FUTURE INTERNET
BRAZILIAN ENVIRONMENT
FOR EXPERIMENTATION

Sistema de Prevenção de Intrusão baseado em SDN/OpenFlow

Visão Geral

Neste tutorial você irá desenvolver um experimento SDN usando o testbed FIBRE voltado para área de Segurança da Informação. No experimento será possível estudar, implantar e testar um sistema de detecção de intrusos integrado com um controlador OpenFlow capaz de identificar ataques cibernéticos em um ambiente com múltiplos sistemas autônomos se comunicando via BGP.

Competências desenvolvidas

1. Utilizar o testbed de experimentação FIBRE, criando máquinas virtuais no orquestrador OCF, alocando topologia virtual e definindo recursos do slice;
2. Configuração de enlaces L2 para comunicação na rede, através do padrão e-Line do Metro-Ethernet Forum;
3. Configuração de roteamento BGP via software router convencional (quagga) e via SDN (controlador Ryu);
4. Implantação de Sistema de Detecção de Intrusos (Suricata IDS);
5. Configuração de mecanismos e contenção de intrusos integrando SDN e IDS.

Nota sobre licenciamento

SUMÁRIO

1 INTRODUÇÃO

2 VISÃO GERAL SOBRE SDN/OPENFLOW E O TESTBED FIBRE
- 2.1 Redes Definidas por Software
- 2.2 Redes de testbed e o FIBRE
- 2.3 Exercícios de fixação
- 2.4 Roteiro de laboratório

3. ROTEAMENTO INTER-AS VIA BGP EM SDN/OPENFLOW
- 3.1 Roteamento Inter-AS
- 3.2 Integração entre BGP e SDN/Openflow
- 3.3 Exercícios de fixação
- 3.4 Roteiro de laboratório

4. SISTEMAS DE DETECÇÃO DE INTRUSÃO
- 4.1 Visão geral sobre IDS
- 4.2 Tipos de IDS
- 4.3. Gerenciamento de Regras do IDS
- 4.4 Exercícios de Fixação
- 4.5 Roteiro de Laboratório

5 AÇÕES DE CONTENÇÃO DE INTRUSOS COM SDN
- 5.1 Sistemas de Prevenção de Intrusos
- 5.2 Contenção de intrusos
- 5.3 Uso de SDN para execução de ações de contenção
- 5.4 Exercícios de Fixação
- 5.5 Roteiro de laboratório

6 CONCLUSÃO

7 RESUMO E LIÇÕES APRENDIDAS

AUTORES

AGRADECIMENTOS

REFERÊNCIAS

1 INTRODUÇÃO

A quantidade de ataques das mais variadas naturezas, de downloads de arquivos infectados a acessos a conteúdos maliciosos, têm crescido significativamente nas organizações nos últimos anos. Isso torna a operação de um centro de segurança cibernético cada vez mais complexa e impõe requisitos de tempo de resposta cada vez mais estritos. A automação torna-se, portanto, uma abordagem imprescindível nesse cenário, podendo ser apoiada pela visibilidade e capacidade de contenção de ataques dos Sistemas de Prevenção de Intrusos (IPS) e pela programabilidade da rede incorporada através do paradigma de Redes Definidas por Software (SDN).

Um IPS é uma tecnologia de segurança capaz de detectar e conter ameaças a partir da análise do fluxo de rede e das características das aplicações. Uma das abordagens utilizadas para detectar as ameaças consiste na verificação de assinaturas de ataques no tráfego interno e externo da organização. Uma vez que os ataques sejam detectados pelo sistema IPS, inicia-se o processo de contenção, que visa impedir a continuidade daquele incidente de segurança.

Na contenção dos intrusos pode-se utilizar estratégias de bloqueio, restrição de banda ou isolamento em quarentena, conforme sentido do tráfego, políticas da organização e funcionalidades disponíveis nos equipamentos de rede. Em particular, a utilização do paradigma SDN e de sua implantação mais conhecida, o protocolo Openflow, podem aumentar a flexibilidade na integração do IPS com os elementos de rede da organização, potencializando a adoção de diferentes ações de contramedida.

