, D. N.; Chen, W. T. “A cost-effective shuffling-based defense against http ddos attacks with SDN/nfv”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–7.

[336] Lin, Y. H.; Shen, S. H.; Yang, M. H.; Yang, D. N.; Chen, W. T. “Privacy-preserving deep packet filtering over encrypted traffic in software-defined networks”. In: IEEE International Conference on Communications (ICC), 2016, pp. 1–7.

[337] Lioy, A.; Gardikis, G.; Gaston, B.; Jacquin, L.; Benedictis, M. D.; Angelopoulos, Y.; Xylouris, C. “Nfv-based network protection: The shield approach”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 1–2.

[338] Liu, B.; Bi, J.; Zhou, Y. “Source address validation in software defined networks”. In: ACM SIGCOMM Conference, 2016, pp. 595–596.

[339] Liu, J.; Lai, Y.; Zhang, S. “Fl-guard: A detection and defense system for ddos attack in SDN”. In: International Conference on Cryptography, Security and Privacy, 2017, pp. 107–111.

[340] Liu, W.; Bobba, R. B.; Mohan, S.; Campbell, R. H. “Inter-flow consistency: A novel SDN update abstraction for supporting inter-flow constraints”. In: IEEE Conference on Communications and Network Security (CNS), 2015, pp. 469–478.

[341] Liu, X.; Xue, H.; Feng, X.; Dai, Y. “Design of the multi-level security network switch system which restricts covert channel”. In: IEEE 3rd International Conference on Communication Software and Networks (ICCSN), 2011, pp. 233–237.

[342] Liu, Z.; Yin, X.; Lee, H. “An efficient defense scheme against sip dos attack in SDN using cloud sfw”. In: 9th Asia Joint Conference on Information Security (ASIA JCIS), 2014, pp. 52–55.

[343] Lu, G.; Miao, R.; Xiong, Y.; Guo, C. “Using cpu as a traffic co-processing unit in commodity switches”. In: 1st Workshop on Hot Topics in Software Defined Networks, 2012, pp. 31–36.

[344] Lu, Y.; Wang, M. “An easy defense mechanism against botnet-based ddos flooding attack originated in SDN environment using sflow”. In: 11th International Conference on Future Internet Technologies, 2016, pp. 14–20.

[345] Lu, Z.; Chen, F.; Cheng, G.; Ai, J. “A secure control plane for SDN based on bayesian stackelberg games”. In: 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 1259–1264.

[346] Ludwig, A.; Rost, M.; Foucard, D.; Schmid, S. “Good network updates for bad packets: Waypoint enforcement beyond destination-based routing policies”. In: 13th ACM Workshop on Hot Topics in Networks, 2014, pp. 1–7.

[347] Lukaseder, T.; Hunt, A.; Stehle, C.; Wagner, D.; v. d. Heijden, R.; Kargl, F. “An extensible host-agnostic framework for SDN-assisted ddos-mitigation”. In: IEEE 42nd Conference on Local Computer Networks (LCN), 2017, pp. 619–622.

[348] Luo, S.; Wang, H.; Wu, J.; Li, J.; Guo, L.; Pei, B. “How to defend against sophisticated intrusions in home networks using SDN and nfv”. In: IEEE 83rd Vehicular Technology Conference (VTC Spring), 2016, pp. 1–5.

[349] Luo, S.; Wu, J.; Li, J.; Guo, L. “A multi-stage attack mitigation mechanism for software-defined home networks”, *IEEE Transactions on Consumer Electronics*, vol. 62–2, May 2016, pp. 200–207.

[350] Luo, S.; Wu, J.; Li, J.; Pei, B. “A defense mechanism for distributed denial of service attack in software-defined networks”. In: 9th International Conference on Frontier of Computer Science and Technology, 2015, pp. 325–329.

[351] Ma, D.; Xu, Z.; Lin, D. “Defending blind ddos attack on SDN based on moving target defense”. In: International Conference on Security and Privacy in Communication Networks, 2015, pp. 463–480.

[352] Ma, H.; Ding, H.; Yang, Y.; Mi, Z.; Yang, J. Y.; Xiong, Z. “Bayes-based arp attack detection algorithm for cloud centers”, *Tsinghua Science and Technology*, vol. 21–1, Feb 2016, pp. 17–28.

[353] Ma, H.; Ding, H.; Yang, Y.; Mi, Z.; Zhang, M. “SDN-based arp attack detection for cloud centers”. In: IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and IEEE 12th Intl Conf on Autonomic and Trusted Computing and IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015, pp. 1049–1054.

[354] Maccherani, E.; Femminella, M.; Lee, J.; Francescangeli, R.; Janak, J.; Reali, G.; Schulzrinne, H. “Extending the netserv autonomic management capabilities using openflow”. In: Network Operations and Management Symposium (NOMS), IEEE, 2012, pp. 582–585.

[355] Macedo, R.; de Castro, R.; Santos, A.; Ghamri-Doudane, Y.; Nogueira, M. “Self-organized SDN controller cluster conformations against ddos attacks effects”. In: IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6.

[356] MacFarland, D. C.; Shue, C. A. “The SDN shuffle: Creating a moving-target defense using host-based software-defined networking”. In: 2nd ACM Workshop on Moving Target Defense, 2015, pp. 37–41.

[357] Madson, C.; Glenn, R. “The Use of HMAC-SHA-1-96 within ESP and AH”, 1998. Disponível em https://tools.ietf.org/html/rfc2404. Acesso em 01/06/2018.

[358] Mafioletti, D. R.; Liberatto, A. B.; Villaça, R. d. S.; Dominicini, C. K.; Martinello, M.; Ribeiro, M. R. N. “Latency measurement as a virtualized network function using metherxis”, *SIGCOMM Comput. Commun. Rev.*, vol. 46–4, 2016, pp. 14–16.

[359] Mahesh, A.; Chandrasekaran, A.; ArunKumar, R.; SivaKumar, K.; Vigneshwaran, N. “Cloud based firewall on openflow SDN network”. In: International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 2017, pp. 1–6.

[360] Makanju, A.; Zincir-Heywood, A. N.; Kiyomoto, S. “On evolutionary computation for moving target defense in software defined networks”. In: Genetic and Evolutionary Computation Conference Companion, 2017, pp. 287–288.

[361] Maldonado-Lopez, F. A.; Calle, E.; Donoso, Y. “Detection and prevention of firewall-rule conflicts on software-defined networking”. In: 7th International Workshop on Reliable Networks Design and Modeling (RNDM), 2015, pp. 259–265.

[362] Marotta, A.; Carrozza, G.; Avallone, S.; Manetti, V. “An openflow-based architecture for iaas security”. In: 3rd International Conference on Application and Theory of Automation in Command and Control Systems, 2013, pp. 118–121.

[363] Marsico, A.; Doriguzzi-Corin, R.; Gerola, M.; Siracusa, D.; Schwabe, A. “A non-disruptive automated approach to update SDN applications at runtime”. In: NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 1007–1008.

[364] Martins, J. S. B.; Campos, M. B. “A security architecture proposal for detection and response to threats in SDN networks”. In: IEEE ANDESCON, 2016, pp. 1–4.

[365] Masoud, M.; Jaradat, Y.; Ahmad, A. Q. “On tackling social engineering web phishing attacks utilizing software defined networks (SDN) approach”. In: 2nd International Conference on Open Source Software Computing (OSSCOM), 2016, pp. 1–6.

[366] Masoud, M. Z.; Jaradat, Y.; Jannoud, I. “On preventing arp poisoning attack utilizing software defined network (SDN) paradigm”. In: IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2015, pp. 1–5.

[367] Masse, M. “REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces”. O’Reilly Media, 2011.

[368] Matias, J.; Garay, J.; Mendiola, A.; Toledo, N.; Jacob, E. “Flownac: Flow-based network access control”. In: 3rd European Workshop on Software Defined Networks (EWSDN), 2014, pp. 79–84.

[369] Matsumoto, S.; Hitz, S.; Perrig, A. “Fleet: Defending SDNs from malicious administrators”. In: 3rd Workshop on Hot Topics in Software Defined Networking, 2014, pp. 103–108.

[370] McHale, L.; Casey, J.; Gratz, P.; Sprintson, A. “Stochastic pre-classification for SDN data plane matching”. In: IEEE 22nd International Conference on Network Protocols (ICNP), 2014, pp. 596–602.

[371] McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. “Openflow: Enabling innovation in campus networks”, *SIGCOMM Comput. Commun. Rev.*, vol. 38–2, Mar 2008, pp. 69–74.

[372] Medved, J.; Varga, R.; Tkacik, A.; Gray, K. “Opendaylight: Towards a model-driven SDN controller architecture”. In: IEEE 15th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)(WOWMOM), 2014, pp. 1–6.

[373] Meghana, J. S.; Subashri, T.; Vimal, K. R. “A survey on arp cache poisoning and techniques for detection and mitigation”. In: 4th International Conference on Signal Processing, Communication and Networking (ICSCN), 2017, pp. 1–6.

[374] Mehdi, S. A.; Khalid, J.; Khayam, S. A. “Revisiting traffic anomaly detection using software defined networking”. In: 14th International Conference on Recent Advances in Intrusion Detection, 2011, pp. 161–180.

[375] Merkle, J.; Lochter, M. “Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)”, 2013. Disponível em https://tools.ietf.org/html/rfc7027. Acesso em 01/06/2018.

[376] Meti, N.; Narayan, D. G.; Baligar, V. P. “Detection of distributed denial of service attacks using machine learning algorithms in software defined networks”. In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017, pp. 1366–1371.

[377] Mihai, G.; Victor, V. “Achieving ddos resiliency in a software defined network by intelligent risk assessment based on neural networks and danger theory”. In: IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), 2014, pp. 319–324.

[378] Miyazaki, R.; Kawamoto, J.; Matsumoto, S.; Sakurai, K. “Host independent and distributed detection system of the network attack by using openflow”. In: International Conference on Information Networking (ICOIN), 2017, pp. 236–241.

[379] Mizrahi, T.; Saat, E.; Moses, Y. “Timed consistent network updates”. In: 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 1–14.

[380] Mizrahi, T.; Saat, E.; Moses, Y. “Timed consistent network updates in software-defined networks”, *IEEE/ACM Transactions on Networking*, vol. 24–6, December 2016, pp. 3412–3425.

[381] Mohammadi, R.; Javidan, R.; Conti, M. “Slicots: An SDN-based lightweight countermeasure for TCP SYN flooding attacks”, *IEEE Transactions on Network and Service Management*, vol. 14–2, June 2017, pp. 487–497.

