UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

INE5433 – TRABALHO DE COMCLUSÃO DE CURSO II

Integração entre Rede Definida por Software e Sistema de Detecção de Intrusão para Mitigação de Ataques

Clailton Francisco

Florianópolis - SC

2019/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

Integração entre Rede Definida por Software e Sistema de Detecção de Intrusão para Mitigação de Ataques

Clailton Francisco

13200634

Proposta de trabalho de conclusão de curso a ser apresentado como requisito parcial para a obtenção do grau de Bacharel em Ciências da Computação pela Universidade Federal de Santa Catarina - UFSC

Orientadora:

Profª. Dra. Carla Merkle Westphall

Florianópolis – SC

2019/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Clailton Francisco

Integração ente Rede Definida por Software e Sistema de Detecção de Intrusão para Mitigação de Ataques

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Orientadora: Profª. Dra. Carla Merkle Westphall

Banca Examinadora:

_____

Profª. Dra. Carla Merkle Westphall

Universidade Federal de Santa Catarina

_____

Profº. Dr. Carlos Becker Westphall

Universidade Federal de Santa Catarina

_____

Profº. Dr. Roberto Willrich

Universidade Federal de Santa Catarina

Agradecimentos

Agradeço primeiramente a toda minha família pelo apoio constante. Aos meus pais Walmor e Clarice e meus irmãos Maicon e Larissa por tudo o que me ensinaram durante a vida, pelo incentivo em todos os momentos e todo o apoio que me deram possibilitando a realização desse sonho. Aos meus amigos, que sempre me acompanharam por essa jornada de aprendizado, de desânimo e cansaço, não me deixando jamais desistir, com todo apoio e determinação. Graças a eles eu tinha certeza que não estava sozinho nessa caminhada.

Agradeço minha orientadora, Profª Dra. Carla Merkle Wesphall, pelos conselhos, indicando o caminho correto a ser seguido durante este trabalho. Não deixando de agradecer a todos os professores da UFSC, que contribuíram com o ensinamento necessário para chegar até aqui.

A todos, meus eternos e sinceros agradecimentos.

Resumo

Tendo em vista o crescimento do uso da rede de computadores, os desafios para manter este ambiente estável e seguro estão cada vez maiores. O surgimento do conceito de Redes Definidas por Software, veio com o objetivo diminuir a falta de integração entre os ativos de rede com o do plano de controle, fornecendo uma abstração em três áreas da rede: estado distribuído, encaminhamento e configuração.

Com as vantagens que uma Rede Definida por Software (SDN) proporcionam é possível incorporar novas e velhas técnicas para minimizar os riscos de ataques e aumentar a segurança da rede. SDN torna o gerenciamento e manutenção da rede mais eficiente e com menor custo operacional. Proporciona também uma melhor qualidade de serviço e minimizando os pontos de falha da rede.

Os sistemas de prevenção de detecção de ataques são essenciais para a segurança da rede de computadores. Monitorar o tráfego da rede em tempo real em busca de intrusos para garantir uma rede confiável é um dos seus papéis. E diante da integração entre uma Rede Definida por Software e os Sistemas de Detecção e Prevenção de ataques, este estudo apresenta uma forma de mitigação de ataque usando os avanços da Rede Definida por Software, com o protocolo OpenFlow, tendo como base a integração de tecnologias já estabelecidas na arquitetura de rede convencional.

Palavras-chave: SND, OpenFlow, Segurança em redes SDN, Mitigação de Ataque, IDS.

Abstract

In view of the growing use of the computer network, the challenges to maintaining this stable and secure environment are growing. The emergence of the concept of Software Defined Networks (SDN) came with the objective of reducing the lack of integration between the network assets and the control plan, providing an abstraction in three areas of the network: distributed state, routing and configuration. With the advantages that a SDN provides, it is possible to incorporate new and old techniques to minimize the risk of attacks and increase network security. SDN makes network management and maintenance more efficient and cost effective. It also provides better quality of service and minimizes network failure points. Attack detection prevention systems are essential for the security of the computer network. Monitoring real-time network traffic in search of intruders to secure a trusted network is one of their roles, and in in front of the integration between a SDN and Attack Detection and Prevention Systems, this study presents a form of attack mitigation using Software Defined Network advances with the OpenFlow protocol, based on the integration of technologies already established in the conventional network architecture.

Keywords: SND, OpenFlow, SDN Network Security, Attack Mitigation, IDS.

Lista de Ilustrações

Lista de Abreviaturas

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BGP	Border Gateway Protocol
CAIS	Centro de Atendimento a Incidentes de Segurança
CLI	Command Line Interface
CPU	Central Processing Unit
DoS	Denial of Service
DDoS	Distributed Denial of Service
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IP	Internet Protocol
IPS	Intrusion Prevention System
MAC	Media Access Control
NIDS	Network Intrusion Detection System
NOS	Network Operating System
ONF	Open Networking
OSPF	Open Shortest Path First
OVSDB	Open vSwitch Database
QoS	Quality of Service
REST	Representational State Transfer
RFC	Request for Comments
SDN	Software Defined Networking
SSH	Secure Shell
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol / Internet Protocol

SUMÁRIO

1 Introdução

Com a atual expansão da computação em rede a comunidade da área se encontra em uma complexa situação. O crescimento da computação proporcionou grandes avanços e está levando a infraestrutura atual ao seu limite, fazendo com que as ferramentas tradicionalmente usadas não sejam mais suficientes para suprir o alto nível de complexidade exigida [TIWARI, VARUN; PAREKH RUSHIT; PATEL, VISHAL, 2014].

Com o advento do conceito de Redes Definidas por Software (SND - Software Defined Network), surgiu a possibilidade de programar uma rede semelhante a um computador. O OpenFlow, ou qualquer API (Application Programming Interface) que forneça uma camada de abstração da rede física para o elemento de controle, permite que a rede seja configurada ou manipulada através de software, o que abre possibilidade para maiores inovações [FERNANDES, EDER LEAO; ROTHENBERG, CHRISTIAN ESTEVE, 2014]. Isso faz com que as ferramentas atuais para gerencia de recursos de infraestrutura, que apresentam rigidez na política de mudanças de alto nível tenham um alto nível de complexidade para sua manutenção e sejam progressivamente substituídas.

A SDN fornece abstração em três áreas da rede: estado distribuído, encaminhamento e configuração [MCKEOWN, N. et al. 2008]. Com o OpenFlow/SDN, os usuários podem personalizar as redes de acordo com as necessidades locais, eliminar ferramentas desnecessárias e criar redes virtuais e isoladas. Com isso pode-se aumentar o ritmo da inovação através de software, em vez de hardware.

Em grande escala, a gerência dos recursos da rede de forma independente se torna cada vez mais complicada. Além disso, plataformas legadas, que já não possuem suporte e não podem ser atualizadas quando brechas em sua implementação são expostas, geraram a necessidade de uma forma externa de identificação de ataque. Nesse contexto, foram pensadas em ferramentas que identifiquem tentativas de ataque a estes sistemas através de padrões de comportamento de rede observados. Essas ferramentas são chamadas de IDS

(Sistemas de Detecção de Intrusão), quem estão aptos a monitorar a rede com o intuito de detectar ações intrusivas na rede.

Com isso, trabalho tem por objetivo prover uma demonstração empírica relacionada a segurança envolvendo a tecnologia SDN e o protocolo OpenFlow com técnologia já existente na arquitetura de rede convencional e demonstrar sua eficiência em detectar ataques e atuar para sua mitigação.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é prover um estudo sobre a arquitetura SDN com o protocolo OpenFlow, para mitigar determinados tipos ataques de rede já conhecidos usando um sistema de detecção de intrusão e demonstrar a eficiência em detectar e contornar o problema com integração dessas tecnologias.

Serão feitos testes em um ambiente simulado com o intuito de gerar ataques a uma rede e analisar as formas de mitigar os problemas com as integrações disponíveis em uma SDN e um Sistema de Detecção de Intrusão. Gerar um relatório com os resultados apresentando as formas usadas para testar os ataques e a mitigação.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho que podem ser citados são:

- Simular um ambiente de rede SDN;
- Configurar um IDS (Sistema de Detecção de Intrusão);
- Integrar a rede SDN ao IDS;
- Gerar tráfego e analisar a viabilidade simulando ataques e explorando possíveis formas de mitigar um determinado ataque a rede;
- Documentar os resultados gerados.

1.2 Escopo do Trabalho

Será feita uma revisão bibliográfica sobre as tecnologias e principais conceitos envolvidos no projeto. Serão apresentadas características do protocolo OpenFlow e ferramentas para simulação de ambiente de rede.

Na parte prática, será implementado um ambiente de máquinas virtuais com alguns equipamentos de rede. Pretende-se simular formas de ataque a rede para que se possa avaliar o desempenho das tecnologias envolvidas e documentar os resultados obtidos.

1.3 Estudo de Caso Exploratório

O trabalho será desenvolvido com o auxílio de maquinas virtualizadas para simulação dos equipamentos de rede e software especializados em gerar tráfego de rede. O ambiente de teste será documentado e o tráfego será feito de forma controlada e padronizada com o intuito de usar o que já é de conhecimento em relação aos tipos de ataque a uma rede de computadores.

Inicialmente, serão estudadas e documentadas as funcionalidades disponibilizadas pelo sistema operacional a ser utilizado. Em seguida, os equipamentos serão instalados e configurados em ambiente de testes. E por fim, será apresentado os resultados obtidos.

1.4 Trabalhos Correlatos

Existem vários trabalhos na linha de mitigação de ataque com IDS. [Dalpissol, Vicentini, Santin, 2018] propõe a integração entre o controlador POX e o IDS Snort para monitorar o tráfego de rede. O estudo se baseia em criar uma rede virtualizada via mininet, onde o controlador da rede se comunica com o IDS Snort presente em outra máquina fora da rede do experimento, proporcionando a possibilidade de testar a integração entre o controlador e IDS em redes diferentes. A

forma mitigação dos ataques foram usados como base para o estudo apresentado nesse trabalho.

Já [Manso, Moura, Serrão, 2019] descrevem a mitigação de ataques DDoS através da integração entre SDN e o IDS Snort, e tem como objetivo final bloquear apenas o tráfego malicioso, fazendo com que o tráfego normal não seja afetado. O resultado é obtido com a criação de algoritmos de análise de tráfego usando como base dados de ataques já homologados em outros trabalhos e disponíveis ao público. O trabalho demonstrou a eficiência da SDN ao integrar novos algoritmos de análise de rede com o controlador SDN.

[Fernandes 2016] propões integrar IDSs ao ambiente das SDNs. Especificamente, esse estudo se baseia no uso no protocolo OpenFlow. Propondo uma nova forma de implementação deste tipo de sistema, chamado de IDSFlow, que busca apresentar uma solução para integrar um controlador SDN, com um ou mais IPS/NIDS em uma rede fisicamente distribuída. A ferramenta IDSFlow tem como objetivo minimizar os problemas de integração entre um IDS e o controlador SDN. O IDSFlow leva automaticamente o tráfego de interesse até os pontos de análise, diferentemente dos esquemas atuais, no qual o IDS é colocado como uma ponte por onde já passa o trafego de interesse. Para validação dessa ferramenta foram usados algoritmos que mostraram a capacidade de integração sem onerar o desempenho da rede.

1.5 Estrutura dos capítulos

Este trabalho está organizado da seguinte forma:

O capítulo 2 descreve Redes Definidas por Software. Também trará conceitos sobre a arquitetura dessa tecnologia, bem como suas principais características.

O capítulo 3 descreve de forma sucinta o protocolo openFlow, o switch virtual usado e o controlador usado para a rede.

No capítulo 4 são abordados alguns aspectos de segurança e o conceito de Sistemas de Detecção de Intrusão (IDS).

Já no capitulo 5, são apresentados as tecnologias envolvidas no projeto e as ferramentas envolvidas e a forma de integração entre uma SDN e um IDS. Por fim, o ambiente de simulação juntamente com as formas de ataque usadas e os resultados obtidos são relatados.

No capítulo 7 é apresentada a conclusão deste estudo, bem como sugestões de estudos futuros.

2 Redes Definidas Por Software (SDN)

A infraestrutura de rede se encontra em uma situação complexa: com o sucesso e avanço da área de rede, existem muitas aplicações que funcionam sobre uma estrutura de switches, roteadores, firewalls, servidores e etc., que necessitam de características como: baixa latência, flexibilidade, confiabilidade, segurança e escalabilidade. Entretanto, apesar do aumento das capacidades de processamento dos equipamentos de comunicação e melhoria dos canais de comunicação, a rede do ponto de vista arquitetural, não apresentou tantas mudanças [KULKARNI, 2017].

Desde o momento em que as redes passaram a empregar equipamentos como switches e roteadores para encaminhar pacotes, a estrutura pouco se alterou [CENTENO, 2016]. Fabricantes criam equipamentos com arquiteturas proprietárias e fechadas, fazendo com que o software esteja encapsulado no hardware, tornando-os uma única "entidade". A forma como a rede evoluiu, criou como base um emaranhado de protocolos e camadas de equipamentos, que mesmo permitindo a comunicação da forma como conhecemos hoje, ergueu-se uma enorme barreira para inovações e experimentações necessárias à área de redes de computadores. Qualquer inovação representa um processo de alta complexidade.

Segundo Nick Mckeown, pouca tecnologia é transferida da academia para o mercado, e em geral, a partir de sua concepção, uma inovação leva até 10 anos para entrar em produção [McKeown, 2009].

Essas características fechadas, levantou o questionamento se a arquitetura de redes de computadores de modo geral atingiu um nível de amadurecimento que as tornaram pouco flexíveis. O jargão usado em muitos casos é que a Internet está calcificada, fazendo referência a pouca flexibilidade para mudanças na arquitetura de rede [KULKARNI, 2017].

2.1 Visão Geral da Tecnologia SDN

Essa estrutura pouco maleável das redes tradicionais pode ser percebida pela caracterização do plano de controle e de dados serem unificados, um sistema distribuído de controle dos equipamentos da rede e um uso de uma única

infraestrutura de rede física. Em geral, os algoritmos de roteamento são conjuntos de regras implementadas em hardware dedicado. Estes são projetados para executar encaminhamento de pacotes de forma especifica. Em uma rede convencional, após a recepção de um pacote por um dispositivo de roteamento, ele usa um conjunto de regras incorporadas em sua versão de firmware e encontra o dispositivo destino, bem como o caminho de roteamento para esse pacote [Hu et al. 2014].

A tecnologia de Redes Definidas por Software (Software Defined Networks, ou SDN) é uma arquitetura dinâmica, gerenciável e adaptável. Ou seja, SDN é baseado no conceito de separação entre uma entidade controlada e uma entidade controladora [McKeown, 2009].

O controlador manipula o controle da entidade através de uma interface. Essas Interfaces são principalmente APIs invocadas através de alguma biblioteca ou chamada de sistema. No entanto, interfaces podem ser estendidas através de alguma definição de protocolo. Isso refere-se ao fato de que a rede tem a capacidade de prover uma interface para que uma aplicação configure seus equipamentos de rede, de forma dinâmica, onde é possível mudar o comportamento de encaminhamento de pacote da mesma [Hakiri et al. 2014].

A ONF (Open Networking Foundation) é uma organização sem fins lucrativos dedicada ao desenvolvimento, padronização e comercialização de SDN, que descreve redes definidas por software como: “Uma arquitetura de rede, onde os planos de controle e de dados são separados, o controle da rede é logicamente centralizado, e a camada de infraestrutura é abstraída das aplicações” [ONF, 2015].

A arquitetura SDN, conforme a ONF, possui ainda algumas características importantes:

- Diretamente programável: Ou seja, o controle de rede é diretamente programável porque é desacoplado das funções de encaminhamento;
- Ágil: O controle de encaminhamento permite que os administradores ajustem dinamicamente o fluxo de trafego em toda a rede para atender às necessidades em constantes mudanças;

- Gerencia centralizada: A inteligência de rede é (logicamente) centralizada em controladores SDN baseados em software que mantém uma visão global da rede;
- Configurável: A SDN permite que os gerentes de rede configurem, gerenciem, protejam e otimizem os recursos de rede rapidamente por meio de programas SDN dinâmicos e automatizados. [ONF, 2015].

Essas características permitem que o controle da rede se torne diretamente programável e a infraestrutura subjacente seja abstraída para aplicativos e serviços de rede.

Além das características citadas acima o modelo SDN possui três camadas: o plano de dados, o plano de controle e o plano de aplicação/gestão [ONF, 2015]. Caracterizando um conceito fundamental da arquitetura SDN, que é a separação do plano de controle e do plano de dados [McKeown 2009].

Os Switches de rede tornam-se dispositivos de encaminhamento simples e a lógica de controle é implementada em um controlador lógico centralizado. A figura 1 apresenta de forma geral uma visão de como é apresentada à arquitetura SDN, sendo a sua divisão em camadas de aplicação, controle, dados e infraestrutura. Essas divisões serão abordadas de forma minuciosa no capitulo 2, na sessão 2.3 [ONF, 2015].

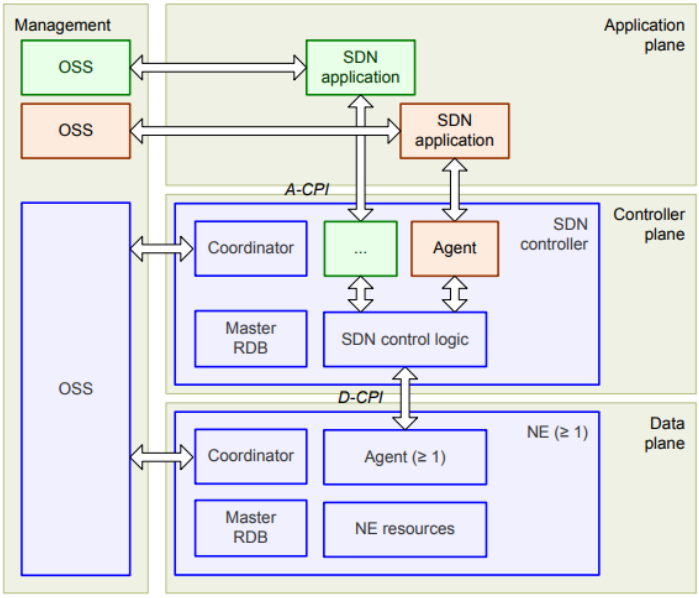

Figura 1 - Visão geral da arquitetura SND, com plano de dados e físico. [Fernandes 2014]

2.2 Arquitetura de Rede Definida por Software

Redes Definidas por Software foi uma iniciativa promissora para a arquitetura de Internet, trazendo uma outra alternativa para o projeto de redes [McKeown 2009]. A ideia mais importante das SDN encontra-se na separação dos planos de dados e de controle em uma interface uniforme, independente de fornecedor, para o mecanismo de encaminhamento (ex.: OpenFlow) [McKeown et al. 2008].

Lantz explica que em uma rede definida por software o plano de controle (ou “sistema operacional de rede”) é separado do plano de dados. Comumente, o sistema operacional de rede observa e controla o estado de toda a rede a partir de um ponto central, oferecendo recursos como: protocolos de roteamento, controle de acesso, virtualização de rede, gestão de energia e também prototipação de novos protocolos [Lantz et al. 2010].

A principal característica das SDNs é que as funcionalidades da rede podem ser facilmente alteradas ou mesmo definidas após a rede ter sido implantada. Novas funcionalidades podem ser adicionadas, sem a necessidade de se modificar o

hardware, permitindo que o comportamento da rede evolua na mesma velocidade que o software [Fernandes 2014].

Contudo, pode-se dizer que o aspecto mais importante de uma rede SDN é o alto grau de flexibilidade, em vez de aguardar um update de software ou um novo produto do fabricante para que uma alteração na arquitetura seja efetuada é possível implementar tal alteração sem a necessidade de aguardar o fabricante [Kreutz, 2015].

É também importante ressaltar o grau de programabilidade da rede SDN, uma vez que os projetistas de redes têm a possibilidade de configurar e escrever suas aplicações por meio de uma camada de abstração de alto nível. Além do mais, isso permite que a cada alteração na lógica de encaminhamento da rede ou adição de uma nova funcionalidade, o código da aplicação seja simplesmente atualizado. Como a camada de aplicação independe do hardware, a evolução desta pode ser dinâmica e constante [BRANDT et al., 2014]. Assim, toda a flexibilidade que uma linguagem de programação pode oferecer, passa a estar disponível para ser utilizada em uma aplicação SDN.