Não obstante, considerando a criticidade e os acordos de nível de serviço das redes atuais, soluções desse tipo precisam ser amplamente validadas em laboratórios ou em ambientes de experimentação antes de serem implantadas em ambiente de produção. Uma das abordagens para experimentações como essa são os *testbeds*, infraestruturas físicas disponíveis para investigação de protocolos, tecnologias e algoritmos. No Brasil, a Rede Nacional de Ensino e Pesquisa (RNP) mantém o FIBRE (do inglês, *Future Internet Brazillian Environment for Experimentation*), que funciona como um laboratório virtual de larga escala para estudantes e pesquisadores testarem novas aplicações e modelos de arquitetura de rede.

Desta forma, o objetivo desta oficina é explorar o ambiente do *testebed* FIBRE na construção de uma estrutura com múltiplos sistemas autônomos com capacidade de detecção de intrusos, executando ações de prevenção com bloqueio de ataques externos e com isolamento de *hosts* internos. Para isso, os alunos farão uso da infraestrutura distribuída do FIBRE para construir e executar as funcionalidades do cenário proposto. Cada grupo representará um Sistema Autônomo (AS), tendo à sua disposição: i) switches Openflow para fazer o encaminhamento de pacotes e aplicação de políticas de segurança conforme definido pelo Controlador SDN; ii) máquina virtual para executar a orquestração da rede através do controlador Openflow Ryu e da aplicação SDN-IPS (contribuição desta oficina); iii) máquina virtual para detecção de intrusos; iv) uma máquina virtual para representar os clientes do AS; v) máquina virtual representando os serviços de produção do AS (ex: páginas web).

Ao longo deste material o leitor encontrará texto de apoio e roteiros de laboratório para viabilizar o cenário de experimentação proposto acima.

2 VISÃO GERAL SOBRE SDN/OPENFLOW E O *TESTBED* FIBRE

Esta seção apresenta uma revisão dos conceitos de SDN/Openflow bem como uma visão geral sobre o *testbed* FIBRE.

2.1 Redes Definidas por Software

A infraestrutura de rede atual, que, no geral, consiste de roteadores, comutadores etc., tem sua lógica de funcionamento definida pelos fabricantes de equipamentos, com base em padrões e protocolos especificados por entidades como IETF e IEEE, ou ainda em padrões proprietários, especificados pelos próprios fabricantes. Nesse modelo, novos padrões de rede podem demorar muito tempo até serem amplamente suportados e adotados. A transição entre IPv4 e IPv6, por exemplo, já se prolonga por mais de dez anos e ainda com baixa adesão. Esse processo de padronização e a adoção pelos fabricantes, em conjunto com a dificuldade de inovação das redes IP atuais, podem acarretar um longo tempo para um novo protocolo de roteamento ser totalmente projetado, avaliado e adotado (Kreutz et al. , 2015). Esse modelo de plataforma dificulta o ajuste ou alteração do comportamento da rede, sendo obstáculo para inovação e para customizações avançadas na lógica de funcionamento dos equipamentos. Com objetivo de alcançar uma plataforma aberta e um padrão para desenvolvimento da rede, foi proposto o paradigma Redes Definidas por Software (SDN, do inglês Software-Defined Networking) (ONF , 2012b; Mckeown , 2009).

O paradigma SDN propõe a separação entre o plano de encaminhamento e o plano de controle da rede, permitindo que aplicações e serviços de rede possam ser diretamente programados em um controlador remoto (Kreutz et al. , 2015; Mckeown , 2009). Esse controlador remoto define o comportamento da rede através de aplicações programadas pelo administrador, que atuam no plano de controle da rede. O controlador, dessa forma, funciona como um sistema operacional da rede (Kreutz et al. , 2015), provendo abstrações de suas funções e serviços. Por exemplo, protocolos de roteamento que eram executados nos equipamentos de rede como BGP e OSPF agora são executados no controlador. O plano de encaminhamento permanece nos equipamentos de rede, chamados de comutadores SDN, que simplesmente aplicam as regras de encaminhamento, definidas pelo controlador remoto. Como exemplo, podemos

analisar a função de espelhamento de tráfego, cuja definição das regras ocorre no controlador, porém a sua aplicação continua sendo no switch ou roteador (comutador SDN).