[382] Mohammadi, R.; Javidan, R.; Keshtgary, M.; Conti, M.; Lal, C. “Practical extensions to countermeasure dos attacks in software defined networking”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 1–6.

[383] Mohan, P. M.; Lim, T. J.; Gurusamy, M. “Fragmentation-based multipath routing for attack resilience in software defined networks”. In: IEEE 41st Conference on Local Computer Networks (LCN), 2016, pp. 583–586.

[384] Mohan, P. M.; Truong-Huu, T.; Gurusamy, M. “Towards resilient in-band control path routing with malicious switch detection in SDN”. In: 10th International Conference on Communication Systems Networks (COMSNETS), 2018, pp. 9–16.

[385] Monshizadeh, M.; Khatri, V.; Kantola, R. “An adaptive detection and prevention architecture for unsafe traffic in SDN enabled mobile networks”. In: IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 883–884.

[386] Monshizadeh, M.; Khatri, V.; Kantola, R. “Detection as a service: An SDN application”. In: 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 285–290.

[387] Morales, L. V.; Murillo, A. F.; Rueda, S. J. “Extending the floodlight controller”. In: IEEE 14th International Symposium on Network Computing and Applications, 2015, pp. 126–133.

[388] Moraney, J.; Raz, D. “Efficient detection of flow anomalies with limited monitoring resources”. In: 12th International Conference on Network and Service Management (CNSM), 2016, pp. 55–63.

[389] Morzhov, S.; Alekseev, I.; Nikitinskiy, M. “Firewall application for floodlight SDN controller”. In: International Siberian Conference on Control and Communications (SIBCON), 2016, pp. 1–5.

[390] Morzhov, S. V.; Nikitinskiy, M. A. “Development and research of the prefirewall network application for floodlight SDN controller”. In: Moscow Workshop on Electronic and Networking Technologies (MWENT), 2018, pp. 1–4.

[391] Mostovich, D.; Fabrikantov, P.; Vladyko, A.; Buinevich, M. “High-level vulnerabilities of software-defined networking in the context of telecommunication network evolution”. In: IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017, pp. 184–186.

[392] Mousavi, S. M.; St-Hilaire, M. “Early detection of ddos attacks against SDN controllers”. In: International Conference on Computing, Networking and Communications (ICNC), 2015, pp. 77–81.

[393] Mowla, N. I.; Doh, I.; Chae, K. “An efficient defense mechanism for spoofed ip attack in SDN based cdni”. In: International Conference on Information Networking (ICOIN), 2015, pp. 92–97.

[394] Mushi, M.; Dutta, R. “Data-driven study of network administration in the evolving landscape of software defined networking”. In: Workshop on Human Centered Big Data Research, 2014, pp. 14–18.

[395] Nadar, S.; Chaudhari, S. “Proactive-routing path update in software defined networks(SDN)”. In: International Conference on Intelligent Computing and Control (I2C2), 2017, pp. 1–3.

[396] Nadig, D.; Ramamurthy, B.; Bockelman, B.; Swanson, D. “Identifying anomalies in gridftp transfers for data-intensive science through application-awareness”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2018, pp. 7–12.

[397] Nagai, R.; Kurihara, W.; Higuchi, S.; Hirotsu, T. “Design and implementation of an openflow-based tcp syn flood mitigation”. In: 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2018, pp. 37–42.

[398] Nagarathna, R.; Shalinie, S. M. “Slamhha: A supervised learning approach to mitigate host location hijacking attack on SDN controllers”. In: 4th International Conference on Signal Processing, Communication and Networking (ICSCN), 2017, pp. 1–7.

[399] Nam, T. M.; Phong, P. H.; Khoa, T. D.; Huong, T. T.; Nam, P. N.; Thanh, N. H.; Thang, L. X.; Tuan, P. A.; Dung, L. Q.; Loi, V. D. “Self-organizing map-based approaches in ddos flooding detection using SDN”. In: International Conference on Information Networking (ICOIN), 2018, pp. 249–254.

[400] Namal, S.; Ahmad, I.; Gurtov, A.; Ylianttila, M. “Enabling secure mobility with openflow”. In: Future Networks and Services (SDN4FNS), IEEE SDN for, 2013, pp. 1–5.

[401] Nanda, S.; Zafari, F.; DeCusatis, C.; Wedaa, E.; Yang, B. “Predicting network attack patterns in SDN using machine learning approach”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016, pp. 167–172.

[402] Natanzi, S. B. H.; Majma, M. R. “Secure distributed controllers in SDN based on ecc public key infrastructure”. In: International Conference on Electrical and Computing Technologies and Applications (ICECTA), 2017, pp. 1–5.

[403] Natanzi, S. B. H.; Majma, M. R. “Secure northbound interface for SDN applications with ntru public key infrastructure”. In: IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), 2017, pp. 452–458.

[404] Navid, W.; Bhutta, M. N. M. “Detection and mitigation of denial of service (dos) attacks using performance aware software defined networking (SDN)”. In: International Conference on Information and Communication Technologies (ICICT), 2017, pp. 47–57.

[405] Ndonda, G. K.; Sadre, R. “A low-delay SDN-based countermeasure to eavesdropping attacks in industrial control systems”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 1–7.

[406] Neelam, D.; Prasenjit, M.; Shashank, S.; Rahamatullah, K. “Research trends in security and ddos in SDN”, *Security and Communication Networks*, vol. 9–18, 2016, pp. 6386–6411.

[407] Nehra, A.; Tripathi, M.; Gaur, M. S. “Ficur: Employing SDN programmability to secure arp”. In: IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017, pp. 1–8.

[408] Nehra, A.; Tripathi, M.; Gaur, M. S. “’global view’ in SDN: Existing implementation, vulnerabilities & threats”. In: 10th International Conference on Security of Information and Networks, 2017, pp. 303–306.

[409] Nehra, A.; Tripathi, M.; Gaur, M. S. “Requirement analysis for abstracting security in software defined network”. In: 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2017, pp. 1–8.

[410] Networks, E.-C. “Edge-core as4600-54t switch datasheet”, 2018. Disponível em https://www.edge-core.com/_upload/images/AS4600-54T_ONIE_DS_R04_20151029.pdf. Acesso em 01/06/2018.

[411] NEU, C.; TATSCH, C.; LUNARDI, R.; MICHELIN, R.; OROZCO, A.; ZORZO, A. “Lightweight ips for port scan in openflow SDN networks”. In: IEEE/IFIP Workshop on Security for Emerging Distributed Network Technologies, 2018, pp. 1–12.

[412] Neu, C. V.; Zorzo, A. F.; Orozco, A. M. S.; Michelin, R. A. “An approach for detecting encrypted insider attacks on openflow SDN networks”. In: 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016, pp. 210–215.

[413] Nguyen, S. N.; Choi, J.; Kim, K. “Suspicious traffic detection based on edge gateway sampling method”. In: 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2017, pp. 243–246.

[414] Nguyen, T. H.; Yoo, M. “Attacks on host tracker in SDN controller: Investigation and prevention”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2016, pp. 610–612.

[415] Nguyen, T.-H.; Yoo, M. “Analysis of link discovery service attacks in SDN controller”. In: International Conference on Information Networking (ICOIN), 2017, pp. 259–261.

[416] Nobakht, M.; Sivaraman, V.; Boreli, R. “A host-based intrusion detection and mitigation framework for smart home iot using openflow”. In: 11th International Conference on Availability, Reliability and Security (ARES), 2016, pp. 147–156.

[417] Numan, M.; Hashim, F.; Latiff, N. A. A. “Detection and mitigation of arp storm attacks using software defined networks”. In: IEEE 13th Malaysia International Conference on Communications (MICC), 2017, pp. 181–186.

[418] Nunes, B.; Mendonca, M.; Nguyen, X.-N.; Obraczka, K.; Turletti, T. “A survey of software-defined networking: Past, present, and future of programmable networks”, *Communications Surveys Tutorials, IEEE*, vol. 16–3, 2014, pp. 1617–1634.

[419] Oktian, Y.; Lee, S.; Lee, H. “Mitigating denial of service (dos) attacks in openflow networks”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2014, pp. 325–330.

[420] Oktian, Y. E.; Lee, S.; Lee, H.; Lam, J. “Secure your northbound SDN api”. In: 7th International Conference on Ubiquitous and Future Networks, 2015, pp. 919–920.

[421] Oktian, Y. E.; Lee, S.; Lee, H.; Lam, J. “Distributed SDN controller system: A survey on design choice”, *Computer Networks*, vol. 121, 2017, pp. 100 – 111.

[422] Ombase, P. M.; Kulkarni, N. P.; Bagade, S. T.; Mhaisgawali, A. V. “Dos attack mitigation using rule based and anomaly based techniques in software defined networking”. In: International Conference on Inventive Computing and Informatics (ICICI), 2017, pp. 469–475.

[423] ONF. “Open networking foundation”, 2011. Disponível em https://www.opennetworking.org. Acesso em 01/06/2018.

[424] ONRC. “Open networking research center”, 2012. Disponível em www.http://onrc.net/. Acesso em 01/06/2018.

[425] Open Networking Foundation. “Software-Defined Networking: The New Norm for Networks”, 2012. Disponível em https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. Acesso em 01/06/2018.

[426] Open Networking Foundation. “Threat analysis for the SDN architecture”, 2016. Tech. Recomm. TR-530. Disponível em https://www.opennetworking.org/wp-content/uploads/2014/10/Threat_Analysis_for_the_SDN_Architecture.pdf. Acesso em 01/06/2018.

[427] O’Raw, J.; Laverty, D. M.; Morrow, D. J. “Iec 61850 substation configuration language as a basis for automated security and SDN configuration”. In: IEEE Power Energy Society General Meeting, 2017, pp. 1–5.

[428] Orozco, A. M. S.; Neu, C. V.; Michelin, R. A.; Zorzo, A. F. “Security analysis of forwarding strategies in network time measurements using openflow”. In: 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016, pp. 148–153.

[429] Othman, W. M.; Chen, H.; Al-Moalmi, A.; Hadi, A. N. “Implementation and performance analysis of SDN firewall on pox controller”. In: IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 1461–1466.

[430] Padekar, H.; Park, Y.; Hu, H.; Chang, S.-Y. “Enabling dynamic access control for controller applications in software-defined networks”. In: 21st ACM on Symposium on Access Control Models and Technologies, 2016, pp. 51–61.

[431] Padmaja, S.; Vetriselvi, V. “Mitigation of switch-dos in software defined network”. In: International Conference on Information Communication and Embedded Systems (ICICES), 2016, pp. 1–5.

[432] Padon, O.; Immerman, N.; Karbyshev, A.; Lahav, O.; Sagiv, M.; Shoham, S. “Decentralizing SDN policies”. In: 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2015, pp. 663–676.