Na Figura 2 são apresentados os Planos de Controle (Control Plane), Plano de Encaminhamento (Forwarding Plane) ou Plano de Dados (Data Plane), Planos de Gerenciamento (Management Plane)/Plano de Aplicação (Application Plane) e o Plano Operacional (Operational), e como cada um se comunica na arquitetura. Também é possível observar as interfaces de comunicação denominadas Extremidade-Sul (SoudthBound). Este trabalho também irá abordar as interfaces de comunicação Extremidade-Norte (Northbound), Extremidade-Oeste(Westbound) e Extremidade-Leste (Eastbound). Esses componentes são necessários para a comunicação entre os componentes de rede em cada plano [RFC 7426].

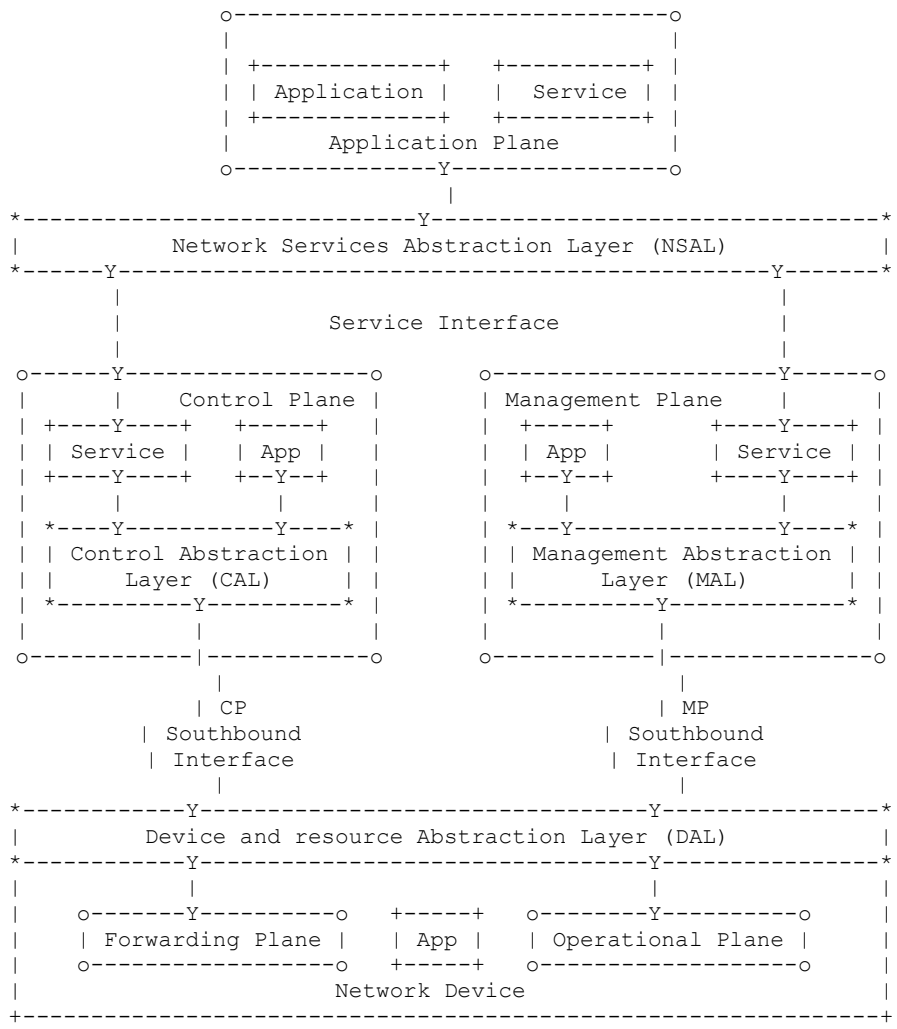

Figura 2 - Arquitetura definida pela RFC 7426.
[RFC 7426]

2.3 Plano de dados

O plano de dados é composto por comutadores e roteadores, que tem o papel encaminhar os dados na rede. Contudo, diferentemente de roteadores e comutadores convencionais, eles não carregam implementações de protocolos e problemas complexos. Assim, esses comutadores não são capazes de interpretar regras para qualquer protocolo de camada superior, como ICMP, TCP ou UDP e

regras para os pacotes IP. A "inteligência" está concentrada no plano de controle. Esse plano oferece abstrações e funcionalidades básicas para a implementação das políticas e propriedades desejadas pela aplicação/entidade gestora da rede [Ortiz, 2018]. Com a retirada do plano de controle dos dispositivos, pode-se modelar e moldar dinamicamente a rede, de acordo com as necessidades do gestor, permitindo que a gerência da rede seja feita de forma muito mais fácil e eficiente.

Em um dispositivo configurado para operar com algum protocolo SDN, como por exemplo o OpenFlow, apenas estão instaladas tabelas de fluxos, que decidem as ações que serão tomadas com cada pacote. Nesse contexto, um fluxo é um conjunto de pacotes com valores semelhantes. Essas tabelas são compostas de regras de compatibilidade, isto é, regras que comparam valores de campos do cabeçalho do pacote bit a bit, como o destino, protocolo e número de porta, com algum valor esperado, e ações a serem tomadas com o pacote caso a regra seja válida [Feamster et al. 2014]. Além disso, as tabelas de fluxo possuem contadores, que mantêm estatísticas de utilização e permitem a remoção de fluxos inativos. Ações que podem ser realizadas são:

- Encaminhamento de pacotes para uma porta de saída;
- Encapsular e encaminhar o pacote para o controlador;
- Descartar pacotes;
- Enviar pacotes para a próxima tabela de fluxo ou para alguma tabela especial;
- Enviar pacotes para a sequência de processamento normal do equipamento nas camadas 2 ou 3.

Essa última ação permite que um equipamento compatível com o protocolo OpenFlow também possa realizar o processamento normal da internet. A prioridade entre as regras segue a ordem das linhas de uma tabela e, posteriormente, a ordem das tabelas em uma sequência de processamento.

2.4 Plano de Controle

O plano de controle concentra as tomadas de decisão da rede definida por software. Ele é composto pelo sistema operacional da rede (Network Operating

System - NOS), que oferece uma abstração logicamente centralizada. Esse plano fornece abstrações, serviços essenciais e APIs para os gestores e desenvolvedores da rede. Essa interação se dá por meio das interfaces das extremidades oeste (westbound) ou leste (eastbound) [Guo et al. 2016].

Isso significa que um desenvolvedor não precisa mais saber de todos os detalhes de uma transmissão de pacotes para definir uma política de rede, facilitando o seu desenvolvimento e diminuindo as chances de erro. O controlador não necessariamente é um equipamento específico implantado na rede. Ele é um componente de software da SDN, cujas aplicações de controle são implementadas por APIs e é de fundamental importância [Kreutz, 2015].

Na figura 3 é apresentado uma política de rede baseada em SDN.

SrcAddr:192.168.0.0/16 \ (SrcAddr:10.0.0.1 V DstPort:7000) → {Switch 2}

Figura 3 - Política de roteamento.

Essa regra define que qualquer pacote originado da subrede 192.168.0.0/16 deve ser encaminhado para o comutador (Switch) 2. Exceto os que possuem o IP de origem 10.0.0.1 ou que sejam destinados à porta 7000.

Os NOS do plano de controle tem como principais funções a análise do estado da rede, fornecendo informações sobre a topologia, descoberta de dispositivos conectados à rede, distribuição de configurações da rede, gerenciamento de dispositivos, encaminhamento de dados pelo caminho mais curto, mecanismos de segurança, além de recebimento, processamento e encaminhamento de eventos [Ortiz, 2018].

Para executar as suas funções, o NOS possui instalada uma pilha de softwares, incluindo protocolos, como por exemplo OSPF e BGP, que utilizam um dos protocolos para decidir o melhor caminho para um determinado pacote, e ajusta todos os equipamentos de modo que o pacote siga o caminho determinado pelo algoritmo utilizado.

Existem controladores centralizados e distribuídos. Os centralizados possuem a desvantagem de que um único controlador da rede é um ponto de falha. Um único controlador pode não ser o suficiente para gerenciar uma rede com muitos

elementos no plano de dados [PUSHPA, 2018]. Contudo, explorando o paralelismo em computadores de alto desempenho, é possível usar um NOS centralizado para processar milhões de fluxos por segundo, podendo atender a redes de datacenters.

Já um controlador distribuído pode ser escalado para atender aos requisitos de qualquer ambiente. Ele pode ser um conjunto de NOS concentrados em uma localidade ou vários elementos de processamento distantes um do outro. A vantagem de um controlador distribuído é a maior resiliência para falhas físicas e lógicas. No entanto, um obstáculo é manter o estado da rede sempre atualizado para todos os componentes do controlador**.**

Para a comunicação entre os nós do controlador distribuído, são usadas APIs especiais, chamadas de Westbound API e Eastbound API. Cada controlador implementa suas próprias APIs desse tipo. Algumas de suas funções são transmissão de dados entre os nós do controlador, algoritmos para a manutenção da consistência dos dados e recursos de monitoramento e notificação (por exemplo, verificar se um dos nós está ligado e notificação de que um nó assumiu o lugar de outro no controle de um conjunto de dispositivos de rede) [PUSHPA, 2018].

2.5 Plano de Aplicação/Gestão

Aplicativos e serviços que usam serviços do controle e/ou plano de gerenciamento formam o plano de aplicação. Além disso, os serviços dispostos no plano de aplicação podem fornecer serviços para outros serviços e aplicativos que residem no plano de aplicação através da interface de serviço [KREUTZ, 2015]. Exemplos de aplicativos incluem descoberta de topologia de rede, provisionamento, reserva de caminho, etc.

As principais aplicações de rede em uma SDN exercem as funções mais comuns em uma rede, como o roteamento do tráfego, balanceamento de carga e aplicação de políticas de segurança. Além disso, algumas funções mais específicas são realizadas, como economia de energia, aplicação de Qualidade de Serviço (Quality of Service – QoS), virtualização de redes, gerenciamento de mobilidade em redes sem-fio, engenharia de tráfego, dentre outras [ORTIZ, 2018].

Tal plano é responsável também por tomar decisões de fluxo. Cabe ao controlador, quando iniciado, se conectar aos comutadores da rede. Sem uma aplicação instalada, o controlador não realiza as ações. Quando chega uma mensagem do comutador requisitando informações a sobre qual o fluxo de encaminhamento desconhecido, o coordenador redireciona a mensagem à uma aplicação para que essa tome as devidas ações.

2.6 Interfaces de Comunicação

As interfaces de comunicação são denominadas pontos de interação entre duas entidades. E isso ocorre através de uma comunicação entre processos (IPC) ou um protocolo de rede. Se as entidades são colocadas em locais diferentes, a interface é geralmente implementada através de um protocolo de rede [Lamb, Heileman, 2014].

2.6.1 Extremidade-Norte (Northbound)

A Interface Northbound é uma parte importante da arquitetura SDN. Ela é uma API responsável pela comunicação entre o plano de aplicação e o plano de controle, permitindo que aplicações utilizadas por gestores de rede efetuem o controle e o monitoramento das funções da rede sem ter que ajustar os detalhes mais minuciosos da comunicação [Oktian, Lee, 2015]. Isso é possível devido à Interface Southbound, descrita mais adiante.

A principal ideia para esse tipo de interface refere-se ao fato de que um padrão seja estabelecido, possibilitando que seja gerada uma abstração que independa da linguagem de programação e do controlador. Contudo, ainda não se tem um consenso sobre o padrão a ser seguido. Cada controlador especifica sua própria API. Para exemplificar, pode-se citar a API REST que é amplamente usada [KREUTZ et al. 2015].

As funções principais de uma Northbound são traduzir os requisitos das aplicações de gerenciamento em instruções de baixo nível para os dispositivos da

rede e transmitir estatísticas sobre a rede, que foram geradas nos dispositivos da rede e processadas pelo controlador [Oktian, Lee, 2015].

2.6.2 Extremidade-Sul (SouthBound)

A Interface Southbound é o meio de comunicação entre os conjuntos de controle e encaminhamento de dados. Essa interface tem como principal objetivo a separação entre os planos de controle e dados. Ela é também uma API e é por meio dela que os controladores da SDN podem informar os requisitos das aplicações para a rede, possibilitando a reprogramação dos equipamentos para que eles exerçam diversas funções de firewall, como controle de fluxo, sistemas de detecção de intrusos (IDS) e também roteamento e comutação. Essa reprogramação é feita adicionando ou removendo regras das tabelas de fluxos, descritas no plano de dados [Oktian, Lee, 2015].

Essa interface também é usada para os equipamentos de rede se comunicarem com o controlador enviando avisos de eventos caso ocorra uma mudança de porta ou enlace, provendo ao administrador estatísticas de fluxo para melhor detalhamento das características da rede e o envio de pacotes para o controlador em dois casos: Quando alguma das regras instaladas no equipamento tem como comando "enviar para o controlador ou se o equipamento de plano de dados não souberem o que fazer com um pacote, ou seja, quando não existe uma regra definida para pacotes com alguma característica.

Todos esses tipos de comunicação estão definidos na Interface Southbound mais utilizada, o OpenFlow [HAKIRI, 2014]. Contudo há outras APIs, como DCAN, OVSDB, IETF ForCES e Ethanet. Esses dois últimos sendo antecessores diretos do OpenFlow [NUNES, 2014]. Por exemplo, a interface OVSDB permite aos elementos de controle a criação de várias instâncias de comutadores virtuais, configurar políticas de Qualidade de Serviço nas interfaces, configurar túneis e gerenciar filas. Por esse motivo, ela é uma interface complementar ao OpenFlow para o uso com Open vSwitch.

2.6.3 Extremidade-Oeste(Westbound)

A Westbound tem como objetivo transportar informações entre os múltiplos planos de controle de SDN distintos. Com essa solução é possível obter uma visão abrangente da rede que podem influenciar nas decisões de rota de cada controlador. É mais comumente usado em protocolos BGP entre domínios remotos da SDN [HAKIRI, 2014].

2.6.4 Extremidade-Leste (Eastbound)

A Eastbound é responsável pela comunicação entre as SDN e redes convencionais. Também não possuem uma padronização e sua implementação depende de tecnologias não-SDN utilizadas na rede [HAKIRI, 2014].

As duas últimas extremidades listadas têm como característica a possibilidade de cada controlador implementar suas próprias regras. Possibilitando algumas funções de transmissão de dados entre nós do controlador, algoritmo para manutenção da consistência dos dados e recursos de monitoramento e notificação. Um exemplo, é a possibilidade de verificar se um dos nós está ligado e notificação de que um nó assumiu o lugar de outro no controle de um conjunto de dispositivos de rede.

3 Protocolo OpenFlow e Controladores

Em uma SDN, a interface de comunicação entre os comutadores e controladores ocorre por meio de uma interface de programação. O OpenFlow é o protocolo padrão para SDN e tem como ideia básica utilizar funções de manipulação das tabelas de fluxos, que são oferecidas nos comutadores. A ONF (Open Network Foundation), fundada em 2011, é uma organização com o objetivo de gerenciar as especificações do OpenFlow e promover o uso e desenvolvimento de soluções abertas para SDN [MCKEOWN, 2008].

O plano de dados de um Comutador OpenFlow consiste em uma Tabela de Fluxos com uma ação correspondente a cada um desses fluxos. Portanto, isso permite que os fluxos sejam tratados de maneira diferente. Assim, pode ser definido um fluxo de tráfego experimental e outro de tráfego de produção, possibilitando isolamento entre eles. Esse tráfego de produção é tratado da mesma forma que seria na ausência do OpenFlow. Com isso, o OpenFlow pode ser visto com uma generalização do conceito de VLANs, que também permitem o isolamento de tráfego, mas de forma menos flexível [MCKEOWN, 2008].

3.1 Tabela de Fluxo OpenFlow

Cada entrada na tabela de fluxo de um Switch tem uma simples ação anexada a ela. Existem três ações básicas:

- Encaminhamento de pacotes do fluxo para um ou mais portas do equipamento. Permitindo que os pacotes sejam transmitidos pela rede;
- Encapsular e transmitir pacotes do fluxo entre os controladores. Os pacotes são sempre enviados por um canal Seguro, onde ele é encapsulado e enviado ao controlador destino. Geralmente é utilizado pelo primeiro pacote de um novo fluxo;
- Descartar pacotes do fluxo. Isso pode ocorrer para conter ataques DDoS (Negação de serviço), por questão de segurança ou para reduzir a transmissão de mensagens de descoberta enviadas a partir de um ou mais hosts.

Cada entrada da tabela possui três campos:

- Uma regra que define o fluxo. Ou seja, como os pacotes devem ser classificados;
- A regra de como o pacote deve ser processado;
- Estatísticas que mantém o número de pacotes e bytes de cada fluxo, e o tempo decorrido deste último pacote enviado com sucesso no fluxo.

O cabeçalho de fluxo é composto uma tupla de doze itens como é mostrado na imagem abaixo.

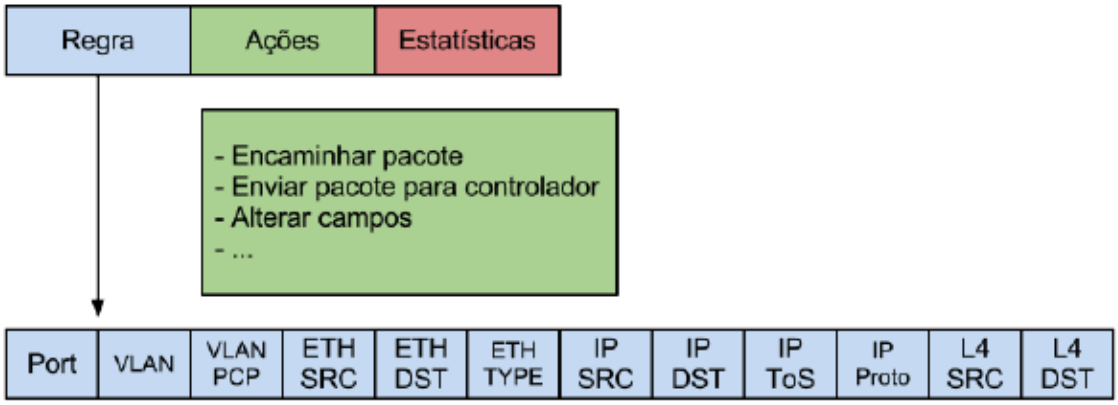

Figura 4 - Cabeçalho OpenFlow.
[GUEDES, 2012]

3.2 Comutador OpenFlow

Um Comutador OpenFlow é composto por, no mínimo, três partes. A primeira consiste na Tabela de Fluxos, que possui uma ação para cada fluxo definido, como descrito anteriormente. A segunda consiste no Canal Seguro que conecta comutador e controlador da rede. O Controlador é utilizado para a configuração da Tabela de Fluxos. A terceira parte consiste no Protocolo OpenFlow, que possibilita a comunicação do Comutador com o Controlador. Com isso os desenvolvedores podem configurar a Tabela de Fluxos sem a necessidade de programar o comutador [CENTENO, 2016].

Existem dois tipos de comutadores OpenFlow. O primeiro tipo consiste em um comutador ou roteador com OpenFlow habilitado que, além de suportar as funcionalidades do OpenFlow, realiza as funções comuns de um comutador ou roteador. O segundo consiste em um comutador OpenFlow dedicado, que não

suporta encaminhamento comum de camada 2 e camada 3 [MCKEOWN, 2008]. A figura 5 exemplifica os componentes desses dois tipos.