Uma das propostas mais adotadas para o paradigma SDN é o protocolo *Openflow* (Mckeown et al., 2008; ONF , 2012a), que define uma API de comunicação entre o controlador e o comutador SDN (Southbound API). A arquitetura *Openflow* é composta por: i) comutadores *Openflow*, que são responsáveis pelo encaminhamento dos pacotes com base em regras predefinidas; e ii) o controlador, que gerencia os comutadores *Openflow*, definindo suas regras de funcionamento. Conforme ilustrado na Figura 2.1, cada comutador *Openflow* contém uma ou mais tabelas de fluxos (*flow tables*), que são usadas no casamento e encaminhamento de pacotes, e um canal seguro, que conecta o comutador ao controlador. Cada tabela de fluxos contém entradas que são usadas na comparação e encaminhamento de pacotes. Cada entrada da tabela de fluxos consiste de campos de casamento (*rules*), um conjunto de instruções a serem aplicadas em pacotes correspondentes (*actions*) e contadores (*stats*). Fica a cargo do controlador adicionar, remover ou modificar entradas na tabela de fluxos, bem como consultar estatísticas (ONF, 2012a).

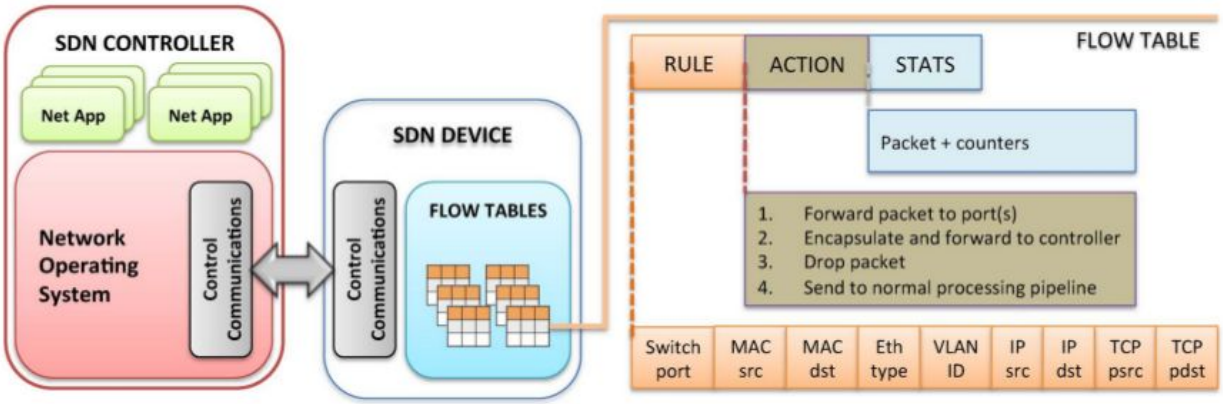


Figura 2.1. Comutador SDN com Openflow habilitado (Kreutz et al., 2015).

Ademais, deve-se configurar no comutador *Openflow* o conjunto de portas nas quais o controle SDN será aplicado (também conhecido como *datapath*). Um comutador *Openflow* pode operar com todas as portas pertencendo ao *datapath Openflow*, com algumas portas operando com protocolos legados e outras operando com *Openflow*, ou ainda com portas híbridas cujo

funcionamento, ora é definido pelo controlador, ora funciona com protocolos legados[1]. As portas que fazem parte do *datapath Openflow* são chamadas de portas SDN, incluindo as portas híbridas. Cada comutador Openflow é identificado por um *Datapath ID*, cujo formato se assemelha à um endereço MAC (ex: 00:00:0c:c4:7a:5e:98:95).

Quando um comutador *Openflow* recebe um pacote em uma porta SDN, ele consulta a tabela de fluxos para decidir o que fazer com o pacote (exemplo: encaminhá-lo para uma porta, descartá-lo, encaminhá-lo por inundação etc.). Caso não haja correspondência previamente estabelecida, o pacote (ou apenas seus cabeçalhos) é enviado para o controlador. O controlador, então, processa o pacote, por meio de aplicações e algoritmos, e "ensina" o comutador *Openflow* a tratar os próximos pacotes daquele fluxo, inserindo uma nova entrada na tabela de fluxos do comutador. Além disso, é possível definir regras iniciais no comutador *Openflow*, a fim de que ele possa encaminhar determinado tráfego sem depender do controlador.