[433] Paladi, N. “Towards secure SDN policy management”. In: IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), 2015, pp. 607–611.

[434] Pamukchiev, A.; Jouet, S.; Pezaros, D. P. “Distributed network anomaly detection on an event processing framework”. In: 14th IEEE Annual Consumer Communications Networking Conference (CCNC), 2017, pp. 659–664.

[435] Pan, J.; Paul, S.; Jain, R. “A survey of the research on future internet architectures”, *Communications Magazine, IEEE*, vol. 49–7, 2011, pp. 26–36.

[436] Pan, X.; Yegneswaran, V.; Chen, Y.; Porras, P.; Shin, S. “Hogmap: Using SDNs to incentivize collaborative security monitoring”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 7–12.

[437] Pandya, B.; Parmar, S.; Saquib, Z.; Saxena, A. “Framework for securing SDN southbound communication”. In: International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017, pp. 1–5.

[438] Park, T.; Kim, Y.; Shin, S. “Unisafe: A union of security actions for software switches”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 13–18.

[439] Park, Y.; Chandaliya, P.; Muralidharan, A.; Kumar, N.; Hu, H. “Dynamic defense provision via network functions virtualization”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2017, pp. 43–46.

[440] Park, Y.; Chang, S. Y.; Krishnamurthy, L. M. “Watermarking for detecting freeloader misbehavior in software-defined networks”. In: International Conference on Computing, Networking and Communications (ICNC), 2016, pp. 1–6.

[441] Parker, T.; Jones, J.; Mayberry, J.; Chanman, G.; Staples, Z.; McEachen, J.; Tummala, M. “Defensive cyber operations in a software-defined network”. In: 49th Hawaii International Conference on System Sciences (HICSS), 2016, pp. 5561–5568.

[442] Parulkar, G.; Tofigh, T.; Leenheer, M. D. “SDN control of packet-over-optical networks”. In: Optical Fiber Communication Conference, 2015, pp. 1–27.

[443] Passarella, A. “Review: A survey on content-centric technologies for the current internet: Cdn and p2p solutions”, *Comput. Commun.*, vol. 35–1, 2012, pp. 1–32.

[444] Passito, A.; Mota, E.; Bennesby, R.; Fonseca, P. “Agnos: A framework for autonomous control of software-defined networks”. In: IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), 2014, pp. 405–412.

[445] Pattaranantakul, M.; Tseng, Y.; He, R.; Zhang, Z.; Meddahi, A. “A first step towards security extension for nfv orchestrator”. In: ACM International Workshop on Security in Software Defined Networks: Network Function Virtualization, 2017, pp. 25–30.

[446] Pelle, I.; Lévai, T.; Németh, F.; Gulyás, A. “One tool to rule them all: A modular troubleshooting framework for SDN (and other) networks”. In: 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 1–7.

[447] Pena, J.; Yu, W. “Development of a distributed firewall using software defined networking technology”. In: 4th IEEE International Conference on Information Science and Technology (ICIST), 2014, pp. 449–452.

[448] Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. “Systematic mapping studies in software engineering”. In: 12th International Conference on Evaluation and Assessment in Software Engineering, 2008, pp. 68–77.

[449] Phan, T. V.; Bao, N. K.; Kim, Y.; Lee, H.-J.; Park, M. “Optimizing resource allocation for elastic security vnfs in the SDNfv-enabled cloud computing”. In: International Conference on Information Networking (ICOIN), 2017, pp. 163–166.

[450] Phan, T. V.; Bao, N. K.; Park, M. “A novel hybrid flow-based handler with ddos attacks in software-defined networking”. In: Intl IEEE Conferences on Ubiquitous

Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016, pp. 350–357.

[451] Phan, T. V.; Toan, T. V.; Tuyen, D. V.; Huong, T. T.; Thanh, N. H. “Openflowsia: An optimized protection scheme for software-defined networks from flooding attacks”. In: IEEE 6th International Conference on Communications and Electronics (ICCE), 2016, pp. 13–18.

[452] Pica8 Open Networking. “Pica8 p-3297 datasheet”, 2014. Disponível em http://www.colfaxdirect.com/store/pc/catalog/pica8-p3297.pdf. Acesso em 01/06/2018.

[453] Piedrahita, A. F. M.; Rueda, S.; Mattos, D. M. F.; Duarte, O. C. M. B. “Flowfence: a denial of service defense system for software defined networking”. In: Global Information Infrastructure and Networking Symposium (GIIS), 2015, pp. 1–6.

[454] Pinto, P.; Pinto, L. “The use of parametric statistical tests to detect network traffic anomalies on SDN architectures”, *WSEAS Transactions on Communications*, vol. 13, 2014, pp. 659 – 664.

[455] Pisharody, S.; Chowdhary, A.; Huang, D. “Security policy checking in distributed SDN based clouds”. In: IEEE Conference on Communications and Network Security (CNS), 2016, pp. 19–27.

[456] Polat, H.; Polat, O. “The effects of dos attacks on odl and pox SDN controllers”. In: 8th International Conference on Information Technology (ICIT), 2017, pp. 554–558.

[457] Porras, P.; Shin, S.; Yegneswaran, V.; Fong, M.; Tyson, M.; Gu, G. “A security enforcement kernel for openflow networks”. In: 1st Workshop on Hot Topics in Software Defined Networks, 2012, pp. 121–126.

[458] POX homepage, 2018. Disponível em http://www.noxrepo.org/pox. Acesso em 01/06/2018.

[459] Presuhn, R. “Version 2 of the protocol operations for the simple network management protocol (snmp)”, 2002. Disponível em http://www.ietf.org/rfc/rfc3416.txt. Acesso em 01/06/2018.

[460] Pritchard, S. W.; Hancke, G. P.; Abu-Mahfouz, A. M. “Security in software-defined wireless sensor networks: Threats, challenges and potential solutions”. In: IEEE 15th International Conference on Industrial Informatics (INDIN), 2017, pp. 168–173.

[461] Qasmaoui, Y.; Haqiq, A. “Solid-flow: A flow rules security mechanism for SDN”. In: 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), 2017, pp. 1–7.

[462] Qazi, Z. A.; Tu, C.-C.; Chiang, L.; Miao, R.; Sekar, V.; Yu, M. “Simple-fying middlebox policy enforcement using SDN”, *SIGCOMM Comput. Commun. Rev.*, vol. 43–4, Aug 2013, pp. 27–38.

[463] Qi, C.; Wu, J.; Cheng, G.; Ai, J.; Zhao, S. “An aware-scheduling security architecture with priority-equal multi-controller for SDN”, *China Communications*, vol. 14–9, 2017, pp. 144–154.

[464] Qi, C.; Wu, J.; Hu, H.; Cheng, G. “Dynamic-scheduling mechanism of controllers based on security policy in software-defined network”, *Electronics Letters*, vol. 52–23, 2016, pp. 1918–1920.

[465] Qi, C.; Wu, J.; Hu, H.; Cheng, G.; Liu, W.; Ai, J.; Yang, C. “An intensive security architecture with multi-controller for SDN”. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016, pp. 401–402.

[466] Qian, Y.; You, W.; Qian, K. “Openflow flow table overflow attacks and countermeasures”. In: European Conference on Networks and Communications (EuCNC), 2016, pp. 205–209.

[467] Qiu, X.; Cheng, F.; Wang, W.; Zhang, G.; Qiu, Y. “A security controller-based software defined security architecture”. In: 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), 2017, pp. 191–195.

[468] Raghav, P.; Dua, A. “Enhancing flow security in ryu controller through set operations”. In: 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 1265–1269.

[469] Raghavan, B.; Casado, M.; Koponen, T.; Ratnasamy, S.; Ghodsi, A.; Shenker, S. “Software-defined internet architecture: Decoupling architecture from infrastructure”. In: 11th ACM Workshop on Hot Topics in Networks, 2012, pp. 43–48.

[470] Rahman, M.; Despins, C.; Affes, S. “Service differentiation in software defined virtual heterogeneous wireless networks”. In: IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB), 2015, pp. 1–5.

[471] Ranjbar, A.; Komu, M.; Salmela, P.; Aura, T. “An SDN-based approach to enhance the end-to-end security: SSL/TLS case study”. In: NOMS - IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 281–288.

[472] Rao, R. G.; Nene, M. J. “Sedos-7: A proactive mitigation approach against edos attacks in cloud computing”. In: International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017, pp. 965–970.

[473] Rawat, D. B.; Reddy, S. R. “Software defined networking architecture, security and energy efficiency: A survey”, *IEEE Communications Surveys Tutorials*, vol. 19–1, Firstquarter 2017, pp. 325–346.

[474] Rengaraju, P.; Kumar, S. S.; Lung, C. H. “Investigation of security and qos on SDN firewall using mac filtering”. In: International Conference on Computer Communication and Informatics (ICCCI), 2017, pp. 1–5.

[475] Rengaraju, P.; Ramanan, V. R.; Lung, C. H. “Detection and prevention of dos attacks in software-defined cloud networks”. In: IEEE Conference on Dependable and Secure Computing, 2017, pp. 217–223.

[476] Robertson, S.; Alexander, S.; Micallef, J.; Pucci, J.; Tanis, J.; Macera, A. “Cindam: Customized information networks for deception and attack mitigation”. In: IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2015, pp. 114–119.

[477] Rodriguez Natal, A.; Jakab, L.; Portolés, M.; Ermagan, V.; Natarajan, P.; Maino, F.; Meyer, D.; Cabellos Aparicio, A. “LISP-MN: Mobile networking through LISP”, *Wireless Personal Communications*, vol. 70–1, 2013, pp. 253–266.

[478] Rodriguez-Natal, A.; Portoles-Comeras, M.; Ermagan, V.; Lewis, D.; Farinacci, D.; Maino, F.; Cabellos-Aparicio, A. “Lisp: a southbound SDN protocol?”, *IEEE Communications Magazine*, vol. 53–7, 2015, pp. 201–207.

[479] Rosendo, D.; Endo, P. T.; Sadok, D.; Kelner, J. “An autonomic and policy-based authorization framework for openflow networks”. In: 13th International Conference on Network and Service Management (CNSM), 2017, pp. 1–5.

[480] Rotsos, C.; Sarrar, N.; Uhlig, S.; Sherwood, R.; Moore, A. W. “Oflops: An open framework for openflow switch evaluation”. In: 13th International Conference on Passive and Active Measurement, 2012, pp. 85–95.

[481] Rupasinghe, P. L.; Kulatunga, K. M. D. S. B.; Murray, I.; Keseva, K. “SDN based security solution for legislative email communications: Safe guarding communication”. In: International Conference on Computing, Communication and Automation (ICCCA), 2016, pp. 620–624.