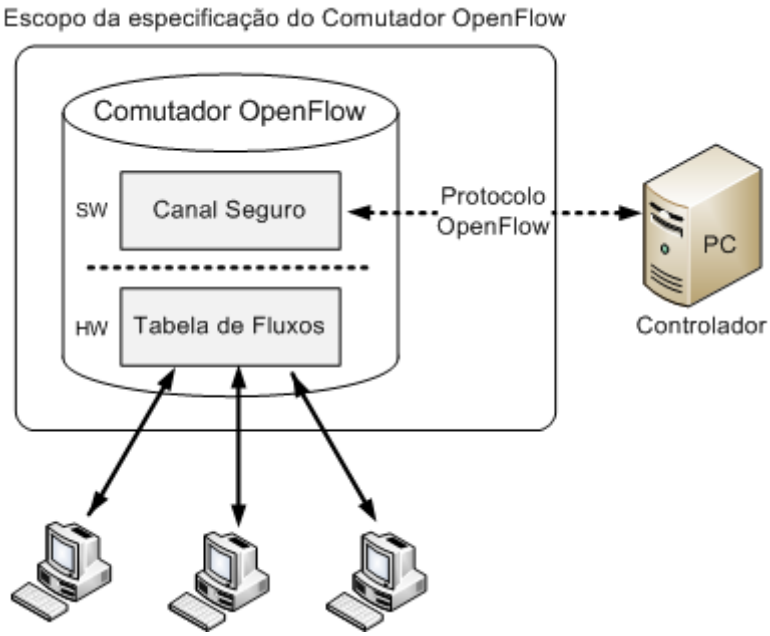

Figura 5 - Comutador OpenFlow.

3.3 Uso das APIs SDN como medida de contenção a ataques

O uso de SDN juntamente com o protocolo Openflow no que se refere a contenção de ataques, pode aumentar a gama de contramedidas a serem adotadas, principalmente através da flexibilização e programabilidade que são incorporadas a tecnologia. Além disso, Openflow fornece APIs padronizadas para comunicação com os switches, que disponibiliza uma ampla possibilidade de estratégias que visam bloquear os ataques.

A figura 6 ilustra o uso de uma API para integração do sistema, geralmente através de REST. As APIs de comunicação com switches, API sul e API de comunicação com aplicações SDN, API norte.

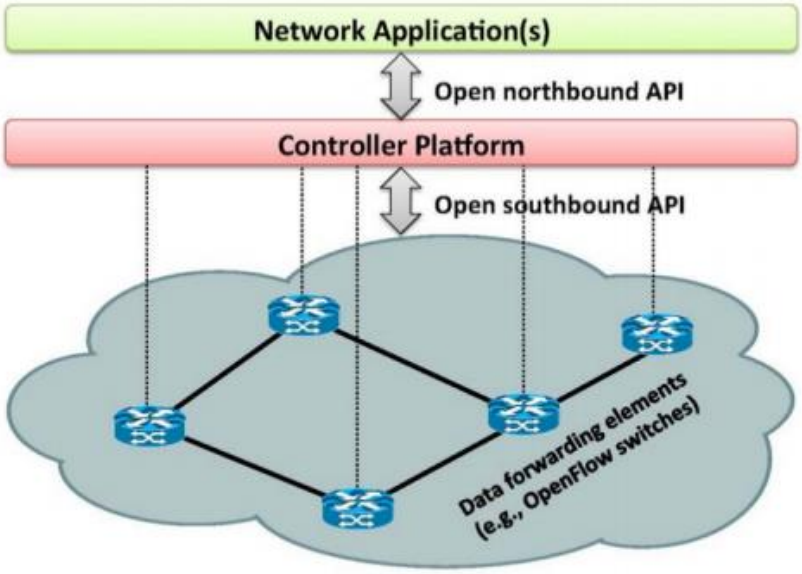

Figura 6 - Arquitetura de SDN e APIs norte e sul.
[Kreutz et al., 2015]

Através dessas APIs os controladores da SDN podem comunicar os requisitos das aplicações para a rede, reprogramando os equipamentos para que eles desempenhem diversas funções, como controle e desvio de fluxo, firewall, sistemas de detecção de intrusão (IDS), além do roteamento e comutação. Essa reprogramação ocorre adicionando ou removendo regras das tabelas de fluxos.

3.4 Open vSwitch (OV Controller)

Open vSwitch é um comutador virtual que suporta a OpenFlow e serve para a criação de múltiplos switches virtuais, permitindo a automação da rede através da programação. Nestes switches é possível implementar diversos protocolos, entre eles o OpenFlow, e também definir um controlador como o POX, o NOX, ou o próprio controlador nativo do Open vSwitch [CENTENO, 2016].

O Open vSwitch também permite a criação de switches virtuais em máquinas físicas diferentes, similar ao Software da VMware. Desta forma é possível criar diversas redes virtuais para testes.

3.5 Controladores SDN baseados em OpenFlow

Ao remover o poder de decisão dos elementos que compõem as redes, quem assume essa responsabilidade é o controlador SDN. No caso do OpenFlow, o responsável pelas decisões é o controlador OpenFlow. Trata-se de um software, que implementa o protocolo OpenFlow, tornando viável o gerenciamento centralizado da rede. Ele é o responsável por monitorar, gerenciar e tomar decisões referente à rede de dados. Pacotes não reconhecidos na tabela de fluxos dos switches OpenFlow são enviados para o controlador, para que este possa tratar e tomar alguma ação levando em consideração a lógica pré-programada.

Abaixo segue alguns dos controladores baseados em OpenFlow mais usados atualmente.

3.5.1 NOX

O NOX foi um dos primeiros controladores OpenFlow. Foi com ele que se introduziu a ideia do sistema operacional de rede. Baseado em C++, é focado em velocidade de processamento dos fluxos.

Seu propósito é prover uma interface de programação de alto nível funcionando com um sistema Operacional para redes. A ideia básica é fornecer uma interface genérica de programação que permite o desenvolvimento de aplicações de gerenciamento de rede [Gude,N., et al, 2008].

3.5.2 POX

O POX é um controlador que surgiu de diversas APIs utilizadas no NOX. O início ele era usado para experimentos educacionais, tendo em vista que sua interface era mais amigável que seu antecessor. Posteriormente foi verificada a possibilidade de utilizar o POX para implementações fora do ambiente acadêmico.

O POX vêm sendo desenvolvido com o objetivo em iniciativas de pesquisa e ensino. Ele é considerado o irmão mais novo do NOX, sua essência é

uma plataforma simples para o desenvolvimento e prototipagem rápida de software de controle de rede usando o Python [CENTERO, 2016].

3.5.3 RYU

O RYU é um controlador SDN, implementado em Python e é um programa de código aberto. Ele é utilizado para gerenciamento de redes e aplicações de controle. Um de seus pontos fortes é o suporte para várias protolocos, tais como Network Configuration Protocol (NETCONF), Configuration Procotol (OF-Config), o OpenFlow, e outros.

RYU fornece componentes de software com APIs muito bem definidas, o que facilita a criação de novas formas de gerenciamento de redes e aplicações de controle. Ele prove suporte as versões do OpenFlow 1.0 a 1.5 e também dispões de integrações que auxiliam em seu desenvolvimento com outras plataformas [RYU v.4.30, 2018].

3.5.4 FlowVisor

O FlowVisor é uma ferramenta desenvolvida em maio de 2009 em Stanford. Consiste basicamente de um controlador OpenFlow especializado, o seu diferencial em relação aos demais, é que sua funcionalidade atua como um proxy entre um comutador OpenFlow e múltiplos controladores OpenFlow [COSTA, 2015], ou seja, ele atua como um intermediador entre as requisições de um controlador e o comutador. Isso permite que o FlowVisor divida a rede e tenha um controle distribuído, já que quando um controlador OpenFlow envia algum comando para o comutador ele passa antes pelo FlowVisor que determina se ele tem autorização ou não para realizar esta ação [SHERWOOD, 2009].

4 Segurança em redes SDN

Atualmente com a evolução da rede de computadores e seu amplo engajamento, a segurança se tornou um ponto crítico, tendo em vista que ferramentas para capturar trafego, quebrar sistemas de criptografia, capturar senhas, e ataques de negação de serviço se tornaram cada vez mais comuns e sofisticados [CENTENO, 2016].

A grande quantidade de dados ou informações a ser processadas e armazenadas demandam um considerável esforço de segurança para que sua integridade se mantenha inalterada. Isso implica em sistemas de proteção cada vez mais robustos e complexos, que atendam a demanda, cada vez mais crescente da rede. E que sejam capazes de detectar um ataque e aplicar soluções de mitigação em um curto período de tempo para que os serviços de rede não sejam interrompidos ou que não sofram degradação ao longo do tempo.

Nas redes de computadores, a segurança das informações e disponibilidade dos serviços é fundamental para manter as informações integras e confidenciais [ZHANG H. et al, 2018].

Segundo [Pereira et al., 2016], existem diversos mecanismos para proteger uma rede de computadores, dentre eles, criptografia, controle de acesso, Segurança Física e pessoal e por fim contar com integração de hardware e software de segurança.

4.1 Segurança no contexto de SDN

Por padrão uma SDN não possui ferramenta para detecção de intrusão, o que por consequência, a torna vulnerável à maioria dos ataques aos quais as redes tradicionais são vulneráveis [Ajaeiya et al. 2017].

Embora a separação do plano de controle e de dados na SDN permita aos desenvolvedores criar novas ferramentas de segurança, o uso integrado de outras ferramentas de detecção de intrusão (IDS) traz novas formas e soluções mais robustas, aumentando a flexibilização da arquitetura SDN, já que com o plano de

controle possibilita o redirecionamento de pacotes no caso de uma detecção de ataque, por exemplo [Dalpissol, Vicentini, Santin, 2018].

[Zanna et al. 2014] mostraram que é possível integrar um sistema de detecção de intrusão (IDS) com um controlador SDN para realizar detecção e bloqueio de ataques de negação de serviço.

4.2 Ataques de Intrusão

Ataque de intrusão pode ser caracterizado como um conjunto de ações que tentam comprometer recursos de um sistema de computador ou diversas tentativas de explorar informações de qualquer natureza. Estes ataques têm como objetivo corromper três componentes básicos da segurança de informação [Campos, M. e Martins, J. 2017].

- **Integridade**: este princípio estabelece que toda informação só possa sofrer acréscimos, reduções ou atualizações por pessoas previamente autorizadas, o que mantém suas características originais e que, portanto, é integra.
- **Confidencialidade**: princípio que estabelece a garantia que somente pessoas previamente autorizadas tenham acesso à informação. Além disso, estabelece os casos em que a divulgação da existência da informação é proibida.
- **Disponibilidade**: garantir que toda informação deve estar sempre acessível às pessoas previamente autorizadas, no momento em que necessitarem utilizá-la. Sua eficiência está atrelada aos princípios vistos anteriormente, pois no momento em que a informação está disponível é necessário que sejam garantidas a confidencialidade e a integridade do material a ser acessado.

Esse tipo de ataque pode agir de diversas formas baseando-se em falhas de segurança de sistemas, aplicações, protocolos ou relacionado a falhas de configurações. Também podem ocorrer nas diferentes camadas do modelo TCP/IP

e, de acordo com a camada que atuam, esses ataques são classificados em quatro grupos

- Negação de Serviços ou Denial of Service - DoS: esse tipo de ataque tem como objetivo gera indisponibilidade de recursos em um determinado sistema ou equipamento, como consumo excessivo de memória, processador ou rede. Para isso, um atacante gera muitas requisições de modo que os recursos do sistema fiquem sobrecarregados, ao ponto de não conseguir mais atender as requisições de usuários legítimos, impedindo os acessos aos recursos ou informações no local onde está alocado o sistema.
- Remoto para Usuário: correspondem a situações em que o intruso possui conectividade com a máquina vítima, sem possuir uma conta de usuário e explorando alguma vulnerabilidade existente, consegue obter acesso local à máquina. Ataques da classe reconhecimento (Probing) são normalmente empregados em uma etapa que antecede o ataque ou intrusão.
- Para super usuário: englobam todos os ataques em que o intruso possui acesso ao sistema como um usuário normal e consegue elevar seu nível de privilégio para o de um usuário especial (como o usuário root em sistemas Unix ou administrador em outras plataformas).
- Exploração: busca explorar vulnerabilidades do sistema, de aplicações instaladas sobre o sistema ou ainda falhas do usuário que permitam ataque futuros. É normalmente empregado em uma etapa que precede outros tipos de ataque.

Os ataques de intrusão ocorrem das mais diversas formas e em todas as camadas do Modelo TCP/IP. Contudo, existem sistemas que possuem componentes que desempenham funções como sensores e analisadores de eventos, que provêm

a capacidade de detectar, analisar e identificar cada tipo de evento, os Sistemas de Detecção de Intrusão, ou Intrusion Detection System – IDS.

Em resumo, o sistema seguro deve proteger dados e recursos contra acessos não autorizados, interferência e bloqueios de uso. Com base nessas premissas, uma intrusão pode ser caracterizada como uma ação ou conjunto de ações que visem comprometer a confidencialidade, integridade ou disponibilidade de um dado sistema.

4.3 Sistema de detecção de intrusão

Os IDS são sistemas automáticos que funcionam como analisadores de tráfego, que monitoram em tempo real o tráfego na rede e detectam tentativas não autorizadas de acesso à infraestrutura lógica. São ferramentas de segurança que, como antivírus e firewalls, se destinam a segurança de um ambiente de rede, de modo que os requisitos básicos de segurança em uma rede não sejam corrompidos [CARVALHO, 2018].

Os IDS são considerados como uma das principais ferramentas de defesa contra-ataques a sistemas, se tornando um componente importante para qualquer sistema de segurança em rede de computadores. Seu monitoramento se baseia nos tipos conhecidos de ataques e também verificando alterações de comportamento no tráfego de dados, onde ocorrendo uma tentativa de intrusão, o sistema é capaz de gerar um alerta informando o ocorrido aos responsáveis pela segurança [Fernandes, 2017].

Um IDS pode utilizar basicamente duas formas de detecção:

- **Baseado em anomalias**: Nesse método é feito uma busca identificar de um ataque na ocorrência de algum evento fora do padrão, como tráfego em portas incomuns, anomalias da rede e elevado volume de tráfego, tentativa de acesso não autorizado a um recurso do sistema, comportamentos anormais do sistema que possam indicar atividade maliciosa na rede [Fernandes, 2017].

- **Baseado em regras ou assinaturas**: Neste tipo, o IDS visa a detecção de intrusos através da utilização de uma base de dados de ataques conhecidos. Esta base de dados é composta por um conjunto de regras que caracteriza determinado intruso. [Fernandes, 2017]

Na arquitetura de rede tradicional, o IDS atua de algumas formas, sendo uma delas na identificação de ataques dentro de uma rede. Essa forma é denominada NIDS (Network Intrusion Detection System) que consiste em monitorar uma rede e buscar identificar atividades que envolvam todos os dispositivos dessa rede [Golveia, 2017].

Em uma rede SDN, um IDS pode fazer a inspeção dos pacotes que trafegam na rede gerando alarmes caso identifique alguma anormalidade. O controlador da rede, baseando-se em ações pré-determinadas pelo administrador, aplica ações para responder a intrusão [KREUTZ et al. 2015]. A figura 7 ilustra essa interação entre IDS e o controlador SDN. Onde é possível observar a utilização das interfaces Northbound e Southbound para a comunicação entre a aplicação de rede, no caso o IDS e os operadores da rede, tendo como intermediador o controlador da rede.

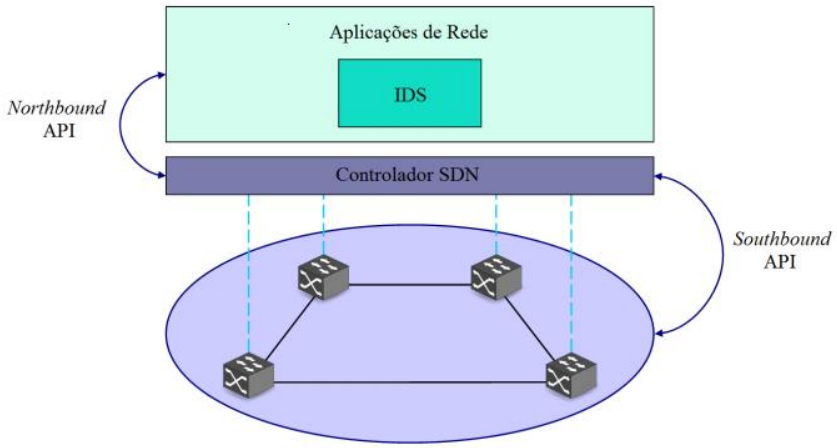

Figura 7 - Interação entre IDS e SDN.

Com a utilização das APIs do protocolo OpenFlow é possível usar um canal seguro para comunicação entre switch e controlador, fazendo com que a comunicação não sofra com ataques mal-intencionados [Ortiz, 2018]. Para dar confiabilidade ao canal de comunicação, a interface de acesso recomendada é o protocolo Secure Socket Layer – SSL, amplamente utilizado, que utiliza a cifragem dos dados trafegados utilizando certificado confiável [Fernandes, 2017].

5 Ambiente de Simulação

Este capítulo tem como objetivo apresentar as principais tecnologias envolvidas no desenvolvimento deste trabalho. Na seção 5.1 é apresentado o sistema Mininet e suas principais características. A seção 5.2 refere-se ao controlador RYU e os motivos pelos quais ele é usado nos testes e finalmente na seção 5.3 é apresentado o Sistema de Detecção de Intrusão Snort.

5.1 *Tecnologias Envolvidas*

5.1.1 Mininet

Para o ambiente de simulação, foi usada uma máquina virtual disponível no site oficial do Mininet[1]. Essa é uma maneira simples já que todas as dependências de pacotes, binários ferramentas do OpenFlow já estão pré-instaladas e o Kernel também está ajustado para os testes necessários nesse trabalho.

Mininet é um sistema que permite uma prototipação rápida e simulação de Rede Definida por Software, com ferramentas para criação de controladores de rede, permitindo o desenvolvimento de aplicações com o OpenFlow.

Conforme descrito pelo projeto Mininet [GITHUB Mininet, 2019], os equipamentos virtuais, hosts, switches, links, e controladores são reais, apenas implementados em software, e não hardware. Portanto a performance dos equipamentos está limitada ao computador que está realizando a simulação.

Cada host virtual criado pelo Mininet pode executar programas, ler e escrever arquivos, etc. Isso torna as possibilidades de experimentos muito amplas [GITHUB Mininet, 2019].

Ao criar uma rede no Mininet, são emulados:

[1] http://mininet.org/download/

- **Links**: age como um enlace conectando a duas interfaces de rede virtuais, conectando hosts, switches e controladores.

- **Hosts**: Dispositivo com uma interface de rede independente. Cada host tem interfaces Ethernet, portas e tabelas de roteamento próprias. O disco rígido da máquina virtual é compartilhado entre todos os hosts.

- **Openvswitch**: São dispositivos virtuais que podem ser remotamente gerenciados e configurados pelo Controlador. Openvswitches fornecem o mesmo mecanismo de configuração e funcionamento que switches físicos e foi desenvolvido especialmente para trabalhar com o protocolo OpenFlow.

- **Controladores**: Por meio de um console, definem critérios de manipulação de pacotes dos switches e roteadores, podendo estar em qualquer lugar da rede, desde que a máquina virtual que está executando os switches tenha conectividade IP com o controlador.

O Mininet é uma ferramenta muito confiável com desempenho extremamente rápido. É uma plataforma eficaz para que a comunidade do SDN possa realizar diversos testes com eficiência. A ferramenta permite emular até 4096 hosts em um único terminal, mas perde desempenho quando os recursos solicitados numa nova topologia ultrapassar os disponíveis na CPU, ou exigir largura de banda superior ao da máquina física.

O Miniedit, também desenvolvido pelo próprio projeto Mininet, é uma interface gráfica simples que permite a prototipação da arquitetura de uma rede, conectando hosts com switches e switches com controladores. Permite gerar código que compatível com o Mininet para ainda maior comodidade de simulação. [GITHUB Mininet, 2019].

5.1.2 Controlador RYU

Para o controlador SDN foi optado pelo RYU já que ele dispõe de suporte para as versões mais atuais do OpenFlow. Fornece componentes de software com APIs bem definidas que facilitam o gerenciamento de rede e também foi escrito em python [RYU, 2019].

No decorrer do levantamento de requisitos para o desenvolvimento do trabalho, foi cogitado usar o controlador NOX. Ele foi o primeiro controlador SDN desenvolvido. Inicialmente suportava as linguagens C++ e Python, mas as versões foram separadas: a versão Python tem o nome de POX e a versão C++ ficou como NOX. Desde que foi criado, o NOX era bastante adotado e foi usado como base para outros controladores, mas seu uso está em grande queda frente a novos controladores. Uma das razões é o fato deste controlador não suportar as versões mais recentes do OpenFlow, pois suporta apenas até a versão 1.0.