Existem diferentes abordagens para trabalhar ou experimentar os conceitos de SDN/Openflow, variando desde ambientes emulados até equipamentos reais. Em termos de emulação, uma solução que tem sido amplamente utilizada é o emulador Mininet[2]. O Mininet faz uso de switches *Openflow* baseados em software (tipicamente o Open vSwitch[3]) e permite criar diferentes topologias de rede para testar aplicações e controladores. Além da emulação, outra abordagem que pode ser utilizada é a aplicação de SDN/Openflow em equipamentos reais que já estão disponíveis no mercado, como Brocade, Pica8, Mellanox, entre outros. Esses equipamentos podem ser usados tanto para testes de laboratório isolados quanto em redes de produção, como é o caso do projeto Amlight[4].

Uma abordagem intermediária, que visa investigar novas soluções em equipamentos reais sem impactar nas redes de produção, é a utilização de *testbeds* - infraestruturas especialmente

[1] Poucos fabricantes suportam esse modo de operação, implementando-o, geralmente, através de separação de VLANs, onde algumas VLANs pré-definidas operam em modo SDN e todas as demais seguem o fluxo de encaminhamento padrão do equipamento.
[2] http://mininet.org/
[3] http://openvswitch.org/
[4] https://www.amlight.net/

criadas para experimentação. O conceito de *testbed* e as principais características do FIBRE serão vistos na próxima seção.

2.2 Redes de testbed e o FIBRE

Os *testbeds* permitem a implementação e avaliação de diferentes soluções (tecnologias, aplicações, protocolos, controladores) em um ambiente físico próximo ao ambiente de produção, com sensores, sistemas virtuais e diversos outros recursos. Dessa forma, é possível validar experimentos do ponto de vista de escalabilidade e heterogeneidade.

No Brasil, a RNP mantém o projeto FIBRE[5], um *testbed* federado que conecta infraestruturas de experimentação localizadas no território nacional. Ele é composto por diversas ilhas de experimentação que estão distribuídas em instituições de ensino e pesquisa parceiras do projeto. Em cada uma dessas ilhas há recursos que englobam diferentes tecnologias, como *Openflow*, redes sem fio e redes ópticas. Todos esses mecanismos são gerenciados através de Frameworks de Controle e Monitoramento (do inglês, *Control and Monitoring Frameworks* - CMF). Com o uso de CMFs é possível fazer um gerenciamento centralizado e permitir a execução de experimentos que utilizem recursos localizados em diferentes ilhas, além de implantar mecanismos de medição e monitoramento de uso dos recursos. Atualmente, o CMF utilizado pelo FIBRE é o OCF (*Ofelia Control Framework*), disponível através do portal https://ocf.fibre.org.br.

Na Figura 2.2 é possível observar a distribuição das ilhas do FIBRE no território brasileiro conectadas através da FIBRENet[6]. A FIBRENet é uma rede que se sobrepõe à Rede Ipê (*backbone* da RNP). Para experimentação em SDN/*Openflow*, o usuário do FIBRE tem à sua disposição um conjunto de máquinas virtuais (Virtual Machines - VM) e switches *Openflow* que podem ser utilizados na composição de diversas topologias. O projeto conta ainda com outros recursos para experimentação, como recursos de rede sem fio, porém, tais recursos estão fora do escopo desta oficina. Em relação ao funcionamento e características da estrutura, as VMs são utilizadas para instanciar os recursos requisitados pelo experimentador e os switches

[5] http://fibre.org.br/
[6] http://fibre.org.br/infrastructure/fibrenOpenflowet/

Openflow são baseados em NetFPGA ou equipamentos Pronto/PICA8, permitindo realização de experimentos com *Openflow* 1.0.

Figura 2.2. Representação da FIBRENet.

Além disso, o FIBRE utiliza o *FlowVisor* para realizar técnicas de multiplexação com o objetivo de separar o plano de controle de diferentes experimentadores que estejam executando testes simultaneamente. Assim, o *FlowVisor* é responsável por isolar cada rede de experimento através da criação de fatias ou divisões dos recursos de rede, incluindo: topologia (portas, enlaces, faixa de vlans, etc), tabela de fluxos, limite de fluxos por segundo, entre outras. Dessa maneira, o *FlowVisor* atua como um proxy entre os comutadores *Openflow* e os controladores, interceptando as mensagens *Openflow* para decidir o que fazer de acordo com a política definida para a fatia em questão.