[482] Sahay, R.; Blanc, G.; Zhang, Z.; Toumi, K.; Debar, H. “Adaptive policy-driven attack mitigation in SDN”. In: 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures, 2017, pp. 1–6.

[483] Sahoo, K. S.; Sahoo, B.; Panda, A. “A secured SDN framework for iot”. In: International Conference on Man and Machine Interfacing (MAMI), 2015, pp. 1–4.

[484] Sahoo, K. S.; Tiwary, M.; Sahoo, B. “Detection of high rate ddos attack from flash events using information metrics in software defined networks”. In: 10th International Conference on Communication Systems Networks (COMSNETS), 2018, pp. 421–424.

[485] Sahri, N.; Okamura, K. “Collaborative spoofing detection and mitigation – SDN based looping authentication for dns services”. In: IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 2016, pp. 565–570.

[486] Sahri, N.; Okamura, K. “Protecting dns services from ip spoofing: SDN collaborative authentication approach”. In: 11th International Conference on Future Internet Technologies, 2016, pp. 83–89.

[487] Saida, Y.; Iihoshi, T.; Morita, G.; Watanabe, Y.; Karino, S. “Implementing secure communications for business-use smart devices by applying openflow”, *NEC Technical Journal*, vol. 7–3, 2013, pp. 117 – 121.

[488] Salim, J. H.; Ogawa, K. “SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol”, 2010. Disponível em https://tools.ietf.org/html/rfc5811. Acesso em 01/06/2018.

[489] Samociuk, D. “Secure communication between openflow switches and controllers”. In: International Conference on Advances in Future Internet, 2015, pp. 32–37.

[490] Samson, A.; Gopalan, N. P. “Software defined networking: Identification of pathways for security threats”. In: International Conference on Informatics and Analytics, 2016, pp. 1–6.

[491] Santos, M.; de Oliveira, B.; Margi, C.; Nunes, B.; Turletti, T.; Obraczka, K. “Software-defined networking based capacity sharing in hybrid networks”. In: 21st IEEE International Conference on Network Protocols (ICNP), 2013, pp. 1–6.

[492] Sasaki, T.; Hatano, Y.; Sonoda, K.; Morita, Y.; Shimonishi, H.; Okamura, T. “Load distribution of an openflow controller for role-based network access control”. In: Network Operations and Management Symposium (APNOMS), 15th Asia-Pacific, 2013, pp. 1–6.

[493] Sasaki, T.; Pappas, C.; Lee, T.; Hoefler, T.; Perrig, A. “SDNsec: Forwarding accountability for the SDN data plane”. In: 25th International Conference on Computer Communication and Networks (ICCCN), 2016, pp. 1–10.

[494] Sasaki, T.; Perrig, A.; Asoni, D. E. “Control-plane isolation and recovery for a secure SDN architecture”. In: IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 459–464.

[495] Satasiya, D.; D., R. R. “Analysis of software defined network firewall (sdf)”. In: International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2016, pp. 228–231.

[496] Sathya, R.; Thangarajan, R. “Efficient anomaly detection and mitigation in software defined networking environment”. In: 2nd International Conference on Electronics and Communication Systems (ICECS), 2015, pp. 479–484.

[497] Sattar, D.; Matrawy, A. “An empirical model of packet processing delay of the open vswitch”. In: 25th IEEE International Conference on Network Protocols (ICNP), 2017, pp. 1–6.

[498] Sayeed, M. A.; Sayeed, M. A.; Saxena, S. “Intrusion detection system based on software defined network firewall”. In: 1st International Conference on Next Generation Computing Technologies (NGCT), 2015, pp. 379–382.

[499] Scandariato, R.; Wuyts, K.; Joosen, W. “A descriptive study of microsoft’s threat modeling technique”, *Requir. Eng.*, vol. 20–2, Jun 2015, pp. 163–180.

[500] Schehlmann, L.; Abt, S.; Baier, H. “Blessing or curse? revisiting security aspects of software-defined networking”. In: 10th International Conference on Network and Service Management (CNSM), 2014, pp. 382–387.

[501] Schiff, L.; Schmid, S. “Study the past if you would define the future: Implementing secure multi-party SDN updates”. In: IEEE International Conference on Software Science, Technology and Engineering (SWSTE), 2016, pp. 111–116.

[502] Schiff, L.; Thimmaraju, K.; Schmid, S. “Routing-verification-as-a-service (rvaas): Trustworthy routing despite insecure providers”. In: 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), 2016, pp. 113–119.

[503] Schnepf, N.; Badonnel, R.; Lahmadi, A.; Merz, S. “Automated verification of security chains in software-defined networks with synaptic”. In: IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–9.

[504] Scott-Hayward, S. “Design and deployment of secure, robust, and resilient SDN controllers”. In: 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–5.

[505] Scott-Hayward, S.; Kane, C.; Sezer, S. “Operationcheckpoint: SDN application control”. In: IEEE 22nd International Conference on Network Protocols (ICNP), 2014, pp. 618–623.

[506] Scott-Hayward, S.; Natarajan, S.; Sezer, S. “A survey of security in software defined networks”, *IEEE Communications Surveys Tutorials*, vol. 18–1, Firstquarter 2016, pp. 623–654.

[507] Scott-Hayward, S.; O’Callaghan, G.; Sezer, S. “SDN security: A survey”. In: Future Networks and Services (SDN4FNS), IEEE SDN for, 2013, pp. 1–7.

[508] Seeber, S.; Rodosek, G. “Improving network security through SDN in cloud scenarios”. In: 10th International Conference on Network and Service Management (CNSM), 2014, pp. 376–381.

[509] Seeber, S.; Rodosek, G. D. “Towards an adaptive and effective ids using openflow”. In: Intelligent Mechanisms for Network Configuration and Security, 2015, pp. 134–139.

[510] Seeber, S.; Stiemert, L.; Rodosek, G. D. “Towards an SDN-enabled IDS environment”. In: IEEE Conference on Communications and Network Security (CNS), 2015, pp. 751–752.

[511] Shaghaghi, A.; Kaafar, M. A.; Jha, S. “Wedgetail: An intrusion prevention system for the data plane of software defined networks”. In: ACM on Asia Conference on Computer and Communications Security, 2017, pp. 849–861.

[512] Shalimov, A.; Zuikov, D.; Zimarina, D.; Pashkov, V.; Smeliansky, R. “Advanced study of SDN/openflow controllers”. In: 9th Central & Eastern European Software Engineering Conference in Russia, 2013, pp. 1–6.

[513] Shamseddine, M.; Itani, W.; Kayssi, A.; Chehab, A. “Virtualized network views for localizing misbehaving sources in SDN data planes”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–7.

[514] Shan-Shan, J.; Ya-Bin, X. “The apt detection method in SDN”. In: 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 1240–1245.

[515] Shang, G.; Zhe, P.; Bin, X.; Aiqun, H.; Kui, R. “Flooddefender: Protecting data and control plane resources under SDN-aimed dos attacks”. In: IEEE Conference on Computer Communications, 2017, pp. 1–9.

[516] Sharma, P. K.; Singh, S.; Jeong, Y. S.; Park, J. H. “Distblocknet: A distributed blockchains-based secure SDN architecture for iot networks”, *IEEE Communications Magazine*, vol. 55–9, 2017, pp. 78–85.

[517] Sharma, S.; Staessens, D.; Colle, D.; Pickavet, M.; Demeester, P. “Enabling fast failure recovery in openflow networks”. In: Design of Reliable Communication Networks (DRCN), 8th International Workshop on the, 2011, pp. 164–171.

[518] Sharma, V. “Multi-agent based intrusion prevention and mitigation architecture for software defined networks”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2017, pp. 686–692.

[519] Shi, Y.; Dai, F.; Ye, Z. “An enhanced security framework of software defined network based on attribute-based encryption”. In: 4th International Conference on Systems and Informatics (ICSAI), 2017, pp. 965–969.

[520] Shin, S.; Gu, G. “Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)”. In: 20th IEEE International Conference on Network Protocols (ICNP), 2012, pp. 1–6.

[521] Shin, S.; Gu, G. “Attacking software-defined networks: A first feasibility study”. In: 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 165–166.

[522] Shin, S.; Song, Y.; Lee, T.; Lee, S.; Chung, J.; Porras, P.; Yegneswaran, V.; Noh, J.; Kang, B. B. “Rosemary: A robust, secure, and high-performance network operating system”. In: ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 78–89.

[523] Shin, S.; Xu, L.; Hong, S.; Gu, G. “Enhancing network security through software defined networking (SDN)”. In: 25th International Conference on Computer Communication and Networks (ICCCN), 2016, pp. 1–9.

[524] Shin, S.; Yegneswaran, V.; Porras, P.; Gu, G. “AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks”. In: ACM SIGSAC Conference on Computer: Communications Security, 2013, pp. 413–424.

[525] Shin, Y.; Son, K.; Park, H. “SDN-based security in virtualized environments for cloud computing”, *Life Science Journal*, vol. 11–7, 2014, pp. 642–647.

[526] Shirali-Shahreza, S.; Ganjali, Y. “Efficient implementation of security applications in openflow controller with flexam”. In: 21st Annual Symposium on High-Performance Interconnects (HOTI), 2013, pp. 49–54.

[527] Shirali-Shahreza, S.; Ganjali, Y. “Flexam: Flexible sampling extension for monitoring and security applications in openflow”. In: 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 167–168.

[528] Shoeb, A.; Chithralekha, T. “Resource management of switches and controller during saturation time to avoid ddos in SDN”. In: IEEE International Conference on Engineering and Technology (ICETECH), 2016, pp. 152–157.

[529] Shuangyu, H.; Jianwei, L.; Jian, M.; Jie, C. “Hierarchical solution for access control and authentication in software defined networks”. In: *Network and System Security*, Springer International Publishing, 2014, *Lecture Notes in Computer Science*, vol. 8792, pp. 70–81.

[530] Shukhman, A.; Polezhaev, P.; Ushakov, Y.; Legashev, L.; Tarasov, V.; Bakhareva, N. “Development of network security tools for enterprise software-defined networks”. In: 8th International Conference on Security of Information and Networks, 2015, pp. 224–228.

[531] Shukla, A.; Schmid, S.; Feldmann, A.; Ludwig, A.; Dudycz, S.; Schuetze, A. “Towards transiently secure updates in asynchronous SDNs”. In: ACM SIGCOMM Conference, 2016, pp. 597–598.

[532] Silva, H.; Holanda, R.; Nogueira, M.; Santos, A. “A cross-layer and adaptive scheme for balancing performance and security on wmn data routing”. In: IEEE Global Telecommunications Conference GLOBECOM, 2010, pp. 1–5.