Outro fator importante para a escolha do Ryu foi sua integração nativa com o IDS SNORT. Segundo consta na documentação do Ryu, ele dispõe de duas formas de integração: A primeira forma é em uma máquina executando o IDS e o controlador, que torna o experimento mais simples, e a segunda forma, onde o controlador está em uma máquina e o IDS em outra.

5.1.3 Sistema de Detecção de Intrusão SNORT

Snort é uma ferramenta IDS de código-fonte aberto, desenvolvida por Martin Roesch, sendo muito popular pela sua flexibilidade nas configurações de regras, sua capacidade de fazer análise de tráfego em tempo real e constante atualização. Seu código fonte é desenvolvido em módulos utilizando a linguagem C possuindo documentação de domínio público [SNORT, 2019].

O Snort faz a sua detecção baseada em assinaturas utilizando uma linguagem flexível de regras para analisar o tráfego coletado. Pode ser configurado para funcionar em três modos diferentes:

- **Modo Sniffer**: no qual simplesmente lê os pacotes da rede e os mostra como um fluxo contínuo no terminal de saída;

- **Modo Packet Logger**: neste modo o Snort além de capturar o tráfego, grava as informações de todos os pacotes em disco;

- **Modo Sistema de Detecção de Intrusão de Rede**: Nesse modo, o sistema analisa o tráfego da rede comparando os pacotes capturados com regras pré-definidas, a fim de detectar alguma atividade maliciosa.

Pelo fato de ser livre de distribuição, existe uma base de dados com milhares de assinaturas/regras que são disponíveis para download e que são atualizadas diariamente por usuários Snort que estão espalhados pelo mundo, assim permitindo a realização de atualizações constantes e respostas imediatas contra novos ataques [SNORT, 2019].

Uma das vantagens do Snort é permitir que o usuário produza suas próprias regras. Estas regras constituem a parte mais importante do IDS que podem levar ao sucesso ou não da detecção das intrusões. Esta ferramenta utiliza uma linguagem de descrição simples e fácil de usar, assim permitindo customizar as regras de acordo com as características do ambiente no qual deseja trabalhar, refinando ou ampliando o poder de detecção do Snort [SNORT, 2019].

Um dos diferenciais que uma rede SDN em conjunto com o Snort podem trazer é que os bloqueios dos pacotes podem ser feitos de forma mais eficiente. Com as redes tradicionais, o bloqueio de pacotes é feito, em sua grande maioria, nos firewalls da rede, localizados na borda da rede. Com isso, existe muito tráfego malicioso na rede utilizando banda desnecessariamente em switches que não tem a capacidade de bloquear os mesmos.

5.2 Ambiente de simulação

Para viabilizar os testes, será usado o ambiente de simulação Mininet, que permite emular componentes típicos de arquiteturas SDN. O controlador Ryu e o IDS SNORT estão inseridos na mesma máquina para facilitar a integração dessas duas ferramentas.

As regras dentro da ferramenta SNORT foram desenvolvidas para identificar intrusões dentro de um ambiente de rede controlado.

Esse ambiente é mostrado na figura 8, que evidencia como todos os componentes da rede são simulados no Mininet em uma máquina virtual disponibilizada no site oficial do Mininet. Nessa VM o ambiente já dispões de todos os pacotes necessários para que se crie uma rede SDN, sendo necessário apenas baixar os pacotes do controlador escolhido, que nesse caso foi o controlador RYU.

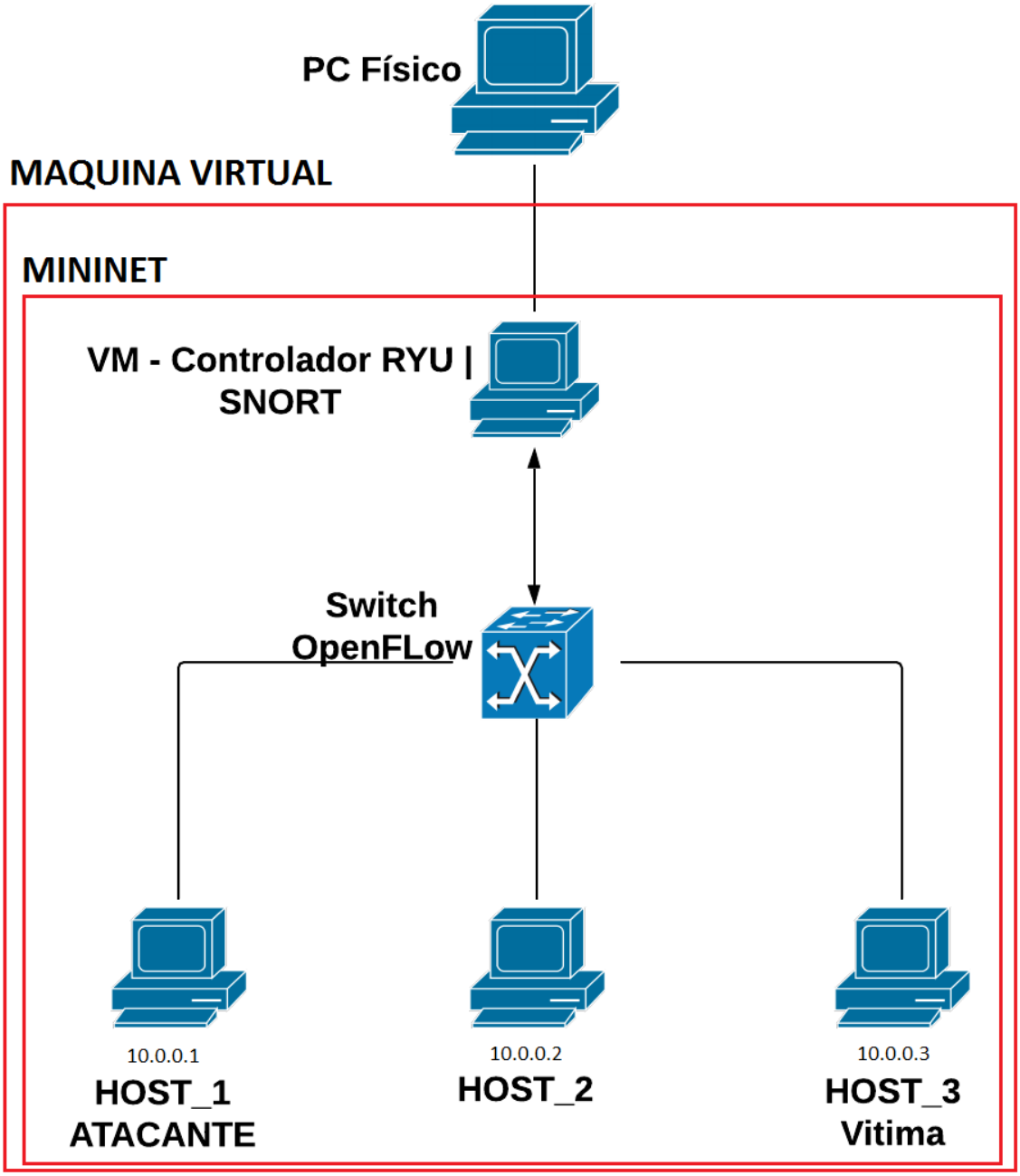

Figura 8 - Ambiente de teste.

5.2.1 Recursos e topologia do Ambiente

Para o ambiente de simulação, mostrado na figura 8, no “PC Físico”, onde foram criadas as máquinas virtuais, foi utilizado um notebook, com processador Intel

Core i5-4200U de 1.6GHz e 8GiB de memória RAM, utilizando um disco com capacidade de 1TB. Como sistema operacional, foi utilizado o Windows 7 Ultimate, para a virtualização das máquinas virtuais foi empregado a versão 6.0.4 do virtualbox.

A máquina virtual denominada “VM – Controlador RYU | SNORT” mostrada na figura 8, é responsável pelo ambiente onde estão rodando o emulador Mininet, o controlador RYU e o IDS SNORT. Foi criada uma máquina com as seguintes especificações: 2 processadores virtuais core, 2GiB de memória RAM, utilizando um disco virtual de 20GB. Utiliza o sistema operacional Ubuntu na versão 14.04 que é disponibilizado oficialmente e o Mininet está na versão 2.2.2.

Nesta máquina virtual, para a criação das regras e identificação de intrusão, foi utilizado a versão 2.9.6.0 GE do SNORT. Para a validação do trabalho, foram desabilitadas as regras pré-definidas no Snort para que não interferisse nos resultados finais desse trabalho. Também nessa máquina virtual, foi usado o WIRESHARK na versão 1.0.6 disponibilizado previamente pela imagem do MININET, para a análise do tráfego e como forma de apoiar a criação das regras.

O ambiente de simulação é flexível o bastante para acomodar diversos cenários. O controlador é capaz de fazer a configuração de diversos switches OpenFlow de acordo com a quantidade de sub-redes que forem criadas. É possível também ampliar o número de controladores, onde cada um terá um grupo de switches para gerenciar. E também é possível alterar a quantidade de IDS na rede.

5.2.2 Configuração do ambiente e Resultados.

Para gerar o ambiente de teste deste trabalho, foi utilizado o controlador SDN e IDS na mesma máquina virtual e por isso foi necessário criar uma topologia de rede na ferramenta Mininet. Sendo assim, foi criada uma topologia, composta por três hosts, um switch e um controlador, todos instanciados através do Mininet, com o comando da figura 9.

```
sudo mn --topo single,3 --mac --controller
remote --switch ovsk,protocols=OpenFlow13
```

Figura 9 - Comando para criação da topologia no mininet.

Esse comando cria uma topologia contendo três hosts e um único switch do tipo openvSwitch, define os endereços MAC de cada host, igual ao seu IP, define um controlador remoto cujo padrão é o localhost e aplica para esse controlador a versão 1.3 do protocolo OpenFlow.

Os passos que o Mininet executa:

- Cria 3 hosts virtuais, cada uma contendo um endereço IP separado.
- Cria um único switch em software OpenFlow no kernel com 3 portas.
- Conecta cada host virtual ao switch com um cabo ethernet virtual.
- Define o endereço MAC de cada host sendo parecido com o seu IP.
- Configura o switch OpenFlow para se conectar a um controlador remoto.
- Atribui a versão 1.3 do protocolo OpenFlow para o controlador.

A figura 10 mostra a saída do comando, sinalizando que foram criados três hosts, h1, h2 e h3, um switch denominado s1 efetuado a conexão “física” entre os hosts e o switch e por fim é iniciado o switch s1 e o controlador denominado c0. Na figura 10 Também é possível observar que é definida a porta 6653 para comunicação entre o controlador c0 e o switch s1.

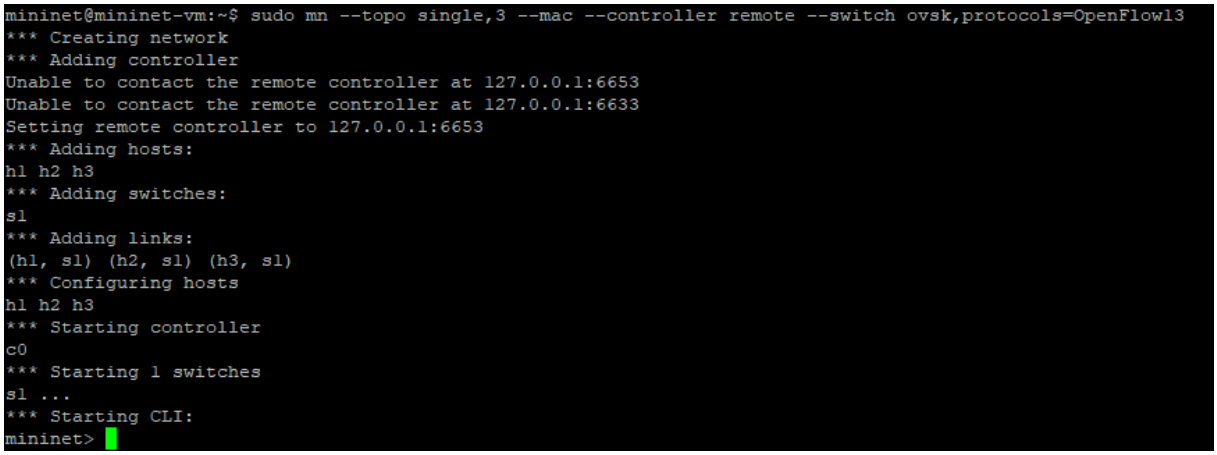

Figura 10 - Criação do ambiente de simulação.

Sem a execução do controlador SDN, mesmo com a instanciação das máquinas, não existe comunicação entre os hosts. Isso se deve ao fato do controlador criar a tabela de fluxo com base no tráfego da rede.

A figura 11 mostra que mesmo quando todos os hosts estão conectados ao switch, a comunicação entre eles não existe.

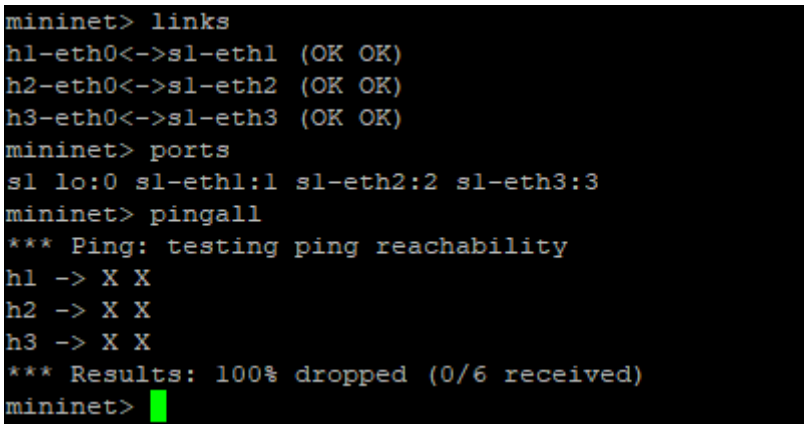

Figura 11 - Comunicação sem ativação do Controlador.

Quando o controlador recebe algum novo quadro para o qual o switch não encontrou nenhuma regra de encaminhamento, o método packet_in_handler, componente base do controlador RYU, cria novas regras de mapeamento que inclui escolher de forma arbitrária uma topologia lógica para onde transitar esse quadro. Essa é a forma básica de como o controlador informa ao swtich para onde o pacote deve ser encaminhado.

A figura 12 mostra a saída do comando dump-flows que exibe informações sobre os pacotes recebidos pelo controlador. Essa saída mostra o caminho que cada pacote fez, mostrando que a tabela de fluxo foi criada e os hosts possuem comunicação entre si, já que o envio de dados na rede é baseado em fluxo.

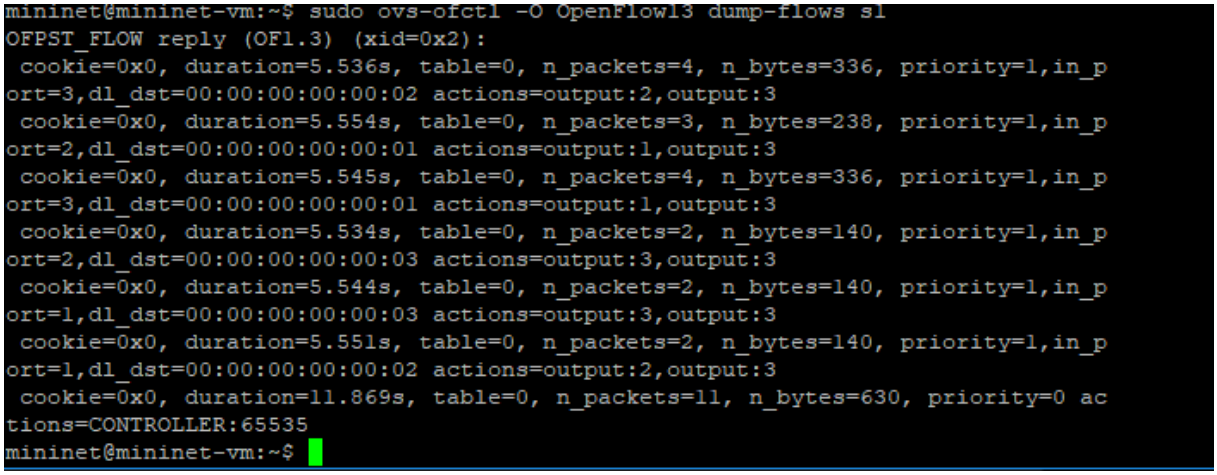

Figura 12 - Fluxo dos pacotes na rede.

Com a execução do controlador, a tabela de fluxo da rede é criada, com isso os hosts, ao encaminhar um pacote, tem um caminho já determinado pela tabela de fluxo. A figura 13 mostra que a comunicação entre os hosts já está estabelecida e cada host possui comunicação com todos da rede.

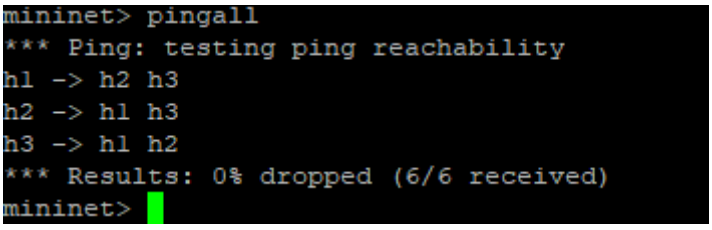

Figura 13 - Comunicação entre os hosts estabelecida.

A seção a seguir descreve os principais componentes usados para que a integração entre controlador SDN RYU e o IDS Snort funcione corretamente.

5.3 Integração SNORT e Controlador RYU

A forma de fazer a integração foi uma decisão importante para o andamento deste trabalho. A falta de integração entre os controladores disponíveis com dispositivos de segurança, mais especificamente com o IDS Snort, para detecção de ataques de intrusão, foi o motivo para a escolha do controlador Ryu.

Este controlador oferece duas formas para a integração com o IDS Snort, na primeira opção, mostrada na figura 14, tanto o Ryu quanto o IDS trabalham em uma mesma máquina, onde o controlador recebe os pacotes de alerta via Unix Domain Socket, que é um mecanismo de comunicação entre processos que permite a troca de dados bidirecional entre processos em execução em uma mesma estação de trabalho [RYU, 2019].

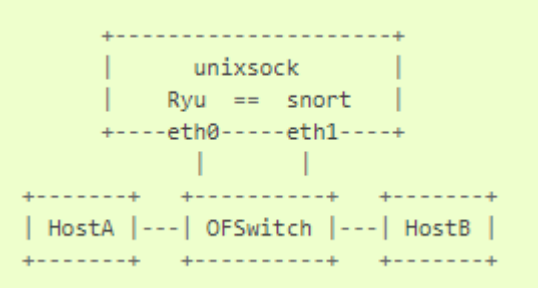

Figura 14 - Arquitetura em uma única maquina.
[RYU, 2019].

Na segunda opção, mostrada na figura 15, o controlador e o IDS estão trabalhando em máquinas diferentes, pois o Snort exige um poder computacional muito alto para o desempenho não ser degradado quando for necessário processar um número elevado de pacotes. O controlador Ryu recebe os alertas via Network Socket, uma conexão segura [RYU, 2019].

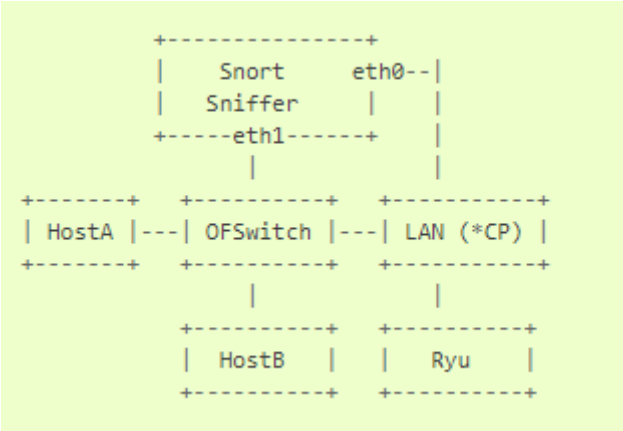

Figura 15 - Arquitetura em duas maquinas.
[RYU, 2019].