Neste curso utilizaremos a infraestrutura do FIBRE para instanciar e executar os experimentos ora propostos. Na seção seguinte serão apresentados todos os procedimentos necessários para fazer uso do *testbed* FIBRE e executar as aplicações SDN aqui desenvolvidas.

2.3 Exercícios de fixação

1. Como SDN/OpenFlow trata as seguintes questões:
    a. Múltiplas sintaxes de CLI;
    b. Funcionalidades dependentes de fabricante – tempo de implantação;
    c. Licenciamento por funcionalidade;
    d. Impossibilidade de testar novas funcionalidades de rede (protocolos);
    e. Customizações restritas aos parâmetros de configuração;
    f. Rede com pouca flexibilidade.
2. Como executar experimentos utilizando o paradigma SDN?
3. O que é um *testbed*? Como funciona o FIBRE?
4. Quem pode ter acesso ao FIBRE e como?

2.4 Roteiro de laboratório

Nesta seção vamos apresentar os passos necessários para obter acesso ao *testbed* FIBRE e realizar um experimento simples de setup de uma rede L2 baseada no padrão “e-Line” do Metro-ethernet Forum[7], cujo objetivo é prover uma conexão ponto-a-ponto entre duas UNIs (do inglês, *User Network Interface*).

Os comandos mostrados abaixo pressupõem que o leitor está trabalhando em uma máquina Linux Debian, porém são facilmente adaptáveis para outras distribuições. A topologia proposta para esse e os próximos experimentos pode ser observada na Figura 2.3.

[7] https://mef.net/Assets/White_Papers/Metro-Ethernet-Services.pdf

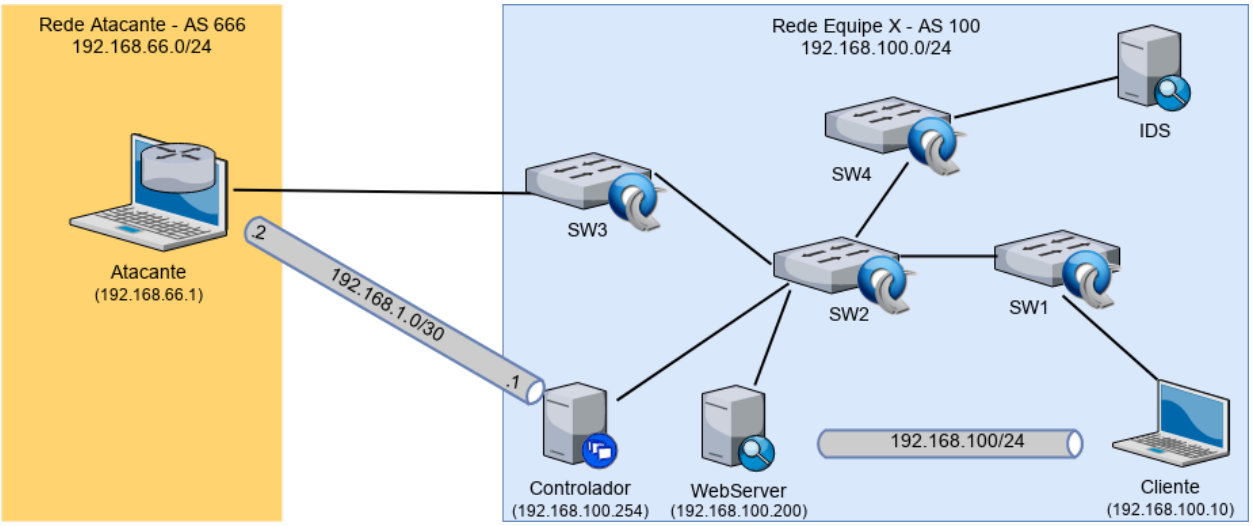


Figura 2.3. Topologia proposta para os experimentos da Oficina.

2.4.1 Criar conta no projeto FIBRE

O primeiro procedimento necessário para acessar o *testbed* é a criação de uma conta no projeto FIBRE. Para isso, basta acessar o portal do FIBRE, escolher a ilha à qual deseja associar sua conta e adicionar suas informações (lembre-se de usar um e-mail institucional). Caso sua instituição não esteja listada, você deverá criar a conta na RNP (instituição acadêmica) ou no CPqD (instituição do mercado).