[533] Singh, S.; Khan, R. A.; Agrawal, A. “Prevention mechanism for infrastructure based denial-of-service attack over software defined network”. In: International Conference on Computing, Communication Automation, 2015, pp. 348–353.

[534] Siniarski, B.; Olariu, C.; Perry, P.; Parsons, T.; Murphy, J. “Real-time monitoring of SDN networks using non-invasive cloud-based logging platforms”. In: IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016, pp. 1–6.

[535] Sirisutthidecha, S.; Maichalernnukul, K. “Reliable virtual channels over vpn for cloud”. In: 8th International C* Conference on Computer Science & Software Engineering, 2015, pp. 133–137.

[536] Skowyra, R.; Bahargam, S.; Bestavros, A. “Software-defined IDS for securing embedded mobile devices”. In: IEEE High Performance Extreme Computing Conference, HPEC 2013, 2013, pp. 1–7.

[537] Skowyra, R.; Bauer, K.; Dedhia, V.; Okhravi, H. “Have no phear: Networks without identifiers”. In: ACM Workshop on Moving Target Defense, 2016, pp. 3–14.

[538] Skowyra, R. W.; Lapets, A.; Bestavros, A.; Kfoury, A. “Verifiably-safe software-defined networks for cps”. In: 2Nd ACM International Conference on High Confidence Networked Systems, 2013, pp. 101–110.

[539] Smith, M.; Dvorkin, M.; Y. Laribi, V. P.; Garg, P.; Weidenbacher, N. “OpFlex Control Protocol”, 2016. Disponível em https://tools.ietf.org/html/draft-smith-opflex-03. Acesso em 01/06/2018.

[540] Smith-perrone, J.; Sims, J. “Securing cloud, SDN and large data network environments from emerging ddos attacks”. In: 7th International Conference on Cloud Computing, Data Science Engineering - Confluence, 2017, pp. 466–469.

[541] Smyth, D.; McSweeney, S.; O’Shea, D.; Cionca, V. “Detecting link fabrication attacks in software-defined networks”. In: 26th International Conference on Computer Communication and Networks (ICCCN), 2017, pp. 1–8.

[542] Son, S.; Shin, S.; Yegneswaran, V.; Porras, P.; Gu, G. “Model checking invariant security properties in openflow”. In: IEEE International Conference on Communications (ICC), 2013, pp. 1974–1979.

[543] Sonba, A.; Abdalkreim, H. “Performance comparison of the state of the art openflow controllers”, Master’s Thesis, School of Information Science, Computer and Electrical Engineering, Halmstad University, 2014.

[544] Sonchack, J.; Aviv, A. J.; Keller, E. “Timing SDN control planes to infer network configurations”. In: ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, 2016, pp. 19–22.

[545] Sonchack, J.; Dubey, A.; Aviv, A. J.; Smith, J. M.; Keller, E. “Timing-based reconnaissance and defense in software-defined networks”. In: 32Nd Annual Conference on Computer Security Applications, 2016, pp. 89–100.

[546] Song, H. “Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane”. In: 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 127–132.

[547] Song, S.; Gebre-Amlak, H.; Banala, G.; Choi, B. Y.; Park, H.; Choi, T.; Zhu, H. “Netaware: Network architecture-aware reliability management schemes for softwareized network systems”. In: IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–5.

[548] Song, S.; Hong, S.; Guan, X.; Choi, B.-Y.; Choi, C. “Neod: Network embedded on-line disaster management framework for software defined networking”. In: IFIP/IEEE International Symposium on Integrated Network Management, 2013, pp. 492–498.

[549] Spirent. “Openflow performance testing white paper”, 2015. Disponível em https://www.spirent.com/. Acesso em 01/06/2018.

[550] Stabler, G.; Rosen, A.; Goasguen, S.; Wang, K.-C. “Elastic ip and security groups implementation using openflow”. In: 6th International Workshop on Virtualization Technologies in Distributed Computing Date, 2012, pp. 53–60.

[551] Stancu, A.; Avram, A.; Skorupski, M.; Vulpe, A.; Halunga, S. “Enabling SDN application development using a netconf mediator layer simulator”. In: 9th International Conference on Ubiquitous and Future Networks (ICUFN), 2017, pp. 658–663.

[552] Steichen, M.; Hommes, S.; State, R. “Chainguard - a firewall for blockchain applications using SDN with openflow”. In: Principles, Systems and Applications of IP Telecommunications (IPTComm), 2017, pp. 1–8.

[553] Swapna, A. I.; Huda, M. R.; Aion, M. K. “Comparative security analysis of software defined wireless networking (sdwn)-bgp and netconf protocols”. In: 19th International Conference on Computer and Information Technology (ICCIT), 2016, pp. 282–287.

[554] Tammana, P.; Agarwal, R.; Lee, M. “Distributed network monitoring and debugging with switchpointer”. In: USENIX Symposium on Networked Systems Design and Implementation, 2018, pp. 453–466.

[555] Tang, T. A.; Mhamdi, L.; McLernon, D.; Zaidi, S. A. R.; Ghogho, M. “Deep learning approach for network intrusion detection in software defined networking”. In: International Conference on Wireless Networks and Mobile Communications (WINCOM), 2016, pp. 258–263.

[556] Tantar, E.; Palattella, M.; Avanesov, T.; Kantor, M.; Engel, T. “Cognition: A tool for reinforcing security in software defined networks”. In: *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, Springer International Publishing, 2014, *Advances in Intelligent Systems and Computing*, vol. 288, pp. 61–78.

[557] Tasch, M.; Khondoker, R.; Marx, R.; Bayarou, K. “Security analysis of security applications for software defined networks”. In: AINTEC on Asian Internet Engineering Conference, 2014, pp. 23–30.

[558] Tatang, D.; Quinkert, F.; Frank, J.; Röpke, C.; Holz, T. “SDN-guard: Protecting SDN controllers against SDN rootkits”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 297–302.

[559] Tatlicioglu, S.; Civanlar, S.; Gorkemli, B.; Lokman, E.; Balci, A. M.; Eliacik, C. B. “A security services platform for software defined networks”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016, pp. 39–43.

[560] Tavares, J.; Mamede, H. S.; Amaral, P.; Pinto, P. “Software-defined controllers: Where are we?” In: 12th Iberian Conference on Information Systems and Technologies (CISTI), 2017, pp. 1–6.

[561] Taylor, C. R.; MacFarland, D. C.; Smestad, D. R.; Shue, C. A. “Contextual, flow-based access control with scalable host-based SDN techniques”. In: 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[562] Taylor, D.; Wu, T.; Mavrogiannopoulos, N.; Perrin, T. “Using the Secure Remote Password (SRP) Protocol for TLS Authentication”, 2007. Disponível em https://tools.ietf.org/html/rfc5054. Acesso em 01/06/2018.

[563] Telecommunication, I. “Itu-t recommendation g.114”, 2003. Disponível em https://www.itu.int/rec/T-REC-G.114-200305-I. Acesso em 01/06/2018.

[564] Tesfamicael, A. D.; Liu, V.; Foo, E.; Caelli, W. “Modeling for performance and security balanced trading communication systems in the cloud”. In: IEEE 36th International Performance Computing and Communications Conference (IPCCC), 2017, pp. 1–7.

[565] Thimmaraju, K.; Schiff, L.; Schmid, S. “Outsmarting network security with SDN teleportation”. In: IEEE European Symposium on Security and Privacy (EuroS P), 2017, pp. 563–578.

[566] Toseef, U.; Pentikousis, K. “Implementation of c-bas: Certificate-based aaa for SDN experimental facilities”. In: IEEE 4th Symposium on Network Cloud Computing and Applications (NCCA), 2015, pp. 36–42.

[567] Toseef, U.; Zaalouk, A.; Rothe, T.; Broadbent, M.; Pentikousis, K. “C-bas: Certificate-based aaa for SDN experimental facilities”. In: 3rd European Workshop on Software Defined Networks (EWSDN), 2014, pp. 91–96.

[568] Tran, T. V.; Ahn, H. “A network topology-aware selectively distributed firewall control in SDN”. In: International Conference on Information and Communication Technology Convergence (ICTC), 2015, pp. 89–94.

[569] Tran, T. V.; Ahn, H. “Flowtracker: A SDN stateful firewall solution with adaptive connection tracking and minimized controller processing”. In: International Conference on Software Networking (ICSN), 2016, pp. 1–5.

[570] Tri, H. T. N.; Kim, K. “Assessing the impact of resource attack in software defined network”. In: International Conference on Information Networking (ICOIN), 2015, pp. 420–425.

[571] Tripathy, B. K.; Sethy, A. G.; Bera, P.; Rahman, M. A. “A novel secure and efficient policy management framework for software defined network”. In: IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 2016, pp. 423–430.

[572] Trung, P. V.; Huong, T. T.; Tuyen, D. V.; Duc, D. M.; Thanh, N. H.; Marshall, A. “A multi-criteria-based ddos-attack prevention solution using software defined networking”.

In: International Conference on Advanced Technologies for Communications (ATC), 2015, pp. 308–313.

[573] Tselios, C.; Politis, I.; Kotsopoulos, S. “Enhancing SDN security for iot-related deployments through blockchain”. In: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2017, pp. 303–308.

[574] Tseng, Y.; Pattaranantakul, M.; He, R.; Zhang, Z.; Naït-Abdesselam, F. “Controller dac: Securing SDN controller with dynamic access control”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–6.

[575] Tseng, Y.; Zhang, Z.; Naït-Abdesselam, F. “Srv: Switch-based rules verification in software defined networking”. In: IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 477–482.

[576] Tu, H.; Li, W.; Li, D.; Yu, J. “A scalable flow rule translation implementation for software defined security”. In: Network Operations and Management Symposium (APNOMS), 16th Asia-Pacific, 2014, pp. 1–5.

[577] Turner, S. “Using SHA2 Algorithms with Cryptographic Message Syntax”, 2010. Disponível em https://tools.ietf.org/html/rfc5754. Acesso em 01/06/2018.

[578] Turner, S.; Chen, L. “Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms”, 2011. Disponível em https://tools.ietf.org/html/rfc6151. Acesso em 01/06/2018.

[579] Ujcich, B. E.; Thakore, U.; Sanders, W. H. “Attain: An attack injection framework for software-defined networking”. In: 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2017, pp. 567–578.

[580] Van, N. T.; Bao, H.; Thinh, T. N. “An anomaly-based intrusion detection architecture integrated on openflow switch”. In: 6th International Conference on Communication and Network Security, 2016, pp. 99–103.