Para este trabalho, foi decidido que para fins de testes, a solução mais adequada a ser executada é o controlador Ryu e o IDS executavam na mesma máquina. Foi escolhido dessa forma para simplificação das configurações dos módulos e também para não comprometer os experimentos devido à alta demanda de CPU e memória da máquina física.

A figura 16, mostra como a arquitetura foi desenvolvida para este trabalho.

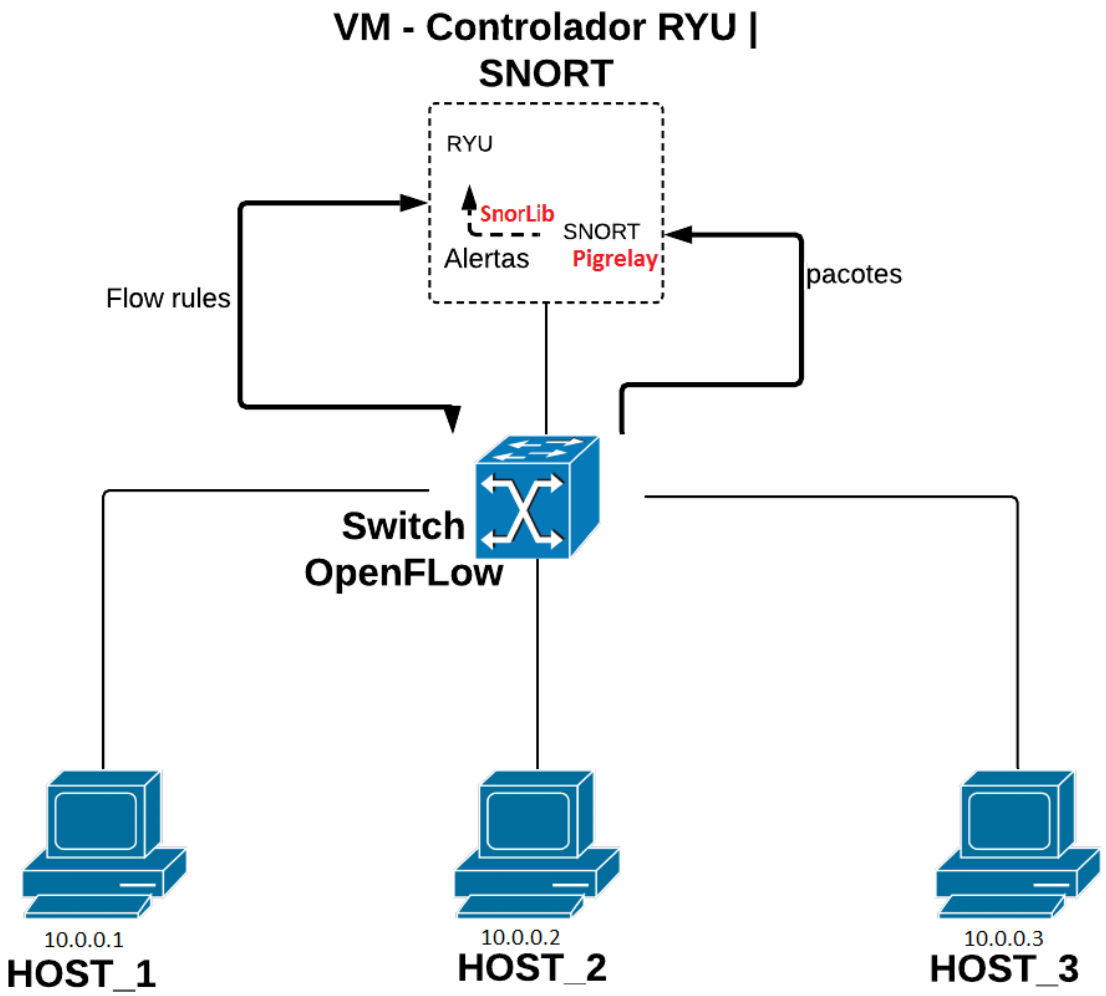

Figura 16 - Arquitetura referente a integração entre controlador SDN e IDS.

A figura 16 mostra que para monitorar os pacotes entre os hosts, são necessárias duas interfaces de rede na máquina virtual que integra o RYU e o Snort. Uma para conexão do controlador ao switch e outra em modo promíscuo para conexão com o Snort. Os módulos, pigrelay e snortlib, são os responsáveis pelo funcionamento da integração entre controlador e IDS.

Todo tráfego que é enviado na rede é espelhado para interface conectada ao Snort. Se alguma regra detectar alguma anomalia na rede, o IDS enviará alertas ao controlador via Unix Socket, já que a comunicação entre controlador e IDS é feita processo a processo. Após isso, o controlador Ryu poderá tomar a decisão que for necessária em sua rede.

Para que a integração aconteça, é necessário usar o módulo simple_switch_snort.py disponível no projeto do Ryu, que introduz a função de detecção de intrusão do Snort na rede SDN e encaminha pacotes recebidos da rede para o Snort que por sua vez envia os alertas de possíveis intrusões para o controlador.

A figura 17 apresenta a execução do controlador que realiza a conexão com o IDS, onde é necessário modificar o modo de conexão para que funcione via Unix Domain Socket.

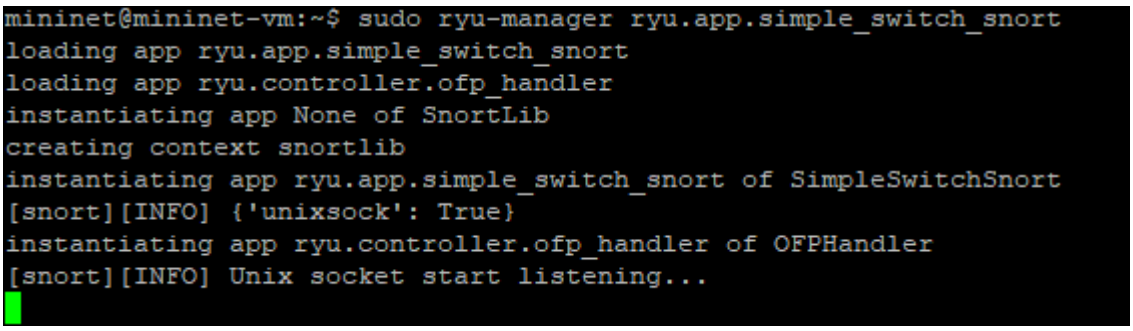

Figura 17 - Conexão entre controlador e Snort.

Também é possível observar que é instanciado o módulo SnortLib, responsável pelo recebimento dos alertas. Para ser possível o recebimento, o SnortLib inicia um socket e fica escutando a rede por conexões.

5.3.1 Módulo Pigrelay

O pigrelay é um software presente na máquina Snort, que verifica os alertas em tempo real. A cada alerta gerado, o pigrelay envia ao controlador o pacote que gerou o alerta e a mensagem informativa sobre o alerta. Esse conjunto de informação é denominado Alertpkt [J. LIN, 2019].

Essa ação é necessária para que o controlador possa, a cada pacote identificado como malicioso, mostrar na tela de execução um alerta com os dados do atacante. Isso é necessário para que o operador da rede possa analisar todo o tráfego malicioso de forma indepentende.

Para o pigrelay ser capaz de enviar os alertas para o controlador, ele deve detectar quando um alerta é gerado. Para isso, o pigrelay fica escutando o socket que o Snort inicializa. Essa é uma função que o Snort executa para que seja possível fazer tratamento dos pacotes que geraram os alertas. O Snort envia diversas informações sobre o pacote que gerou o evento de alerta.

5.3.2 Módulo *SnortLib*

O SnortLib é responsável pelo recebimento dos alertas do pigrelay. Para que seja possível realizar o recebimento, o Snortlib instancia um socket e permanece escutando a rede por conexões. Essa conexão é utilizada pelo pigrelay, como descrito anteriormente. O pigrelay envia diversas informações para o módulo SnortLib, que, por sua vez, faz a separação das informações enviadas. Esse tratamento faz com que o controlador Ryu agora seja capaz de identificar as informações referentes ao pacote que gerou o alerta no Snort, como o MAC Address, endereço IP dentre outros atributos.

Após o tratamento da mensagem, o módulo SnortLib gera um evento que é capturado pelo Módulo de Processamento. Esse evento contém diversas informações, provenientes do Alertpkt e da mensagem do Snort, porém no formato que o controlador Ryu utiliza. Assim o controlador Ryu pode analisar o pacote que gerou o alerta caso necessário, além de identificar a mensagem enviada pelo Snort.

5.4 Criação de regras no SNORT

Para cada tipo de ataque existem alguns padrões, o que ajuda na elaboração de regras de intrusão. E para auxiliar a criação das regras dessa subseção foi utilizado o Wireshark, que é uma ferramenta que possibilita a análise de tráfego gerado na rede, e os organiza por protocolos através de uma interface, com a possibilidade de utilizar filtros conforme a necessidade do operador. O tráfego gerado durante os testes de intrusão são representados de forma semelhante ao Wireshark, como é possível observar na figura 18.

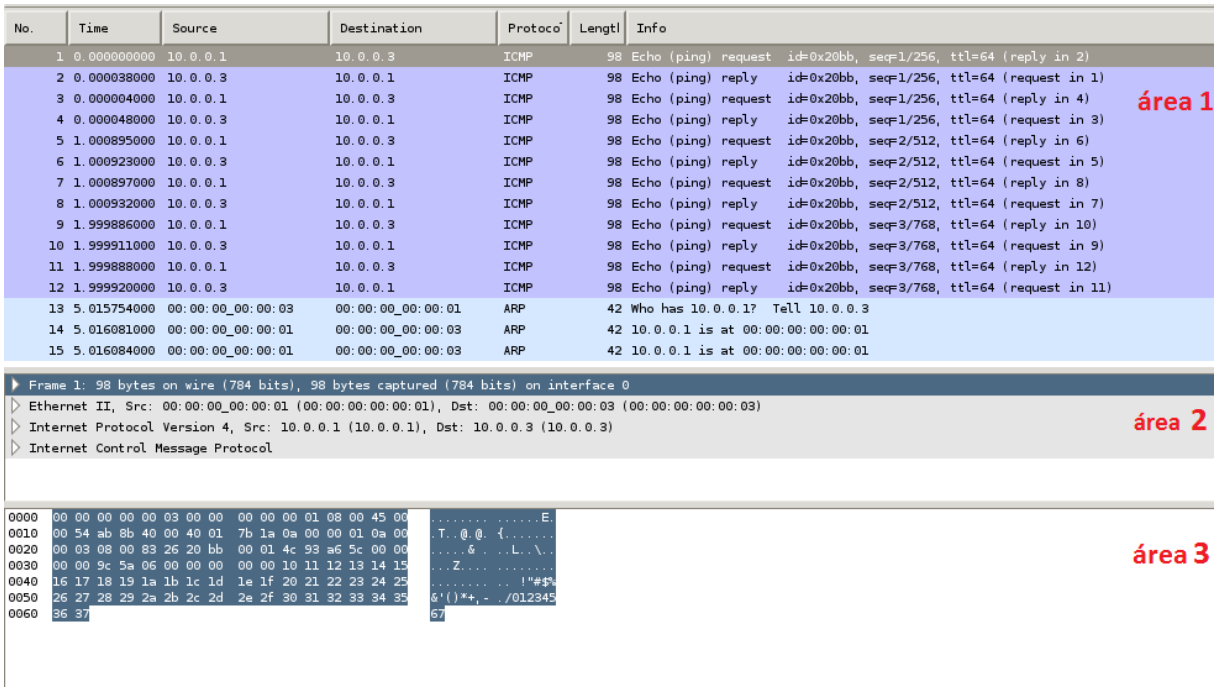

Figura 18 - Dados coletados pelo wireshark.

É possível verificar os pacotes no wireshark através da sua interface gráfica que é dividida em três áreas de visualização, conforme mostrado na figura 18, Cada linha da área 1 representa um pacote capturado e é possível visualizar o endereço que realizou o ataque e o host de destino do ataque, o protocolo ao qual o ataque é direcionado, o tamanho do pacote e informações sobre o pacote.

Ao selecionar uma linha específica, podemos obter mais informações sobre os dados capturados, como é mostrado na área 2, onde é possível ver os detalhes do pacote. Ao expandir cada linha da área 2, o Wireshark vai dividindo os dados. Com isso, podemos facilmente encontrar a porta de destino de um determinado protocolo selecionando a entrada na opção correspondente e procurando a porta de destino, conforme destacado na figura 18. Quando selecionado uma linha da área 2, a área 3 mostra os bytes brutos do pacote que são vistos em hexadecimal e código ASCII (American Standard Code for Information Interchange) como é destaque na área três da interface.

5.4.1 Especificação de Regra para ocorrência de intrusão ICMP Flooding

A especificação desta regra se baseia na análise de tráfego gerado pelo ataque direcionado ao protocolo ICMP. O ataque é caracterizado por causar um grande volume de tráfego no host vítima, de forma que sua banda fique congestionada. Essa forma de ataque pode deixar o sistema lento ou até mesmo derrubá-lo.

Um ataque ICMP Flooding se baseia no envio constante a partir de um endereço IP, de uma grande quantidade de pacotes echo request (ping) até que o limite de requests por segundo seja ultrapassado. A figura 19 apresenta a captura do pacote no Wireshark, onde é possível observar que em aproximadamente seis segundos, foram capturados 280787 pacotes.

No.	Time	Source	Destination	Protocol	Length	Info
280772	5.970309000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=be99) [Reassembled in #280806]
280773	5.970309000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=7400, ID=be99) [Reassembled in #280806]
280774	5.970310000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=8880, ID=be99) [Reassembled in #280806]
280775	5.970311000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=10360, ID=be99) [Reassembled in #280806]
280776	5.970312000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=11840, ID=be99) [Reassembled in #280806]
280777	5.970313000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=13320, ID=be99) [Reassembled in #280806]
280778	5.970313000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=14800, ID=be99) [Reassembled in #280806]
280779	5.970314000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=16280, ID=be99) [Reassembled in #280806]
280780	5.970315000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=17760, ID=be99) [Reassembled in #280806]
280781	5.970315000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=19240, ID=be99) [Reassembled in #280806]
280782	5.970316000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=20720, ID=be99) [Reassembled in #280806]
280783	5.970317000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=22200, ID=be99) [Reassembled in #280806]
280784	5.970318000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=23680, ID=be99) [Reassembled in #280806]
280785	5.970319000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=25160, ID=be99) [Reassembled in #280806]
280786	5.970319000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=26640, ID=be99) [Reassembled in #280806]
280787	5.970320000	10.0.0.3	10.0.0.1	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=28120, ID=be99) [Reassembled in #280806]

Figura 19 - Ataque ICMP.

Para gerar o tráfego no host atacante, foi executado um comando de ping “ping -f -s 56500 10.0.0.3”, onde o parâmetro –f envia pacotes de forma ágil para cada ECHO_REQUEST enviado é mostrado um “.” no terminal, enquanto que para cada ECHO_REPLY recebido pelo host atacado é impresso um backspace. Isso fornece uma exibição rápida de quanto pacotes estão sendo descartados. Já o parâmetro –s especifica o número de bytes de dados a serem enviados, sendo o padrão 56 bytes, que se traduz em 64 bytes de dados ICMP quando combinados

com os 8 bytes de dados de cabeçalho ICMP. No caso do teste, o número de bytes de dados é 56500, tendo em vista que se o datagrama for muito grande ele será fragmentado, o que diminui o desempenho do sistema atacado.

Com base no tráfego gerado, pode-se verificar que as mensagens de ICMP são de um mesmo host e os dados se repetem, como é mostrado na figura 20.

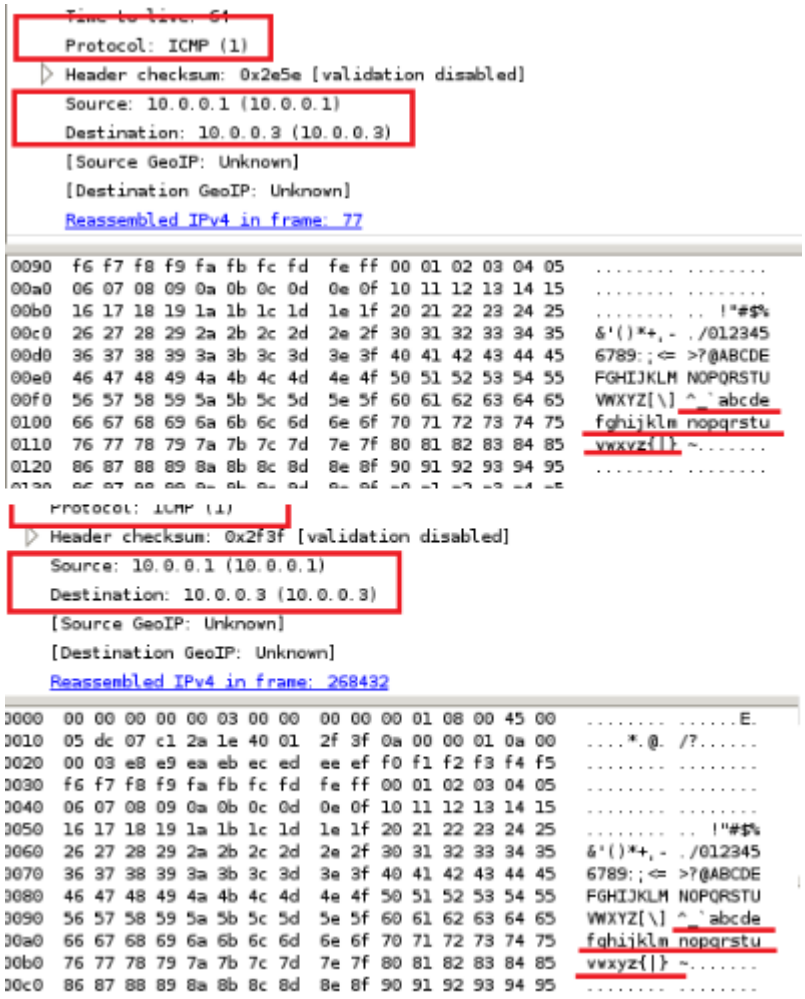

Figura 20 - Datagrama ICMP capturados com wireshark.

Com isso é possível adicionar uma regra que monitora a quantidade de pacotes enviados de um mesmo host. Se esse host passar da quantidade de pacotes enviados em um intervalo de tempo determinado, é gerado um alerta.

Após analisar os dados coletados pelo wireshark, foi definida a regra mostrada na figura 21:

```
alert icmp any any -> any any (msg:"Flood attack"; threshold: type threshold, track by_dst, count 100, seconds 1; sid: 1000001;)
```

Figura 21 - Regra de ICMP Flood.

Onde “alert ICMP any any” significa que é de qualquer origem, “-> any any. O campo “msg: "Flood attack"” é a mensagem que o Snort irá mostrar quando ocorrer uma intrusão. Por fim, os campos “count 100, seconds 1”, categoriza a contagem de pacotes no tempo de 1 segundo de um mesmo destino “track by_dst”.

A função da regra criada é identificar qualquer host de qualquer rede que tente realizar um alto tráfego ICMP a algum host da rede que está sendo monitorada. O padrão para verificar se é um ataque ou um acesso normal é baseado na quantidade de tentativas durante 1 segundo.

5.4.2 Especificação de Regra para ocorrência de intrusão TCP Flooding

O desenvolvimento desta segunda regra se baseia em outro ataque já conhecido, denominado TCP SYN Flood, também denominado de ataque DoS (Denial of Service), que se baseia em uma característica do protocolo TCP/IP, que permite a um host atacante esgotar os recursos da vítima por meio de conexões TCP semiabertas.

Essa conexão TCP acontece em três etapas. Um cliente que deseja iniciar uma conexão em determinada porta aberta pelo servidor, no caso a vítima, envia um SYN. O servidor, por sua vez, responde com um SYN-ACK e para finalizar o processo de estabelecimento de conexão o cliente envia um ACK e a conexão estará pronta para transferência de dados. O ataque TCP SYN Flood ocorre quando

um atacante inicia diversas conexões TCP sem, no entanto, enviar o ACK, obrigando o servidor a aguardar a finalização do processo. Tendo em vista que cada porta TCP possui um número limitado de conexões, o servidor ignora novas conexões até que ocorre timeout e as conexões fiquem inválidas.