[581] Varga, P.; Kathareios, G.; Máté, A.; Clauberg, R.; Anghel, A.; Orosz, P.; Nagy, B.; Tóthfalusi, T.; Kovács, L.; Gusat, M. “Real-time security services for SDN-based datacenters”. In: 13th International Conference on Network and Service Management (CNSM), 2017, pp. 1–9.

[582] Veena, S.; Manju, R. “Detection and mitigation of security attacks using real time SDN analytics”. In: International conference of Electronics, Communication and Aerospace Technology (ICECA), 2017, pp. 87–93.

[583] Veeramani, S.; Rout, B.; Noor Mahammad, S. “Novel approach to secure channel using c-scan and microcontroller in openflow”. In: IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), 2013, pp. 1–4.

[584] Viega, J.; Messier, M.; Chandra, P. “Network Security with OpenSSL: Cryptography for Secure Communications”. O’Reilly Media, 2008.

[585] Visoottiviseth, V.; Lertviriyasawat, S.; Suppiyatrakoon, P.; Chitkornkitsil, P.; Yamai, N. “Reflo: Reactive firewall system with openflow and flow monitoring system”. In: IEEE Region 10 Conference, 2017, pp. 2273–2278.

[586] Vizarreta, P.; Heegaard, P.; Helvik, B.; Kellerer, W.; Machuca, C. M. “Characterization of failure dynamics in SDN controllers”. In: 9th International Workshop on Resilient Networks Design and Modeling (RNDM), 2017, pp. 1–7.

[587] Vizarreta, P.; Trivedi, K.; Helvik, B.; Heegaard, P.; Kellerer, W.; Machuca, C. M. “An empirical study of software reliability in SDN controllers”. In: 13th International Conference on Network and Service Management (CNSM), 2017, pp. 1–9.

[588] Vizváry, M.; Vykopal, J. “Future of ddos attacks mitigation in software defined networks”. In: *Monitoring and Securing Virtualized Networks and Services*, Springer Berlin Heidelberg, 2014, *Lecture Notes in Computer Science*, vol. 8508, pp. 123–127.

[589] Wang, A.; Guo, Y.; Hao, F.; Lakshman, T.; Chen, S. “Scotch: Elastically scaling up SDN control-plane using vswitch based overlay”. In: 10th ACM International on Conference on Emerging Networking Experiments and Technologies, 2014, pp. 403–414.

[590] Wang, B.; Zheng, Y.; Lou, W.; Hou, Y. “Ddos attack protection in the era of cloud computing and software-defined networking”. In: IEEE 22nd International Conference on Network Protocols (ICNP), 2014, pp. 624–629.

[591] Wang, H. “Authentic and confidential policy distribution in software defined wireless network”. In: International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, pp. 1167–1171.

[592] Wang, H.; Xu, L.; Gu, G. “Floodguard: A dos attack prevention extension in software-defined networks”. In: 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015, pp. 239–250.

[593] Wang, J.; Wang, Y.; Hu, H.; Sun, Q.; Shi, H.; Zeng, L. “Towards a security-enhanced firewall application for openflow networks”. In: *Cyberspace Safety and Security*, Springer International Publishing, 2013, *Lecture Notes in Computer Science*, vol. 8300, pp. 92–103.

[594] Wang, J.; Wen, R.; Li, J.; Yan, F.; Zhao, B.; Yu, F. “Detecting and mitigating target link-flooding attacks using SDN”, *IEEE Transactions on Dependable and Secure Computing*, 2018, pp. 1–13.

[595] Wang, L.; Li, Q.; Jiang, Y.; Wu, J. “Towards mitigating link flooding attack via incremental SDN deployment”. In: IEEE Symposium on Computers and Communication (ISCC), 2016, pp. 397–402.

[596] Wang, M.; Liu, J.; Chen, J.; liu, X.; Mao, J. “Perm-guard: Authenticating the validity of flow rules in software defined networking”. In: IEEE 2nd International Conference on Cyber Security and Cloud Computing, 2015, pp. 127–132.

[597] Wang, M.; Liu, J.; Mao, J.; Cheng, H.; Chen, J. “Nsv-guard: Constructing secure routing paths in software defined networking”. In: IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016, pp. 293–300.

[598] Wang, M.; Liu, J.; Mao, J.; Cheng, H.; Chen, J.; Qi, C. “Routeguardian: Constructing secure routing paths in software-defined networking”, *Tsinghua Science and Technology*, vol. 22–4, Aug 2017, pp. 400–412.

[599] Wang, P.; Chao, K. M.; Lin, H. C.; Lin, W. H.; Lo, C. C. “An efficient flow control approach for SDN-based network threat detection and migration using support vector machine”. In: IEEE 13th International Conference on e-Business Engineering (ICEBE), 2016, pp. 56–63.

[600] Wang, Q.; Zhao, Z.; Zhang, H. “Ddos defense mechanism based on software defined network”. In: IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 1122–1127.

[601] Wang, R.; Jia, Z.; Ju, L. “An entropy-based distributed ddos detection mechanism in software-defined networking”. In: IEEE Trustcom/BigDataSE/ISPA, 2015, pp. 310–317.

[602] Wang, R.; Zhang, Z.; Ju, L.; Jia, Z. “A novel openflow-based ddos flooding attack detection and response mechanism in software-defined networking”, *Int. J. Inf. Sec. Priv.*, vol. 9–3, Jul 2015, pp. 21–40.

[603] Wang, S.; Chavez, K. G.; Kandeepan, S. “Seco: SDN secure controller algorithm for detecting and defending denial of service attacks”. In: 5th International Conference on Information and Communication Technology (ICoIC7), 2017, pp. 1–6.

[604] Wang, T.; Chen, H. “Sguard: A lightweight SDN safe-guard architecture for dos attacks”, *China Communications*, vol. 14–6, 2017, pp. 113–125.

[605] Wang, T.; Hamdi, M.; Chen, J. “Enforcing timely network policies installation in openflow-based software defined networks”. In: IEEE International Conference on Communications (ICC), 2017, pp. 1–6.

[606] Wang, X.; Chen, M.; Xing, C. “Sdsnm: A software-defined security networking mechanism to defend against ddos attacks”. In: 9th International Conference on Frontier of Computer Science and Technology, 2015, pp. 115–121.

[607] Wang, X.; Liu, Z.; Li, J.; Yang, B.; Qi, Y. “Tualatin: Towards network security service provision in cloud datacenters”. In: 23rd International Conference on Computer Communication and Networks (ICCCN), 2014, pp. 1–8.

[608] Wang, Y.; Chen, Q.; Yi, J.; Guo, J. “U-tri: Unlinkability through random identifier for SDN network”. In: Workshop on Moving Target Defense, 2017, pp. 3–15.

[609] Wang, Y.; Guo, J.; Zhang, J.; Guan, Z. “Moving os fingerprint adaptively in SDN network”. In: 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 438–442.

[610] Wang, Y.; Zhang, Y.; Singh, V.; Lumezanu, C.; Jiang, G. “Netfuse: Short-circuiting traffic surges in the cloud”. In: IEEE International Conference on Communications (ICC), 2013, pp. 3514–3518.

[611] Wang, Z.; Hu, H.; Zhang, C. “On achieving SDN controller diversity for improved network security using coloring algorithm”. In: 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 1270–1275.

[612] Wang, Z.; Shou, G.; Hu, Y.; Guo, Z. “Research and implementation of resource scheduling mechanisms based on software defined security”. In: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2016, pp. 372–376.

[613] Wang, Z.; Tao, D.; Lin, Z. “Dynamic virtualization security service construction strategy for software defined networks”. In: 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), 2016, pp. 139–144.

[614] Wasserman, M.; Hartman, S. “Security analysis of the open networking foundation (onf) openflow switch specification”, Technical Report draft-mrw-SDNsec-openflow-analysis-00, IETF Secretariat, Fremont, CA, USA, 2013.

[615] Wei, H. C.; Tung, Y. H.; Yu, C. M. “Counteracting udp flooding attacks in SDN”. In: IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 367–371.

[616] Wei, L.; Fung, C. “Flowranger: A request prioritizing algorithm for controller dos attacks in software defined networks”. In: IEEE International Conference on Communications (ICC), 2015, pp. 5254–5259.

[617] Wen, X.; Chen, Y.; Hu, C.; Shi, C.; Wang, Y. “Towards a secure controller platform for openflow applications”. In: 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 171–172.

[618] Wen, X.; Yang, B.; Chen, Y.; Hu, C.; Wang, Y.; Liu, B.; Chen, X. “SDNshield: Reconciliating configurable application permissions for SDN app markets”. In: 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2016, pp. 121–132.

[619] Wilczewski, D. “Security considerations for equipment controllers and SDN”. In: IEEE International Telecommunications Energy Conference (INTELEC), 2016, pp. 1–5.

[620] Wrona, K.; Oudkerk, S.; Szwaczyk, S.; Amanowicz, M. “Content-based security and protected core networking with software-defined networks”, *IEEE Communications Magazine*, vol. 54–10, October 2016, pp. 138–144.

[621] Wu, H.; Li, X.; Scoglio, C.; Gruenbacher, D. “Middlebox resources management using openflow”. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2016, pp. 976–977.

[622] Wu, W.; Liu, R.; Ni, W.; Kaafar, D.; Huang, X. “Ac-prot: An access control model to improve software-defined networking security”. In: IEEE 85th Vehicular Technology Conference (VTC Spring), 2017, pp. 1–5.

[623] Xiao, P.; Li, Z.; Qi, H.; Qu, W.; Yu, H. “An efficient ddos detection with bloom filter in SDN”. In: IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 1–6.

[624] Xing, T.; Huang, D.; Xu, L.; Chung, C.-J.; Khatkar, P. “Snortflow: A openflow-based intrusion prevention system in cloud environment”. In: 2nd GENI Research and Educational Experiment Workshop, 2013, pp. 89–92.

[625] Xing, T.; Xiong, Z.; Huang, D.; Medhi, D. “SDNips: Enabling software-defined networking based intrusion prevention system in clouds”. In: 10th International Conference on Network and Service Management (CNSM), 2014, pp. 308–311.

[626] Xing, X.; Luo, T.; Li, J.; Hu, Y. “A defense mechanism against the dns amplification attack in SDN”. In: IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), 2016, pp. 28–33.

[627] Xu, A.; Li, M.; Cai, J.; Xue, N.; Zhang, J.; Liu, D.; Craig, P.; Huang, X. “Improving efficiency of authenticated openflow handshake using coprocessors”. In: 8th International Conference on Information Technology in Medicine and Education (ITME), 2016, pp. 576–580.

[628] Xu, H.; Su, J.; Zong, X.; Yan, L. “Attack identification for software-defined networking based on attack trees and extension innovation methods”. In: 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017, pp. 485–489.

[629] Xu, T.; Gao, D.; Dong, P.; Foh, C. H.; Zhang, H. “Mitigating the table-overflow attack in software-defined networking”, *IEEE Transactions on Network and Service Management*, vol. 14–4, Dec 2017, pp. 1086–1097.

[630] Xu, T.; Gao, D.; Dong, P.; Zhang, H.; Foh, C. H.; Chao, H. C. “Defending against new-flow attack in SDN-based internet of things”, *IEEE Access*, vol. 5, 2017, pp. 3431–3443.

[631] Xu, T.; Gao, D.; Dong, P.; Zheng, T.; Sun, J. “Smartsec: A smart security mechanism for the new-flow attack in software-defined networking”. In: IEEE 85th Vehicular Technology Conference (VTC Spring), 2017, pp. 1–7.

[632] Xu, X.; Hu, L. “A software defined security scheme based on SDN environment”. In: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017, pp. 504–512.

[633] Xu, Y.; Liu, Y. “Ddos attack detection under SDN context”. In: The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[634] Xue, N.; Huang, X.; Zhang, J. “S2net: A security framework for software defined intelligent building networks”. In: IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 654–661.

[635] y. Chen, K.; Junuthula, A. R.; Siddhrau, I. K.; Xu, Y.; Chao, H. J. “SDNshield: Towards more comprehensive defense against ddos attacks on SDN control plane”. In: IEEE Conference on Communications and Network Security (CNS), 2016, pp. 28–36.

[636] Yakasai, S. T.; Zheng, F. C.; Guy, C. G. “Towards policy unification for enterprise network security”. In: IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–5.

[637] Yamaguchi, S.; Nakao, A.; Oguchi, M.; Goto, A.; Yamamoto, S. “Monitoring dynamic modification of routing information in openflow networks”. In: 10th International Conference on Ubiquitous Information Management and Communication, 2016, pp. 1–7.

[638] Yamashita, S.; Tanaka, H.; Hori, Y.; Otani, M.; Watanabe, K. “Development of network user authentication system using openflow”. In: 8th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), 2013, pp. 566–569.

[639] Yan, Q.; Gong, Q.; Yu, F. R. “Effective software-defined networking controller scheduling method to mitigate ddos attacks”, *Electronics Letters*, vol. 53–7, 2017, pp. 469–471.

[640] Yan, Q.; Yu, F. R. “Distributed denial of service attacks in software-defined networking with cloud computing”, *IEEE Communications Magazine*, vol. 53–4, April 2015, pp. 52–59.

[641] Yan, Q.; Yu, F. R.; Gong, Q.; Li, J. “Software-defined networking (SDN) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges”, *IEEE Communications Surveys Tutorials*, vol. 18–1, Firstquarter 2016, pp. 602–622.

[642] Yang, H.; Riley, G. F. “Machine learning based proactive flow entry deletion for openflow”. In: IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

[643] Yang, X.; Han, B.; Sun, Z.; Huang, J. “SDN-based ddos attack detection with cross-plane collaboration and lightweight flow monitoring”. In: IEEE Global Communications Conference, 2017, pp. 1–6.

[644] Yao, G.; Bi, J.; Xiao, P. “Source address validation solution with openflow/nox architecture”. In: 19th IEEE International Conference on Network Protocols (ICNP), 2011, pp. 7–12.

[645] Yao, L.; Dong, P.; Zheng, T.; Zhang, H.; Du, X.; Guizani, M. “Network security analyzing and modeling based on petri net and attack tree for SDN”. In: International Conference on Computing, Networking and Communications (ICNC), 2016, pp. 1–5.

[646] Yoon, C.; Lee, S.; Kang, H.; Park, T.; Shin, S.; Yegneswaran, V.; Porras, P.; Gu, G. “Flow wars: Systemizing the attack surface and defenses in software-defined networks”, *IEEE/ACM Transactions on Networking*, vol. 25–6, Dec 2017, pp. 3514–3530.

[647] Yoon, C.; Shin, S.; Porras, P.; Yegneswaran, V.; Kang, H.; Fong, M.; O’Connor, B.; Vachuska, T. “A security-mode for carrier-grade SDN controllers”. In: 33rd Annual Computer Security Applications Conference, 2017, pp. 461–473.

[648] You, X.; Feng, Y.; Sakurai, K. “Packet in message based ddos attack detection in SDN network using openflow”. In: 5th International Symposium on Computing and Networking (CANDAR), 2017, pp. 522–528.

[649] Youssef, K. Y.; Kamel, H.; Hafez, A. A.; Zekry, A. H. A. “On balance between security and performance for lte wireless networks”. In: 22nd International Conference on Computer Theory and Applications (ICCTA), 2012, pp. 60–65.

[650] Yu, D.; Moore, A.; Hall, C.; Anderson, R. “Authentication for resilience: The case of SDN”. In: *Security Protocols XXI*, Springer Berlin Heidelberg, 2013, *Lecture Notes in Computer Science*, vol. 8263, pp. 39–44.

[651] Yu, J.; Wang, X.; Song, J.; Zheng, Y.; Song, H. “Forwarding programming in protocol-oblivious instruction set”. In: IEEE 22nd International Conference on Network Protocols, 2014, pp. 577–582.

[652] YuHunag, C.; MinChi, T.; YaoTing, C.; YuChieh, C.; YanRen, C. “A novel design for future on-demand service and security”. In: 12th IEEE International Conference on Communication Technology (ICCT), 2010, pp. 385–388.

[653] Yuwen, H.; Zhang, L.; Wang, Z.; Kong, Y. “Probability-based delay scheme for resisting SDN scanning”. In: 2nd IEEE International Conference on Computer and Communications (ICCC), 2016, pp. 1096–1101.

[654] Yürekten, O.; Demirci, M. “Using cyber threat intelligence in SDN security”. In: International Conference on Computer Science and Engineering (UBMK), 2017, pp. 377–382.

[655] Zaalouk, A.; Khondoker, R.; Marx, R.; Bayarou, K. “Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions”. In: Network Operations and Management Symposium (NOMS), IEEE, 2014, pp. 1–9.

[656] Zhang, C.; Hu, G.; Chen, G.; Sangaiah, A. K.; Zhang, P.; Yan, X.; Jiang, W. “Towards a SDN-based integrated architecture for mitigating ip spoofing attack”, *IEEE Access*, vol. 6, 2018, pp. 22764–22777.

[657] Zhang, K.; Qiu, X. “Cmd: A convincing mechanism for mitm detection in SDN”. In: IEEE International Conference on Consumer Electronics (ICCE), 2018, pp. 1–6.

[658] Zhang, L.; Guo, Y.; Yuwen, H.; Wang, Y. “A port hopping based dos mitigation scheme in SDN network”. In: 12th International Conference on Computational Intelligence and Security (CIS), 2016, pp. 314–317.

[659] Zhang, L.; Shou, G.; Hu, Y.; Guo, Z. “Deployment of intrusion prevention system based on software defined networking”. In: 15th IEEE International Conference on Communication Technology (ICCT), 2013, pp. 26–31.

[660] Zhang, L.; Wang, Z.; Gu, K.; Miao, F.; Guo, Y. “Transparent synchronization based port mutation scheme in SDN network”. In: 5th International Conference on Computer Science and Network Technology (ICCSNT), 2016, pp. 581–585.

[661] Zhang, L.; Wei, Q.; Gu, K.; Yuwen, H. “Path hopping based SDN network defense technology”. In: 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016, pp. 2058–2063.

[662] Zhang, M.; Bi, J.; Bai, J.; Dong, Z.; Li, Y.; Li, Z. “Ftguard: A priority-aware strategy against the flow table overflow attack in SDN”. In: SIGCOMM Posters and Demos, 2017, pp. 141–143.

[663] Zhang, M.; Hou, J.; Zhang, Z.; Shi, W.; Qin, B.; Liang, B. “Fine-grained fingerprinting threats to software-defined networks”. In: IEEE Trustcom/BigDataSE/ICESS, 2017, pp. 128–135.

[664] Zhang, P.; Chen, X.; Ge, Y.; Jin, L. “A parallel processing and synthesis structure for improving access security and efficiency in SDN environment”, *Chinese Journal of Electronics*, vol. 25–5, 2016, pp. 817–823.

[665] Zhang, S.; Jia, X.; Zhang, W. “Towards comprehensive protection for openflow controllers”. In: 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2017, pp. 82–87.

[666] Zhang, S.; Li, Q.; Wu, J.; Li, J.; Li, G. “A security mechanism for software-defined networking based communications in vehicle-to-grid”. In: IEEE Smart Energy Grid Engineering (SEGE), 2016, pp. 386–391.

[667] Zhang, T.; Bianco, A.; Giaccone, P.; Nezhad, A. P. “Dealing with misbehaving controllers in SDN networks”, *IEEE Global Communications Conference*, 2017, pp. 1–6.

[668] Zhang, Y.; Li, J.; Chen, L.; Ji, Y.; Tang, F. “A novel method against the firewall bypass threat in openflow networks”. In: 9th International Conference on Wireless Communications and Signal Processing (WCSP), 2017, pp. 1–6.

[669] Zhao, Z.; Liu, F.; Gong, D.; Chen, L.; Xiang, F.; Li, Y. “An SDN-based ip hopping communication scheme against scanning attack”. In: IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017, pp. 559–564.

[670] Zheng, J.; Li, Q.; Gu, G.; Cao, J.; Yau, D. K. Y.; Wu, J. “Realtime ddos defense using cots SDN switches via adaptive correlation analysis”, *IEEE Transactions on Information Forensics and Security*, vol. 13–7, July 2018, pp. 1838–1853.

[671] Zhong, H.; Sheng, J.; Cui, J.; Xu, Y. “Scplbs: A smart cooperative platform for load balancing and security on SDN distributed controllers”. In: 3rd IEEE International Conference on Cybernetics (CYBCONF), 2017, pp. 1–6.

[672] Zhou, H.; Wu, C.; Jiang, M.; Zhou, B.; Gao, W.; Pan, T.; Huang, M. “Evolving defense mechanism for future network security”, *IEEE Communications Magazine*, vol. 53–4, 2015, pp. 45–51.

[673] Zhou, L.; Guo, H. “Applying nfv/SDN in mitigating ddos attacks”. In: IEEE Region 10 Conference, 2017, pp. 2061–2066.

[674] Zhou, R.; Lai, Y.; Liu, Z.; Liu, J. “Study on authentication protocol of SDN trusted domain”. In: IEEE 12th International Symposium on Autonomous Decentralized Systems, 2015, pp. 281–284.