A figura 22 mostra a captura dos pacotes via wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
225036	1.511403000	183.213.82.202	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49345 > 0 [SYN] Seq=0 Win=512 Len=0
225037	1.511408000	182.15.157.103	10.0.0.3	TCP	54	49346 > 0 [SYN] Seq=0 Win=512 Len=0
225038	1.511408000	182.15.157.103	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49346 > 0 [SYN] Seq=0 Win=512 Len=0
225039	1.511412000	198.84.249.1	10.0.0.3	TCP	54	49347 > 0 [SYN] Seq=0 Win=512 Len=0
225040	1.511412000	198.84.249.1	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49347 > 0 [SYN] Seq=0 Win=512 Len=0
225041	1.511416000	55.68.255.157	10.0.0.3	TCP	54	49348 > 0 [SYN] Seq=0 Win=512 Len=0
225042	1.511416000	55.68.255.157	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49348 > 0 [SYN] Seq=0 Win=512 Len=0
225043	1.511421000	20.201.105.92	10.0.0.3	TCP	54	49349 > 0 [SYN] Seq=0 Win=512 Len=0
225044	1.511421000	20.201.105.92	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49349 > 0 [SYN] Seq=0 Win=512 Len=0
225045	1.511425000	14.90.183.68	10.0.0.3	TCP	54	49350 > 0 [SYN] Seq=0 Win=512 Len=0
225046	1.511425000	14.90.183.68	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49350 > 0 [SYN] Seq=0 Win=512 Len=0
225047	1.511430000	34.16.126.165	10.0.0.3	TCP	54	49351 > 0 [SYN] Seq=0 Win=512 Len=0
225048	1.511430000	34.16.126.165	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49351 > 0 [SYN] Seq=0 Win=512 Len=0
225049	1.511435000	180.175.218.79	10.0.0.3	TCP	54	49352 > 0 [SYN] Seq=0 Win=512 Len=0
225050	1.511435000	180.175.218.79	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49352 > 0 [SYN] Seq=0 Win=512 Len=0
225051	1.511439000	105.171.185.92	10.0.0.3	TCP	54	49353 > 0 [SYN] Seq=0 Win=512 Len=0
225052	1.511439000	105.171.185.92	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49353 > 0 [SYN] Seq=0 Win=512 Len=0
225053	1.511467000	39.4.84.226	10.0.0.3	TCP	54	49354 > 0 [SYN] Seq=0 Win=512 Len=0
225054	1.511467000	39.4.84.226	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49354 > 0 [SYN] Seq=0 Win=512 Len=0
225055	1.511472000	55.127.55.55	10.0.0.3	TCP	54	49355 > 0 [SYN] Seq=0 Win=512 Len=0
225056	1.511472000	55.127.55.55	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49355 > 0 [SYN] Seq=0 Win=512 Len=0
225057	1.511477000	45.7.222.106	10.0.0.3	TCP	54	49356 > 0 [SYN] Seq=0 Win=512 Len=0
225058	1.511477000	45.7.222.106	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49356 > 0 [SYN] Seq=0 Win=512 Len=0
225059	1.511481000	86.226.61.255	10.0.0.3	TCP	54	49357 > 0 [SYN] Seq=0 Win=512 Len=0
225060	1.511482000	86.226.61.255	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49357 > 0 [SYN] Seq=0 Win=512 Len=0
225061	1.511486000	171.24.193.197	10.0.0.3	TCP	54	49358 > 0 [SYN] Seq=0 Win=512 Len=0
225062	1.511486000	171.24.193.197	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49358 > 0 [SYN] Seq=0 Win=512 Len=0
225063	1.511491000	80.220.13.45	10.0.0.3	TCP	54	49359 > 0 [SYN] Seq=0 Win=512 Len=0
225064	1.511491000	80.220.13.45	10.0.0.3	TCP	54	[TCP Out-Of-Order] 49359 > 0 [SYN] Seq=0 Win=512 Len=0

Figura 22 - Ataque TCP SYN.

Com base no tráfego gerado para os testes de captura com o wireshark, mostrado na figura 23, foi possível observar os padrões de ataque e com isso foi elaborado a segunda regra de alerta no SNORT.

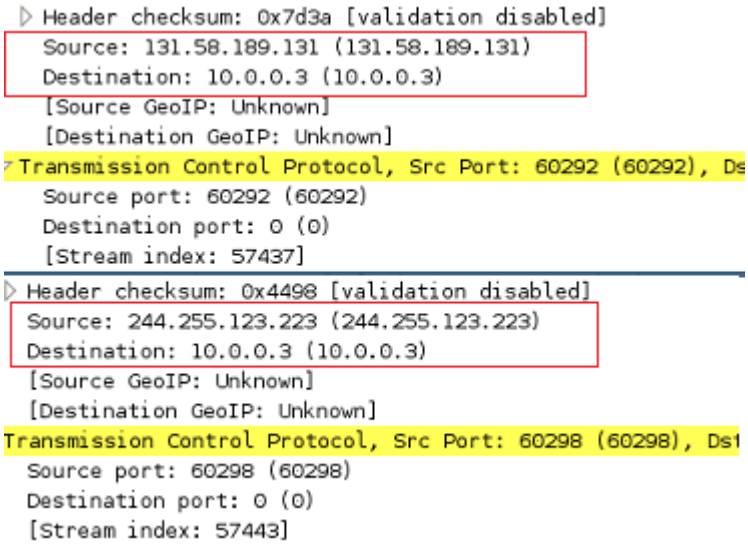

Figura 23 - Datagrama TCP capturado no wireshark.

```
alert tcp any any -> any any (flags: S; msg:"TCP Flood Attack"; flow: stateless; detection_filter: track by_dst, count 50, seconds 1;)
```

Figura 24 - Regra gerada a partir da análise dos pacotes.

A regra mostrada na figura 24 segue o mesmo padrão da que foi criada na subseção anterior. Tendo em vista que o campo “`detection filter`” é rastreada pelo endereço IP de origem ou pelo endereço IP de destino. Isso significa que a contagem é mantida para cada endereço IP de origem exclusivo ou para cada endereço IP de destino exclusivo.

Como o ataque simula IPs de origem diferentes, na figura 24 são mostrados dois exemplos: IPs 131.58.189.131 e 244.255.123.223, o campo "`track`" foi configurado para rastrear a quantidade de SYNs que estão indo para o único destino: “`by_dst`“, o que é feito na regra anterior mesmo ela tendo apenas um IP de origem. Isso permite dar mais flexibilidade as regras caso novos testes sejam efetuados usando mais de uma máquina atacante.

5.5 Resultados dos ataques a rede SDN

As regras criadas na seção 5.4 são caracterizadas como um ataque de negação de serviço (DoS) e tem como foco uma tentativa explícita dos invasores de impedir usuários legítimos de usar algum serviço da rede. O ataque DoS é uma tentativa de fazer com que aconteça uma sobrecarga em um servidor ou computador comum para que recursos do sistema fiquem indisponíveis para seus usuários.

A maioria das abordagens existentes para resolver os problemas gerados por ataques DoS concentra-se na segurança usando soluções, como por exemplo, configuração de firewall, e não nas abordagens de roteamento de pacotes para defender de ameaças DoS, como é mostrado a seguir.

5.5.1 Ataque ICMP Flooding

Conforme já descrito na subseção 5.4.1, o primeiro teste tem como objetivo monitorar o tráfego de ICMP Flooding, para isso as regras de ICMP padrão do Snort foram desabilitadas e deixou-se apenas a regra criada a subseção 5.4.1, em um intervalo de tempo de 5 segundos são gerados diversos alertas de ICMP Flooding. O comando utilizado entre os hosts é o seguinte: “h1 ping –f –s 56500 h3.

A partir do momento que a ferramenta Snort realiza o monitoramento do ambiente SDN e gera os logs de alerta quando ataques são detectados, o controlador mostra os alertas de tais atividades, conforme mostrado na figura 25.

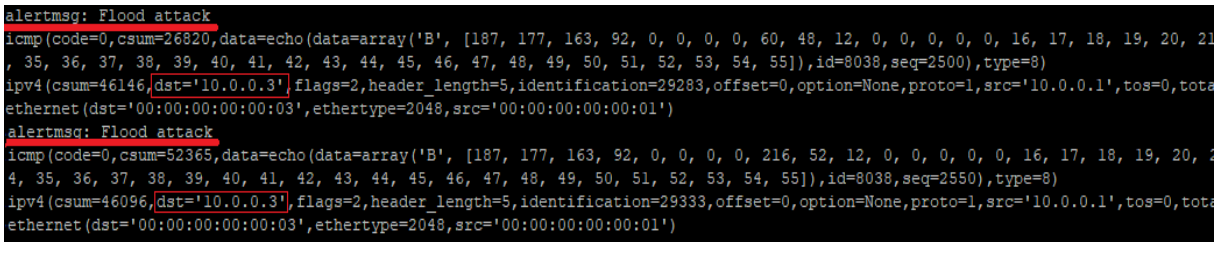

Figura 25 - Saída do controlador RYU.

Assim torna-se possível realizar alterações nos fluxos da rede de forma a mitigar o ataque. Para isso é necessário enviar um comando para alteração da

tabela de fluxo do controlador. Neste sentido o operador da rede SDN deve monitorar o log gerado, e ao identificar os alertas deve enviar a alteração para o controlador bloquear o tráfego malicioso.

```
curl -X POST -d '{"dpid":"id_switch",
"priority":"1", "actions": [{"type": "DROP",
"port":porta_swtich}],"match":{"eth_type":0x080
0,"ip_proto":"1", "ipv4_src":
"ip_ou_rede_atacante"
,"ipv4_dst":"host_vitima"}}' http://localhost:8
080/stats/flowentry/add
```

Figura 26 - Comando para filtrar todo fluxo de um determinado IP.

O comando mostrado na figura 26, realiza uma chamada para o controlador OpenFlow requisitando que todo fluxo entre host atacante (host 1 – IP 10.0.0.1) e vítima (host 3 – IP 10.0.0.3) seja bloqueado, conforme mostrado na figura 27. A partir deste momento todo tráfego destinado ao host 3, será descartados, impossibilitando a comunicação do atacante com a vítima.

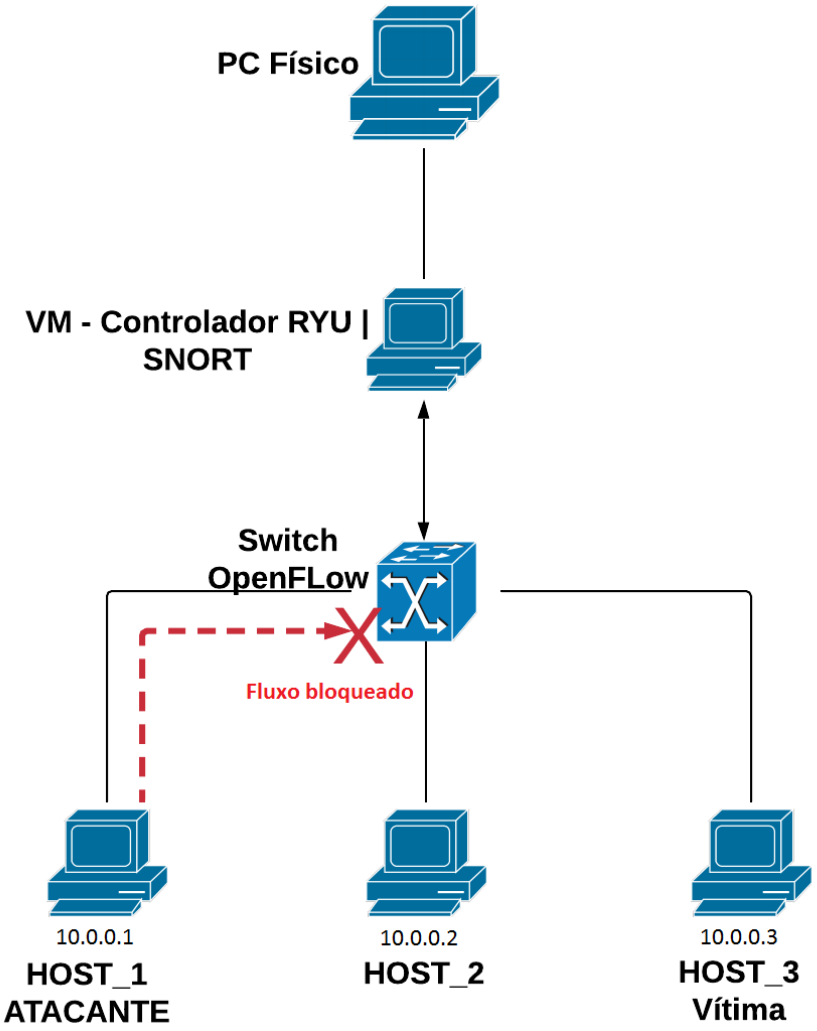

Figura 27 - Bloqueio total da comunicação.

Nesse sentido, podemos bloquear de forma temporária ou permanente o ataque sem impactar os recursos da rede, tais como CPU do switch e do controlador, até que a tentativa de ataque cesse.

5.5.2 Ataque TCP Flooding

O segundo teste a ser realizado foi a identificação do ataque TCP SYN no ambiente simulado, conforme visto na subseção 5.4.2. Para isso, foi executado o seguinte comando: “hping3 –S –flood –rand-source 10.0.0.3”, para gerar um ataque de negação de serviço. Os alertas são ilustrados na Figura 28.

```
alertmsg: TCP Flood attack
ipv4(csum=59724,dst='10.0.0.3',flags=0,header_length=5,identification=13134,offset=0,option=None,proto=6,src='152.161.187.145',tos=0,total_length=40,ttl=64,version=4)
ethernet(dst='00:00:00:00:00:03',ethertype=2048,src='00:00:00:00:00:01')
alertmsg: TCP Flood attack
ipv4(csum=31650,dst='10.0.0.3',flags=0,header_length=5,identification=22546,offset=0,option=None,proto=6,src='218.0.195.24',tos=0,total_length=40,ttl=64,version=4)
ethernet(dst='00:00:00:00:00:03',ethertype=2048,src='00:00:00:00:00:01')
alertmsg: TCP Flood attack
ipv4(csum=56828,dst='10.0.0.3',flags=0,header_length=5,identification=13553,offset=0,option=None,proto=6,src='24.8.69.216',tos=0,total_length=40,ttl=64,version=4)
ethernet(dst='00:00:00:00:00:03',ethertype=2048,src='00:00:00:00:00:01')
```

Figura 28 - Saída do controlador RYU.

Com as regras criadas no Snort, foi possível identificar algumas informações do tráfego malicioso, como por exemplo, IP de destino do ataque, o mac address de quem originou o ataque e o protocolo IP usado.

O tratamento desse ataque foi abordado de forma diferente do primeiro. Foi aplicada a estratégia de desviar o fluxo de pacotes para uma máquina preparada para receber um ataque. A figura 29 ilustra esse senário.

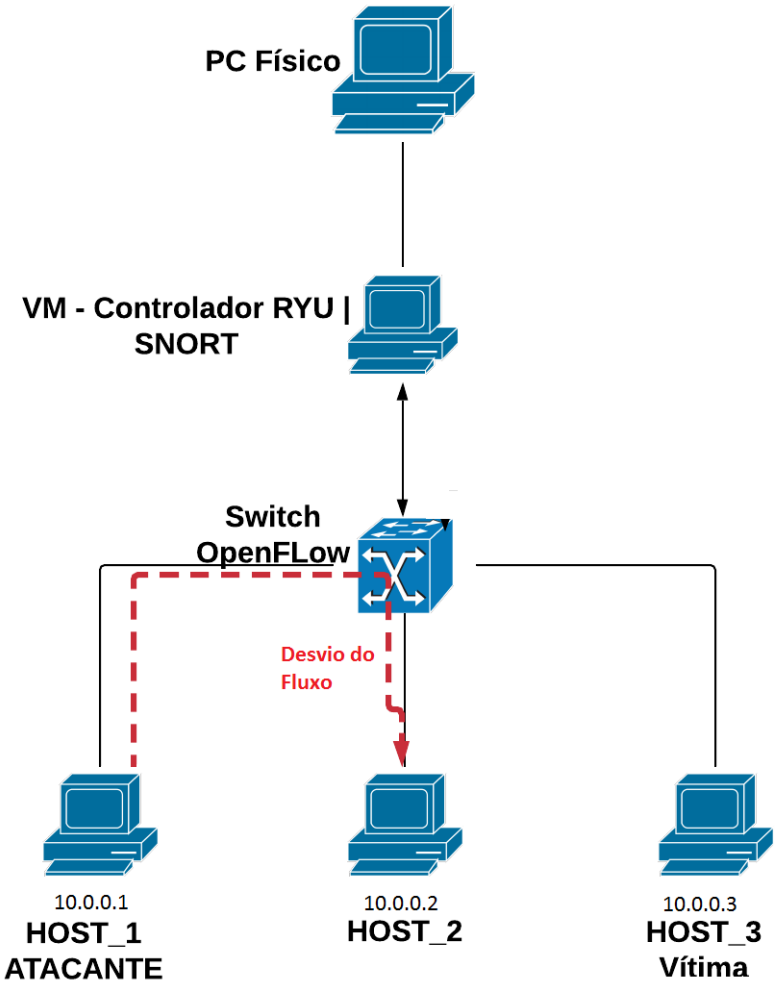

Figura 29 - Desvio do tráfego para um outro host.

Essa abordagem se baseia em desviar o tráfego atacante para uma outra máquina que esteja preparada para receber o fluxo de ataque. Essa forma de mitigação é apresentada por [Dalpissol, Vicentini, Santin, 2018], que mostra resultados satisfatórios, semelhante a medida usada no teste anterior.

Neste sentido, quando o tráfego malicioso for detectado o controlador deve inserir um novo fluxo, que redirecione o tráfego oriundo do host atacante para o host que tem condições técnicas para tratar tal demanda.

Comando para desviar o tráfego para um host previamente preparado é mostrado na figura 30.

```
Curl -d '{"switch": "End_MAC_Switch_Virtual",
"name":"Nome_Regra","priority":"Prioridade_Regra
","src mac":"MAC_Origem" ,"dst-mac":
"MAC_Destino","active":"true","actions":"set-
dst-ip=IP_Redirecionamento,
set-dst-mac=Mac_Redirecionamento,output=
Porta_Redirecionamento"}'
http://IP_Controlador:8080/wm/staticflowentrypu
sher/json
```

Figura 30 - Comando para desviar o fluxo de um determinado IP destino.

O comando mostrado na figura 30 possibilita o desvio do tráfego apenas do host atacante, deixando assim, que os demais hosts acessem o host vítima sem perda de conexão.

6 Conclusões e trabalhos futuros

6.1 Conclusão

A utilização de Redes Definidas por Software integrado com Sistema de Detecção de Intrusão possibilita a elaboração de estratégias de mitigação mais complexas e eficientes unindo os benefícios de flexibilidade e gerência, presentes em SDN, com a possibilidade de elaborar de forma simples novas regras no IDS.

Os experimentos no ambiente simulado no Mininet demonstraram a efetividade da proposta em um cenário controlado. Vale ressaltar que existem diversas formas de mitigar um ataque com o controlador Ryu, caso bem configurado para receber os alertas. Nesse trabalho foi optado por alterar o fluxo via comunicação com a API do controlador.

Os resultados do experimento mostram que o controlador SDN pode filtrar endereços IP suspeitos criando uma lista de bloqueio ou desvio de tráfego pelos resultados do monitoramento integrado entre controlador e IDS. Além disso, a atualização das regras de detecção determinadas pelas assinaturas de ataque apresentadas nesse modelo, pode ser usada como um mecanismo eficiente de detecção de invasões para descobrir conexões de rede suspeitas com base na análise de comportamento anormal no Snort.

6.2 Trabalhos Futuros

Este trabalho apresentou a integração de uma SDN com um IDS como forma de mitigação de ataques. Essa integração tem diversas possibilidades que não foram exploradas nesse trabalho. Com isso, existem diversas frentes de trabalhos futuros:

- Algoritmo para alteração da tabela de fluxo baseado no monitoramento de tráfego analisado.
- Integração de outros controladores SDN com o IDS Snort.
- Integração de outros IDS com o controlador RYU.