[675] Zhu, L.; Tang, X.; Shen, M.; Du, X.; Guizani, M. “Privacy-preserving ddos attack detection using cross-domain traffic in software defined networks”, *IEEE Journal on Selected Areas in Communications*, vol. 36–3, March 2018, pp. 628–643.

[676] Zhu, T.; Feng, D.; Hua, Y.; Wang, F.; Shi, Q.; Liu, J. “Mic: An efficient anonymous communication system in data center networks”. In: 45th International Conference on Parallel Processing (ICPP), 2016, pp. 11–20.

[677] Zhu, T.; Feng, D.; Wang, F.; Hua, Y.; Shi, Q.; Liu, J.; Cheng, Y.; Wan, Y. “Efficient anonymous communication in SDN-based data center networks”, *IEEE/ACM Transactions on Networking*, vol. 25–6, Dec 2017, pp. 3767–3780.

[678] Zou, D.; Lu, Y.; Yuan, B.; Chen, H.; Jin, H. “A fine-grained multi-tenant permission management framework for SDN and nfv”, *IEEE Access*, vol. 6, 2018, pp. 25562–25572.

APÊNDICE A – EXPERIMENTOS ENVOLVENDO A CRIPTOGRAFIA SIMÉTRICA EM AMBIENTES DE NUVEM DISTRIBUÍDA.

Nestes experimentos, foram realizados estudos para avaliar o custo de uma comunicação segura em um ambiente de nuvem distribuída. Para isso, foram observados os impactos do uso de criptografia simétrica para proteger os dados trafegados nesse tipo de comunicação. Para mensurar o impacto, foram capturados o uso de CPU e largura de banda na comunicação entre duas máquinas virtuais, através de canais de comunicação utilizando VPN [535], em um ambiente controlado. O impacto de prover o uso de criptografia foi mensurado através do tráfego de 1GB de dados por dois canais de comunicação, o primeiro criptografado e o segundo foi configurado para trafegar dados sem criptografia.

Para avaliar a comunicação, foram utilizadas duas máquinas virtuais em um ambiente composto por dois processadores Intel Xeon E5530 2.4 GHz (4 núcleos cada) e 16GB de RAM. A virtualização ocorreu via Xen hypervisor v6.2 [65]. Durante essas comunicações, foram capturados o tempo de uso de CPU e o tempo total de transferência de dados. Essas informações foram utilizadas para calcular a largura de banda e o uso de CPU pelo processo de criptografia. O experimento foi replicado para os algoritmos de criptografia simétrica providos pelo OpenSSL (RC2, DES-EDE, BF, CAST5, AES e CAMELLIA, com chave de 128 bits, DES-EDE3, DESX, AES, CAMELLIA, com chave de 192 bits, e AES e CAMELLIA, com chave de 256 bits) [584].

Entre os resultados obtidos nesse estudo, como representado na Figura A.1, o algoritmo que apresentou o menor overhead, com um tamanho de chave de 128 bits (AES, no modo de operação CBC), apresentou um uso de CPU de aproximadamente 56% e uma largura de banda em torno de 200Mbps. Para critérios de comparação, o algoritmo AES no modo CBC foi 500% mais eficiente do que o algoritmo AES no modo CFB1.

Nos experimentos que usam 192 bits de tamanho de chave, com os resultados apresentados na Figura A.2, o algoritmo AES no modo CBC apresentou a melhor taxa de largura de banda por uso de CPU. Entretanto, esse algoritmo no modo CBF1 apresentou a pior taxa. Isso demonstra que somente a troca de um modo de operação de um algoritmo de criptografia pode impactar significativamente no desempenho de uma comunicação segura.

Os resultados do último grupo de experimentos (chave de 256 bits, representados na Figura A.3) demonstram que o algoritmo AES em modo CBC continua sendo a opção com a melhor taxa de desempenho, enquanto que AES nos modos CFB1 e CFB8 apresentam a pior taxa desempenho, levando em consideração a razão entre largura de banca e uso de CPU. Com isso, pode-se concluir que não só a opção de utilizar uma comunicação segura pode impactar severamente no desempenho do ambiente, assim como a escolha dos mecanismos de segurança também influem nesse desempenho.

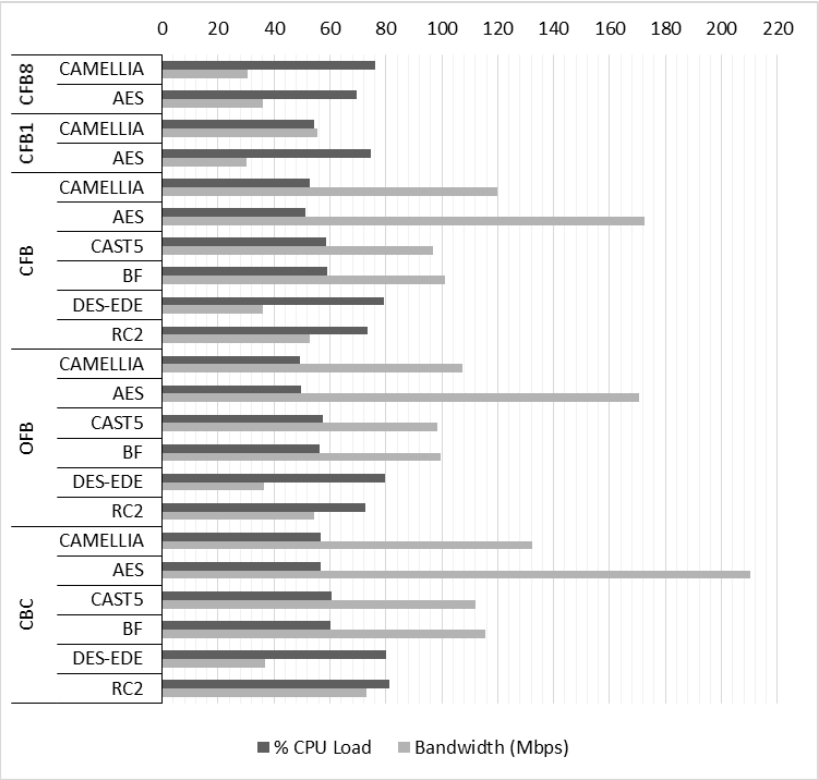

Figura A.1 – Uso de CPU e Largura de banda dos algoritmos de criptografia simétrica com chave de 128 bits

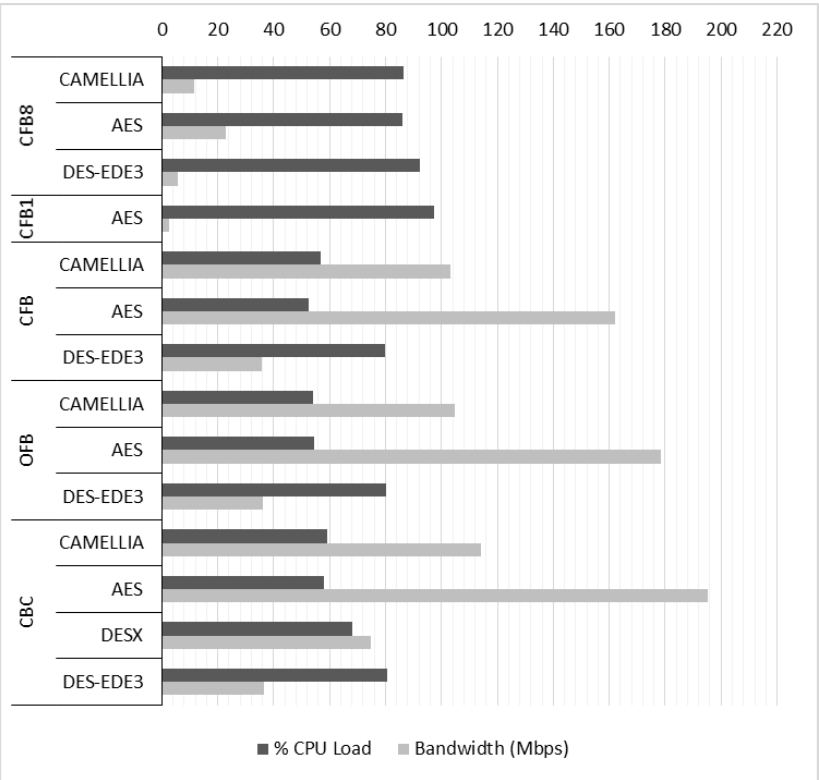

Figura A.2 – Uso de CPU e Largura de banda dos algoritmos de criptografia simétrica com chave de 192 bits

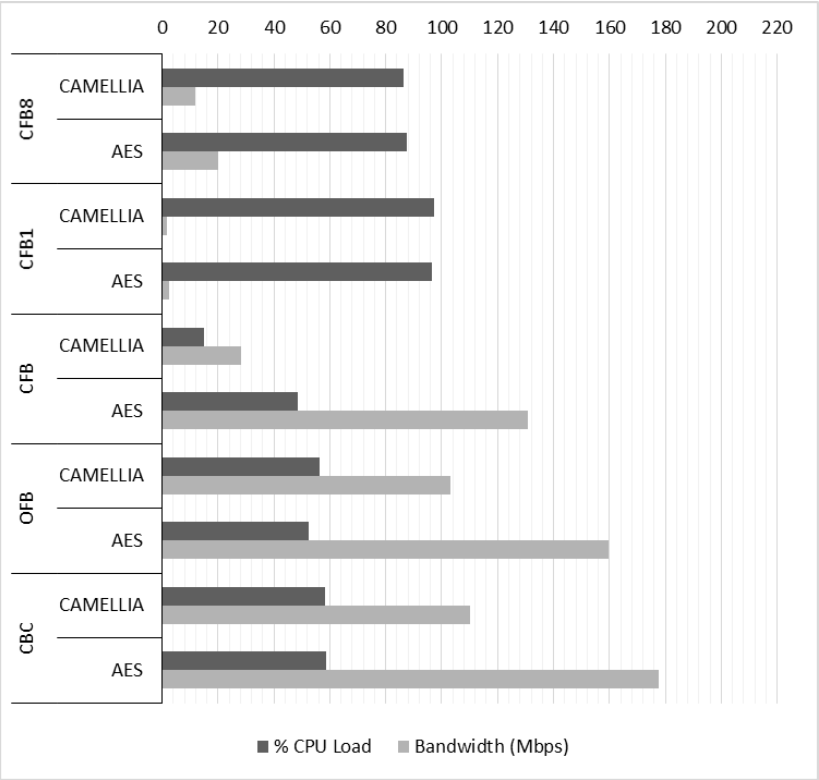

Figura A.3 – Uso de CPU e Largura de banda dos algoritmos de criptografia simétrica com chave de 256 bits

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br