- Simulação do mesmo experimento usando o novo framework P4 (Programming Protocol-Independent Packet Processors), que propõe substituir o OpenFlow futuramente.
- Simular a detecção de outras formas de ataque.
- Utilizar a tecnologia NFV para virtualização do controlador e switch.
- Utilização do mesmo cenário usando um IPS (Intrusion Prevention System).

7 Referência

AJAEIYA, et al. Flow-based Intrusion Detection System for SDN. IEEE Symposium on Computers and Communications (ISCC). 2017.

BRANDT, Markus. et al. Security Analysis of Software Defined Networking Protocols OpenFlow, OF-Config and OVSDB. In: The 2014 IEEE Fifth International Conference on Communications and Electronics (ICCE 2014), DA NANG, Vietnam. [S.l.: s.n.], 2014.

CAMPOS, M. e Martins, J. 2017. Uma proposta de arquitetura de segurança para a detecção e reação a ameaças em redes SDN. Revista Brasileira de Computação Aplicada. 9, 1 (maio 2017), 107-119.

CARVALHO, L. Fernando. 2018. Um Ecossistema para Detecção e Mitigação de Anomalias em Redes Definidas por Software. Tese de Doutorado - Universidade Estadual de Campinas.

CENTENO, PAULO VIEIRA. Uma análise de segurança de Redes Definidas por Sorfware sobre o protocolo OpenFlow. 2016, 83 (Defesa Conclusão de curso) - Universidade Federal de Santa Catarina. Florianópolis. 2016.

COSTA, V. T.; M. K. Costa, L. H. Vulnerabilities and solutions for isolation in flowvisor-based virtual network environments. Journal of Internet Services and Applications, v. 6, n. 1, p. 18, 2015.

DALPISSOL, Laura, Vicentini, J. A. Cleverton, Santin, O. Altair. (2018). Integração de Detecção de Intrusão em Redes Definidas por Software. Anais Estendidos do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg), [S.l.], p. 253 - 261, oct. 2018.

FEAMSTER, Nick; REXFORD, Jennifer; ZEGURA, Ellen, (2014). The road to SDN: An intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, ACM, v. 44, n. 2, p. 87 – 98.

FERNANDES, EDER Leao; ROTHENBERG, Christian Esteve. OpenFlow 1.3 Software Switch, In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 32, 2014. Florianópolis. Resumo... Florianópolis: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2014. 8 p.

FERNANDES, Henrique Santos. (2017). Provendo segurança em redes definidas por software através da integração com sistemas de detecção e prevenção de intrusão. Dissetação de Mestrado - Universidade Federal Fluminense.

GITHUB Mininet. Introduction to Mininet . Disponível em: <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>. Acesso em: 06/03/2019.

GOLVEIA, Feliz Ribeiro. (2017). Mecanismos de Deteção de Intrusão – OSSEC HIDS. Faculdade de Ciência e Tecnologia Departamento de Ciências da Engenharia e da Arquitectura FCT (DCEA). Dissertações de Mestrado.

GUEDES, D. et al. (2012). Redes Definidas por Software: uma abordagem sistêmica para o desenvolvimento das pesquisas em Redes de Computadores. GTA-UFRJ.

GUDE,N., et al, (2008) "NOX: Towards an operating system for networks", ACM SIGCOMM Computer Communication Review

GUO, I. et al. (2016). Security foundation requirements for sdn controllers. Relatório Técnico TR-529, ONF.

HAKIRI, A.; Gokhale, A.; Berthou, P.; Schmidt, D. C.; Gayraud, T. Softwaredefined networking: Challenges and research opportunities for future internet. Computer Networks, v. 75, Part A, p. 453 – 471, 2014.

HAKIRI, A.; et al. (2014). Software-defined Networking: Challenges and Research Opportunities for Future Internet, Computer Networks, v. 75, Part A, p. 453-471.

ZHANG, H. et al.A Survey on Security-Aware Measurement in SDN.Security and Communication Networks Volume 2018, Article ID 2459154, 14 pages

HU, F; Hoa, Q e Bao Ke. (2014). A Survey on Software-Defined Network and OpenFlow: From concept to implementation. IEEE Communication Survey & Tutorials, 16(4), 2181-2206

J. Lin. Github Pigrelay. Disponível em https://github.com/John-Lin/pigrelay. Acesso em 25/03/2019.

KREUTZ, Diego. et al., (2015). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, IEEE, v. 103, n. 1, p. 14 – 76.

KREUTZ, D.; Ramos, F. M. V.; Veríssimo, P. E.; Rothenberg, C. E.; Azodolmolky,S.; Uhlig, S. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, v. 103, n. 1, p. 14–76, 2015.

KULKARNI, Gaurav. (2017). Simplification of Internet Ossification through Software Defined Network Approach.Global Journal Of Computer Science ans Technology: C Software & Data Engineering, Volume 17 Issue 3 Version 1.0

LAMB, C. C.; Heileman, G. L. Towards robust trust in software defined networks. In: 2014 IEEE Globecom Workshops (GC Wkshps), 2014, p. 166–171.

LANTZ, Bob; HELLER, Brandon; MCKEOWN, Nick. (2010) A network in a laptop: rapid prototyping for software-defined networks. In: ACM. Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. [S.l.].

MANSO, Moura, Serrão, (2019). SDN-Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks. Information. 10. 10.3390/info10030106.

MCKEOWN, N. (2009). Software-defined networking. INFOCOM keynote talk, Apr.

MCKEOWN, N., et al. (2008). Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev., 38:69–74

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., v. 38, n. 2, p. 69–74, 2008.

MORIMOTO, Carlos. E. (2011), Redes Guia Pratico, 2ª Edição. Ed. Sul Editores, Editora Sulina.

MTIBAA, A., Harras, K. A., and Alnuweiri, H. (2015). From botnets to mobibots: A novel malicious communication paradigm for mobile botnets. IEEE Communications Magazine, 53(8):61–67

NUNES, B. A. A.; Mendonca, M.; Nguyen, X.-N.; Obraczka, K.; Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications Surveys & Tutorials, v. 16, n. 3, p. 1617-1634.

ONF - Open Networking Fundation. Software-Defined Networking: The New Norm for Netwoks. ONF White Paper. Palo Alto, US: Open Networking Foundation, 2015.

ORTIZ, Viger T. (2018). Análise Experimental de Segurança de Controladores de Código Aberto para Redes Definidas Por Software. Dissertação de Mestrado - Universidade Federal de São Paulo, São Paulo.

OKTIAN, Y. E.; Lee, S.; Lee, H.; Lam, J. Secure your northbound sdn api. In: 2015 Seventh International Conference on Ubiquitous and Future Networks, 2015, p. 919–920.

PUSHPA, J. Research, Pethuru, Raj, (2018). Topology-based analysis of performance evaluation of centralized vs. distributed SDN controller. IEEE International Conference on Current Trends in Advanced Computing (ICCTAC). Bangalore, India.

RYU v.4.30, (2018). Open Source Project. http://osrg.github.io/ryu/.

SHERWOOD, R. et al. (2009). FlowVisor: A Network Virtualization Layer. Technical Report Openflow-tr-2009-1, Stanford University.

SNORT. The Snort Project. Disponível em: <http://www.snort.org/>. Acesso em: 06/03/2019.

TIWARI, VARUN; PAREKH Rushit; PATEL, Vishal. A Survey on Vulnerabilities of Openflow Network and its Impacton SDN/Openflow Controller. World Academics Journal of Engineering Sciences, Department of Computer Science RCC, Gujarat University Ahmedabad, India, 1005, 5, 2014.

VACCA, J. Computer and Information Security Handbook, Second Edition. 2nd.ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013.

Zanna, P., O'Neill, B., Radcliffe, P., Hosseini, S., and Hoque, M. S. U. (2014). Adaptive threat management through the integration of IDS into software defined networks. In International Conference on the Network of the Future (NOF) - Workshop on Smart Cloud Networks & Systems, pages 1–5.

Integração entre Rede Definida por Software e Sistema de Detecção de Intrusão para Mitigação de Ataques

Clailton Francisco[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)

{clailton.francisco}@grad.ufsc.br

Abstract. *With the advantages that a SDN provides, it is possible to incorporate new and old techniques to minimize the risk of attacks and increase network security. SDN makes network management and maintenance more efficient and cost effective. It also provides better quality of service and minimizes network failure points.*

Attack detection prevention systems are essential for the security of the computer network. Monitoring real-time network traffic in search of intruders to secure a trusted network is one of their roles, and in in front of the integration between a SDN and Attack Detection and Prevention Systems, this study presents a form of attack mitigation using Software Defined Network advances with the OpenFlow protocol, based on the integration of technologies already established in the conventional network architecture.

Resumo. *Com as vantagens que uma Rede Definida por Software (SDN) proporcionam é possível incorporar novas e velhas técnicas para minimizar os riscos de ataques e aumentar a segurança da rede. SDN torna o gerenciamento e manutenção da rede mais eficiente e com menor custo operacional. Proporciona também uma melhor qualidade de serviço e minimizando os pontos de falha da rede.*

Os sistemas de prevenção de detecção de ataques são essenciais para a segurança da rede de computadores. Monitorar o tráfego da rede em tempo real em busca de intrusos para garantir uma rede confiável é um dos seus papéis. E diante da integração entre uma Rede Definida por Software e os Sistemas de Detecção e Prevenção de ataques, este estudo apresenta uma forma de mitigação de ataque usando os avanços da Rede Definida por Software, com o protocolo OpenFlow, tendo como base a integração de tecnologias já estabelecidas na arquitetura de rede convencional.

1. Introdução

Com a atual expansão da computação em rede a comunidade da área se encontra em uma complexa situação. O crescimento da computação proporcionou grandes avanços e está levando a infraestrutura atual ao seu limite, fazendo com que as ferramentas tradicionalmente usadas não sejam mais suficientes para suprir o alto nível

de complexidade exigida [TIWARI, VARUN; PAREKH RUSHIT; PATEL, VISHAL, 2014].

Com o advento do conceito de Redes Definidas por Software (SND - Software Defined Network), surgiu a possibilidade de programar uma rede semelhante a um computador. O OpenFlow, ou qualquer API (Application Programming Interface) que forneça uma camada de abstração da rede física para o elemento de controle, permite que a rede seja configurada ou manipulada através de software, o que abre possibilidade para maiores inovações [FERNANDES, EDER LEAO; ROTHENBERG, CHRISTIAN ESTEVE, 2014]. Isso faz com que as ferramentas atuais para gerencia de recursos de infraestrutura, que apresentam rigidez na política de mudanças de alto nível tenham um alto nível de complexidade para sua manutenção e sejam progressivamente substituídas.

A SDN fornece abstração em três áreas da rede: estado distribuído, encaminhamento e configuração [MCKEOWN, N. et al. 2008]. Com o OpenFlow/SDN, os usuários podem personalizar as redes de acordo com as necessidades locais, eliminar ferramentas desnecessárias e criar redes virtuais e isoladas. Com isso pode-se aumentar o ritmo da inovação através de software, em vez de hardware.

Em grande escala, a gerência dos recursos da rede de forma independente se torna cada vez mais complicada. Além disso, plataformas legadas, que já não possuem suporte e não podem ser atualizadas quando brechas em sua implementação são expostas, geraram a necessidade de uma forma externa de identificação de ataque. Nesse contexto, foram pensadas em ferramentas que identifiquem tentativas de ataque a estes sistemas através de padrões de comportamento de rede observados. Essas ferramentas são chamadas de IDS (Sistemas de Detecção de Intrusão), quem estão aptos a monitorar a rede com o intuito de detectar ações intrusivas na rede.

2. Redes Definidas por Software (SDN)

A infraestrutura de rede se encontra em uma situação complexa: com o sucesso e avanço da área de rede, existem muitas aplicações que funcionam sobre uma estrutura de switches, roteadores, firewalls, servidores e etc., que necessitam de características como: baixa latência, flexibilidade, confiabilidade, segurança e escalabilidade.

Desde o momento em que as redes passaram a empregar equipamentos como switches e roteadores para encaminhar pacotes, a estrutura pouco se alterou [CENTENO, 2016]. Fabricantes criam equipamentos com arquiteturas proprietárias e fechadas, fazendo com que o software esteja encapsulado no hardware, tornando-os uma única "entidade". A forma como a rede evoluiu, criou como base um emaranhado de protocolos e camadas de equipamentos, que mesmo permitindo a comunicação da forma como conhecemos hoje, ergueu-se uma enorme barreira para inovações e experimentações necessárias à área de redes de computadores. Qualquer inovação representa um processo de alta complexidade

Essas características fechadas, levantou o questionamento se a arquitetura de redes de computadores de modo geral atingiu um nível de amadurecimento que as tornaram pouco flexíveis. O jargão usado em muitos casos é que a Internet está

calcificada, fazendo referência a pouca flexibilidade para mudanças na arquitetura de rede [KULKARNI, 2017].

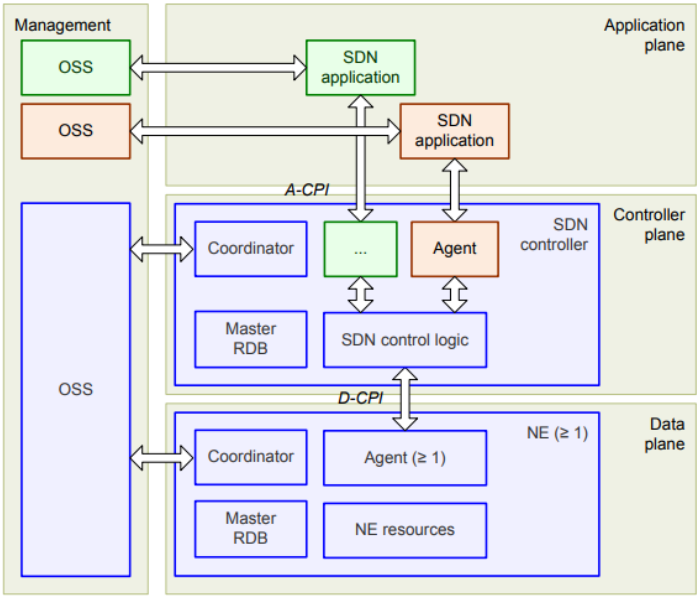

Figura 1 - Visão geral da arquitetura SND, com plano de dados e físico.

[Fernandes 2014]

2.1. Protocolo OpenFlow

Em uma SDN, a interface de comunicação entre os comutadores e controladores ocorre por meio de uma interface de programação. O OpenFlow é o protocolo padrão para SDN e tem como ideia básica utilizar funções de manipulação das tabelas de fluxos, que são oferecidas nos comutadores. A ONF (Open Network Foundation), fundada em 2011, é uma organização com o objetivo de gerenciar as especificações do OpenFlow e promover o uso e desenvolvimento de soluções abertas para SDN [MCKEOWN, 2008].

O plano de dados de um Comutador OpenFlow consiste em uma Tabela de Fluxos com uma ação correspondente a cada um desses fluxos. Portanto, isso permite que os fluxos sejam tratados de maneira diferente. Assim, pode ser definido um fluxo de tráfego experimental e outro de tráfego de produção, possibilitando isolamento entre eles. Esse tráfego de produção é tratado da mesma forma que seria na ausência do OpenFlow. Com isso, o OpenFlow pode ser visto com uma generalização do conceito de VLANs, que também permitem o isolamento de tráfego, mas de forma menos flexível [MCKEOWN, 2008].

2.2. Comutador OpenFlow

Um Comutador OpenFlow é composto por, no mínimo, três partes. A primeira consiste na Tabela de Fluxos, que possui uma ação para cada fluxo definido, como descrito anteriormente. A segunda consiste no Canal Seguro que conecta comutador e controlador da rede. O Controlador é utilizado para a configuração da Tabela de Fluxos. A terceira parte consiste no Protocolo OpenFlow, que possibilita a comunicação do Comutador com o Controlador. Com isso os desenvolvedores podem configurar a Tabela de Fluxos sem a necessidade de programar o comutador [CENTENO, 2016].

Existem dois tipos de comutadores OpenFlow. O primeiro tipo consiste em um comutador ou roteador com OpenFlow habilitado que, além de suportar as funcionalidades do OpenFlow, realiza as funções comuns de um comutador ou roteador. O segundo consiste em um comutador OpenFlow dedicado, que não suporta encaminhamento comum de camada 2 e camada 3 [MCKEOWN, 2008]. A figura 2 exemplifica os componentes desses dois tipos.

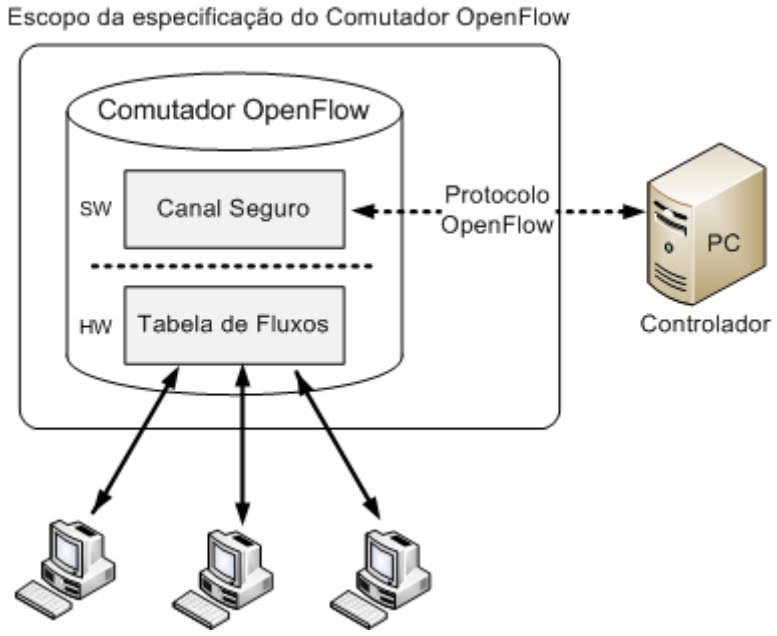

Figura 2 - Comutador OpenFlow.

2.3. Controladores SDN

Ao remover o poder de decisão dos elementos que compõem as redes, quem assume essa responsabilidade é o controlador SDN. No caso do OpenFlow, o responsável pelas decisões é o controlador OpenFlow. Trata-se de um software, que implementa o protocolo OpenFlow, tornando viável o gerenciamento centralizado da rede. Ele é o responsável por monitorar, gerenciar e tomar decisões referente à rede de dados. Pacotes não reconhecidos na tabela de fluxos dos switches OpenFlow são

enviados para o controlador, para que este possa tratar e tomar alguma ação levando em consideração a lógica pré-programada.

3. Segurança em rede SDN

Por padrão uma SDN não possui ferramenta para detecção de intrusão, o que por consequência, a torna vulnerável à maioria dos ataques aos quais as redes tradicionais são vulneráveis [Ajaeiya et al. 2017].

Embora a separação do plano de controle e de dados na SDN permita aos desenvolvedores criar novas ferramentas de segurança, o uso integrado de outras ferramentas de detecção de intrusão (IDS) traz novas formas e soluções mais robustas, aumentando a flexibilização da arquitetura SDN, já que com o plano de controle possibilita o redirecionamento de pacotes no caso de uma detecção de ataque, por exemplo [Dalpissol, Vicentini, Santin, 2018].

[Zanna et al. 2014] mostraram que é possível integrar um sistema de detecção de intrusão (IDS) com um controlador SDN para realizar detecção e bloqueio de ataques de negação de serviço.

4. Ataques de Intrusão

Ataque de intrusão pode ser caracterizado como um conjunto de ações que tentam comprometer recursos de um sistema de computador ou diversas tentativas de explorar informações de qualquer natureza. Estes ataques têm como objetivo corromper três componentes básicos da segurança de informação [Campos, M. e Martins, J. 2017].

Esse tipo de ataque pode agir de diversas formas baseando-se em falhas de segurança de sistemas, aplicações, protocolos ou relacionado a falhas de configurações. Também podem ocorrer nas diferentes camadas do modelo TCP/IP e, de acordo com a camada que atuam, esses ataques são classificados em quatro grupos.

- **Negação de Serviços ou Denial of Service - DoS**: esse tipo de ataque tem como objetivo gera indisponibilidade de recursos em um determinado sistema ou equipamento, como consumo excessivo de memória, processador ou rede.
- **Remoto para Usuário**: correspondem a situações em que o intruso possui conectividade com a máquina vítima, sem possuir uma conta de usuário e explorando alguma vulnerabilidade existente, consegue obter acesso local à máquina. Ataques da classe reconhecimento (Probing) são normalmente empregados em uma etapa que antecede o ataque ou intrusão.
- **Para Super Usuário**: englobam todos os ataques em que o intruso possui acesso ao sistema como um usuário normal e consegue elevar seu nível de privilégio para o de um usuário especial (como o usuário root em sistemas Unix ou administrador em outras plataformas).

- **Exploração**: busca explorar vulnerabilidades do sistema, de aplicações instaladas sobre o sistema ou ainda falhas do usuário que permitam ataque futuros. É normalmente empregado em uma etapa que precede outros tipos de ataque.

5. Ambiente de Simulação

Para viabilizar os testes, será usado o ambiente de simulação Mininet, que permite emular componentes típicos de arquiteturas SDN. O controlador Ryu e o IDS SNORT estão inseridos na mesma máquina para facilitar a integração dessas duas ferramentas.

As regras dentro da ferramenta SNORT foram desenvolvidas para identificar intrusões dentro de um ambiente de rede controlado.

Esse ambiente é mostrado na figura 3, que evidencia como todos os componentes da rede são simulados no Mininet em uma máquina virtual disponibilizada no site oficial do Mininet. Nessa VM o ambiente já dispões de todos os pacotes necessários para que se crie uma rede SDN, sendo necessário apenas baixar os pacotes do controlador escolhido, que nesse caso foi o controlador RYU.

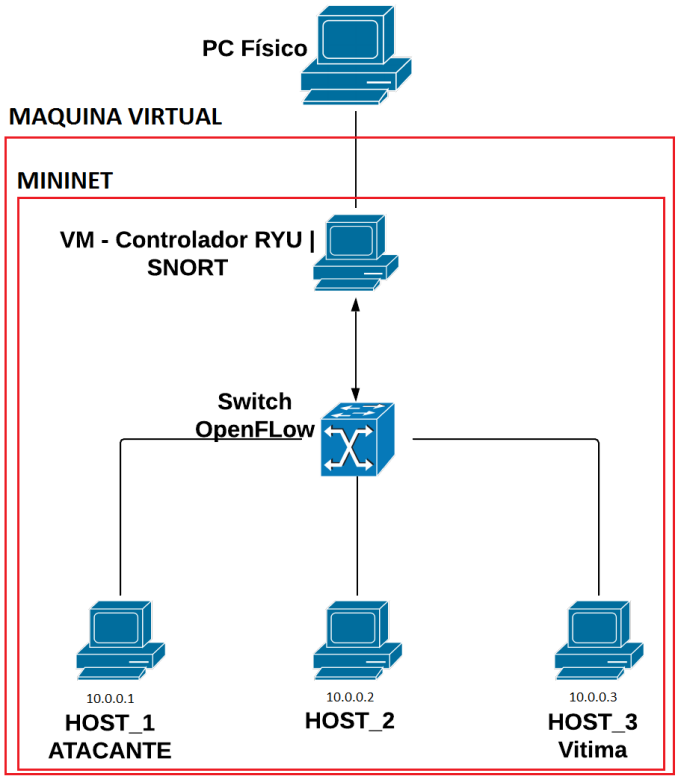

Figura 3 - Ambiente de teste.

5.1. Mininet

Para o ambiente de simulação, foi usada uma máquina virtual disponível no site oficial do Mininet1. Essa é uma maneira simples já que todas as dependências de pacotes, binários ferramentas do OpenFlow já estão pré-instaladas e o Kernel também está ajustado para os testes necessários nesse trabalho.

Mininet é um sistema que permite uma prototipação rápida e simulação de Rede Definida por Software, com ferramentas para criação de controladores de rede, permitindo o desenvolvimento de aplicações com o OpenFlow.

Conforme descrito pelo projeto Mininet [GITHUB Mininet, 2019], os equipamentos virtuais, hosts, switches, links, e controladores são reais, apenas implementados em software, e não hardware. Portanto a performance dos equipamentos está limitada ao computador que está realizando a simulação.

Cada host virtual criado pelo Mininet pode executar programas, ler e escrever arquivos, etc. Isso torna as possibilidades de experimentos muito amplas [GITHUB Mininet, 2019].

5.2. Controlador RYU

Para o controlador SDN foi optado pelo RYU já que ele dispõe de suporte para as versões mais atuais do OpenFlow. Fornece componentes de software com APIs bem definidas que facilitam o gerenciamento de rede e também foi escrito em python [RYU, 2019].

No decorrer do levantamento de requisitos para o desenvolvimento do trabalho, foi cogitado usar o controlador NOX. Ele foi o primeiro controlador SDN desenvolvido. Inicialmente suportava as linguagens C++ e Python, mas as versões foram separadas: a versão Python tem o nome de POX e a versão C++ ficou como NOX. Desde que foi criado, o NOX era bastante adotado e foi usado como base para outros controladores, mas seu uso está em grande queda frente a novos controladores. Uma das razões é o fato deste controlador não suportar as versões mais recentes do OpenFlow, pois suporta apenas até a versão 1.0.

5.3. Sistema de Detecção de Intrusão SNORT

Snort é uma ferramenta IDS de código-fonte aberto, desenvolvida por Martin Roesch, sendo muito popular pela sua flexibilidade nas configurações de regras, sua capacidade de fazer análise de tráfego em tempo real e constante atualização. Seu

[1] http://mininet.org/download/

código fonte é desenvolvido em módulos utilizando a linguagem C possuindo documentação de domínio público [SNORT, 2019].

Uma das vantagens do Snort é permitir que o usuário produza suas próprias regras. Estas regras constituem a parte mais importante do IDS que podem levar ao sucesso ou não da detecção das intrusões. Esta ferramenta utiliza uma linguagem de descrição simples e fácil de usar, assim permitindo customizar as regras de acordo com as características do ambiente no qual deseja trabalhar, refinando ou ampliando o poder de detecção do Snort [SNORT, 2019].

Um dos diferenciais que uma rede SDN em conjunto com o Snort podem trazer é que os bloqueios dos pacotes podem ser feitos de forma mais eficiente. Com as redes tradicionais, o bloqueio de pacotes é feito, em sua grande maioria, nos firewalls da rede, localizados na borda da rede. Com isso, existe muito tráfego malicioso na rede utilizando banda desnecessariamente em switches que não tem a capacidade de bloquear os mesmos.

5.4. Integração SNORT e Controlador RYU

A forma de fazer a integração foi uma decisão importante para o andamento deste trabalho. A falta de integração entre os controladores disponíveis com dispositivos de segurança, mais especificamente com o IDS Snort, para detecção de ataques de intrusão, foi o motivo para a escolha do controlador Ryu.

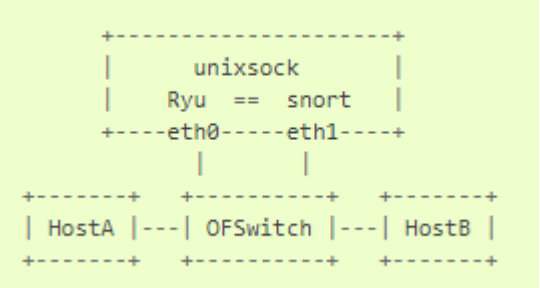

Figura 4 - Arquitetura em uma única maquina.

[RYU, 2019].

6. Resultados

Para gerar o ambiente de teste deste trabalho, foi utilizado o controlador SDN e IDS na mesma máquina virtual e por isso foi necessário criar uma topologia de rede na ferramenta Mininet. Sendo assim, foi criada uma topologia, composta por três hosts, um

switch e um controlador, todos instanciados através do Mininet, com o comando da figura 5.

```
sudo mn --topo single,3 --mac --controller
remote --switch ovsk,protocols=OpenFlow13
```

Figura 5 - Comando para criação da topologia no mininet.

Esse comando cria uma topologia contendo três hosts e um único switch do tipo openvSwitch, define os endereços MAC de cada host, igual ao seu IP, define um controlador remoto cujo padrão é o localhost e aplica para esse controlador a versão 1.3 do protocolo OpenFlow.

6.1. Mitigação de Ataque ICMP Flooding

O primeiro teste tem como objetivo monitorar o tráfego de ICMP Flooding, para isso as regras de ICMP padrão do Snort foram desabilitadas e deixou-se apenas a regra criada a subseção 5.4.1, em um intervalo de tempo de 5 segundos são gerados diversos alertas de ICMP Flooding. O comando utilizado entre os hosts é o seguinte: “h1 ping –f –s 56500 h3.

A partir do momento que a ferramenta Snort realiza o monitoramento do ambiente SDN e gera os logs de alerta quando ataques são detectados, o controlador mostra os alertas de tais atividades, conforme mostrado na figura 6.

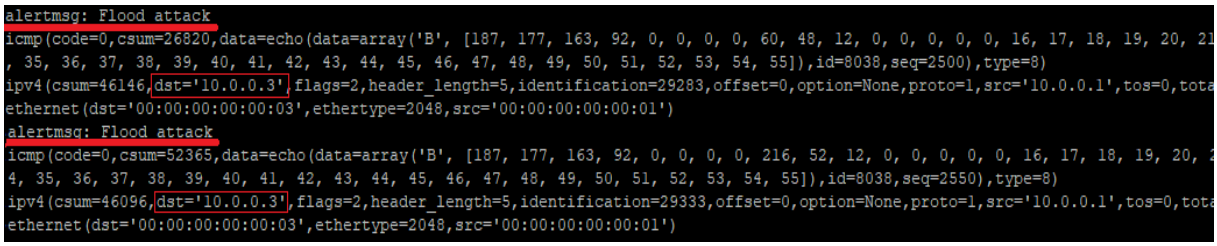

Figura 6 - Saída do controlador RYU.

Assim torna-se possível realizar alterações nos fluxos da rede de forma a mitigar o ataque. Para isso é necessário enviar um comando para alteração da tabela de fluxo do controlador. Neste sentido o operador da rede SDN deve monitorar o log gerado, e ao identificar os alertas deve enviar a alteração para o controlador bloquear o tráfego malicioso.

```
curl -X POST -d '{"dpid":"id_switch", "priority":"1",
"actions": [{"type": "DROP",
"port":porta_swtich}],"match":{"eth_type":0x0800,"ip_proto"
:"1", "ipv4_src": "ip_ou_rede_atacante"
,"ipv4_dst":"host_vitima"}}' http://localhost:8080/stats/fl
owentry/add
```

Figura 7 - Comando para filtrar todo fluxo de um determinado IP.

O comando mostrado na figura 7, realiza uma chamada para o controlador OpenFlow requisitando que todo fluxo entre host atacante (host 1 – IP 10.0.0.1) e vítima (host 3 – IP 10.0.0.3) seja bloqueado, conforme mostrado na figura 8. A partir deste momento todo tráfego destinado ao host 3, será descartado, impossibilitando a comunicação do atacante com a vítima.

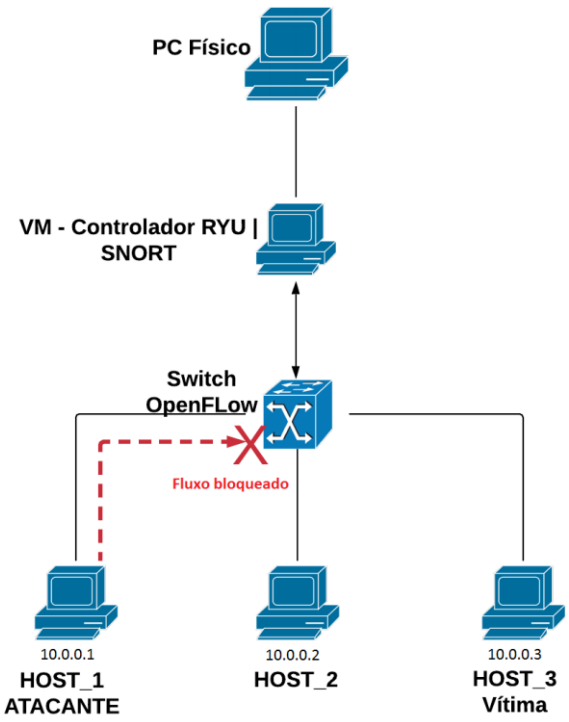

Figura 8 - Bloqueio total da comunicação

6.2 Ataque TCP Flooding

O segundo teste a ser realizado foi a identificação do ataque TCP SYN no ambiente simulado, conforme visto na subseção 5.4.2. Para isso, foi executado o seguinte comando: “hping3 –S –flood –rand-source 10.0.0.3”, para gerar um ataque de negação de serviço. Os alertas são ilustrados na Figura 9.

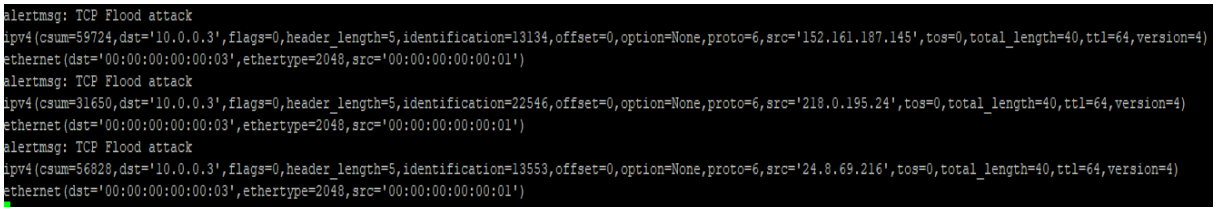

Figura 9 - Saída do controlador RYU.

Com as regras criadas no Snort, foi possível identificar algumas informações do tráfego malicioso, como por exemplo, IP de destino do ataque, o mac address de quem originou o ataque e o protocolo IP usado.

O tratamento desse ataque foi abordado de forma diferente do primeiro. Foi aplicada a estratégia de desviar o fluxo de pacotes para uma máquina preparada para receber um ataque. A figura 10 ilustra esse senário.

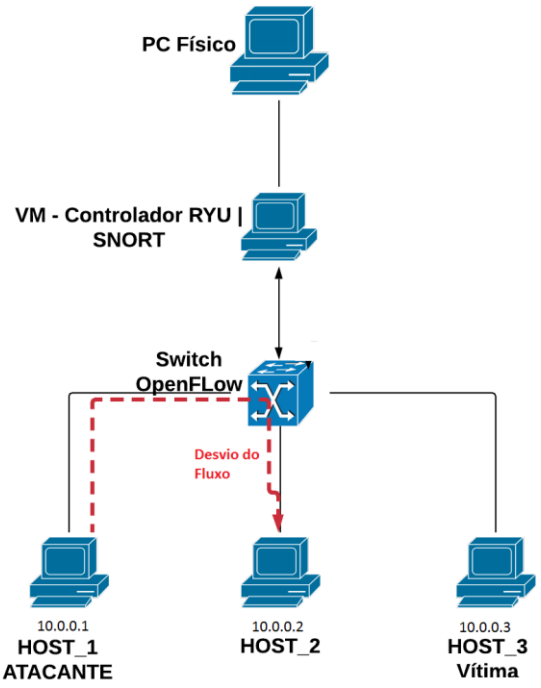

Figura 10 - Desvio do tráfego para um outro host.

Essa abordagem se baseia em desviar o tráfego atacante para uma outra máquina que esteja preparada para receber o fluxo de ataque. Essa forma de mitigação é apresentada por [Dalpissol, Vicentini, Santin, 2018], que mostra resultados satisfatórios, semelhante a medida usada no teste anterior.

Neste sentido, quando o tráfego malicioso for detectado o controlador deve inserir um novo fluxo, que redirecione o tráfego oriundo do host atacante para o host que tem condições técnicas para tratar tal demanda.

Comando para desviar o tráfego para um host previamente preparado é mostrado na figura 11.

```
Curl -d '{"switch": "End_MAC_Switch_Virtual",
"name":"Nome_Regra","priority":"Prioridade_Regra","src
mac":"MAC_Origem" ,"dst-mac":
"MAC_Destino","active":"true","actions":"set-dst-
ip=IP_Redirecionamento,
set-dst-mac=Mac_Redirecionamento,output=
Porta_Redirecionamento"}'
http://IP_Controlador:8080/wm/staticflowentrypusher/json
```

Figura 11 - Comando para desviar o fluxo de um determinado IP destino.

O comando mostrado na figura 30 possibilita o desvio do tráfego apenas do host atacante, deixando assim, que os demais hosts acessem o host vítima sem perda de conexão.

7. Conclusão

A utilização de Redes Definidas por Software integrado com Sistema de Detecção de Intrusão possibilita a elaboração de estratégias de mitigação mais complexas e eficientes unindo os benefícios de flexibilidade e gerência, presentes em SDN, com a possibilidade de elaborar de forma simples novas regras no IDS.

Os experimentos no ambiente simulado no Mininet demonstraram a efetividade da proposta em um cenário controlado. Vale ressaltar que existem diversas formas de mitigar um ataque com o controlador Ryu, caso bem configurado para receber os alertas. Nesse trabalho foi optado por alterar o fluxo via comunicação com a API do controlador.

Os resultados do experimento mostram que o controlador SDN pode filtrar endereços IP suspeitos criando uma lista de bloqueio ou desvio de tráfego pelos resultados do monitoramento integrado entre controlador e IDS. Além disso, a atualização das regras de detecção determinadas pelas assinaturas de ataque apresentadas nesse modelo, pode ser usada como um mecanismo eficiente de detecção de invasões para descobrir conexões de rede suspeitas com base na análise de comportamento anormal no Snort.

References

AJAEIYA, et al. Flow-based Intrusion Detection System for SDN. IEEE Symposium on Computers and Communications (ISCC). 2017.

CAMPOS, M. e Martins, J. 2017. Uma proposta de arquitetura de segurança para a detecção e reação a ameaças em redes SDN. Revista Brasileira de Computação Aplicada. 9, 1 (maio 2017), 107-119.

CENTENO, PAULO VIEIRA. Uma análise de segurança de Redes Definidas por Sorfware sobre o protocolo OpenFlow. 2016, 83 (Defesa Conclusão de curso) - Universidade Federal de Santa Catarina. Florianópolis. 2016.

DALPISSOL, Laura, Vicentini, J. A. Cleverton, Santin, O. Altair. (2018). Integração de Detecção de Intrusão em Redes Definidas por Software. Anais Estendidos do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg), [S.l.], p. 253 - 261, oct. 2018.

FERNANDES, EDER Leao; ROTHENBERG, Christian Esteve. OpenFlow 1.3 Software Switch, In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 32, 2014. Florianópolis. Resumo... Florianópolis: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2014. 8 p.

FERNANDES, Henrique Santos. (2017). Provendo segurança em redes definidas por software através da integração com sistemas de detecção e prevenção de intrusão. Dissetação de Mestrado - Universidade Federal Fluminense.

GITHUB Mininet. Introduction to Mininet . Disponível em: <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>. Acesso em: 06/03/2019.

KULKARNI, Gaurav. (2017). Simplification of Internet Ossification through Software Defined Network Approach.Global Journal Of Computer Science ans Technology: C Software & Data Engineering, Volume 17 Issue 3 Version 1.0

MCKEOWN, N. (2009). Software-defined networking. INFOCOM keynote talk, Apr.

MCKEOWN, N., et al. (2008). Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev., 38:69–74

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., v. 38, n. 2, p. 69–74, 2008.

RYU v.4.30, (2018). Open Source Project. http://osrg.github.io/ryu/.

SNORT. The Snort Project. Disponível em: <http://www.snort.org/>. Acesso em: 06/03/2019.

TIWARI, VARUN; PAREKH Rushit; PATEL, Vishal. A Survey on Vulnerabilities of Openflow Network and its Impacton SDN/Openflow Controller. World Academics Journal of Engineering Sciences, Department of Computer Science RCC, Gujarat University Ahmedabad, India, 1005, 5, 2014.

Zanna, P., O'Neill, B., Radcliffe, P., Hosseini, S., and Hoque, M. S. U. (2014). Adaptive threat management through the integration of IDS into software defined networks. In International Conference on the Network of the Future (NOF) - Workshop on Smart Cloud Networks & Systems, pages 1–5.