Uma vez que sua conta seja aprovada, você receberá um e-mail de confirmação indicando que já é possível utilizar o *testbed*. Nesse mesmo e-mail, serão dadas duas opções de acesso: através do portal local da ilha e através do portal central do *testbed*. Para a oficina, é recomendada a escolha do portal central do *testbed*.

O acesso ao *testbed* é realizado através da VPN do FIBRE. Para isso, faz-se necessário o download e instalação do OpenVPN e a configuração da VPN FIBRE. Todas as informações de instalação e acesso, utilizando uma máquina cliente baseada em Debian 9, são listadas abaixo:

1) Instalar o OpenVPN:

```
sudo apt-get install openvpn
```

2) Realizar o *download* do arquivo: fibre-vpn.zip

```
wget --no-check-certificate
https://wiki.rnp.br/download/attachments/87101626/fibre-vpn.zip
```

3) Descompactar o arquivo no diretório /etc/openvpn/client:

```
sudo unzip fibre-vpn.zip -d /etc/openvpn/
```

4) Acessar a VPN do FIBRE:

```
cd /etc/openvpn/fibre-vpn && sudo openvpn fibre-vpn.ovpn
```

Ao conectar-se com a VPN FIBRE, serão solicitados seu login e senha. Se tudo ocorrer bem, você poderá visualizar uma mensagem de confirmação de inicialização, conforme Figura 2.4.

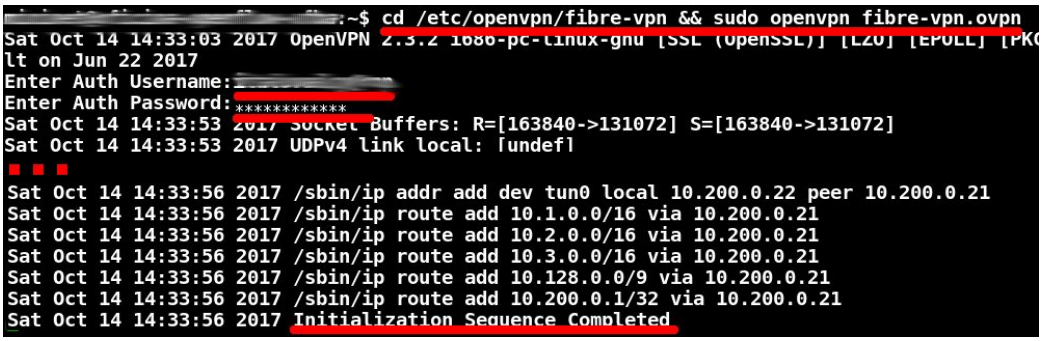


Figura 2.4. Inicialização da VPN do Fibre.

2.4.2 Criar um slice no portal OCF

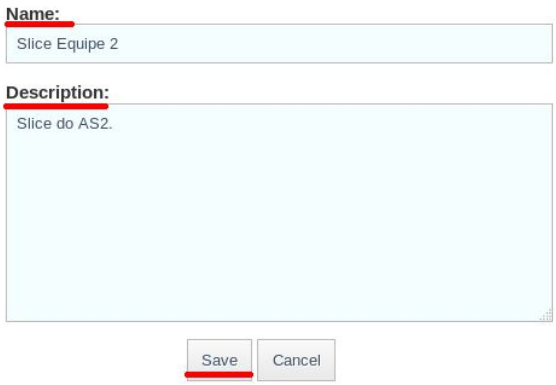
O próximo passo para criar o ambiente de experimentação é acessar o portal OCF (*framework* de controle e monitoramento). Nesse portal, o usuário deverá entrar no projeto “Projeto IPS com SDN/OpenFlow” (caso o projeto não exista, você poderá solicitar a criação de um novo) e criar um *slice.* Nas Figuras 2.5, 2.6, 2.7, 2.8 e 2.9 é demonstrado o procedimento para criação de um *slice*, sendo que o primeiro passo é acessar o projeto criado no FIBRE.

Figura 2.5. Acesso ao projeto no qual será criado o bloco.

Para criar um *slice*, basta clicar na opção “*Create Slice*”, conforme Figura 2.6:

Figura 2.6. Criação do slice

Em seguida, deve-se definir o nome do *slice* e uma descrição, conforme ilustrado na Figura 2.7.



2.7. Definição do nome e descrição do slice.

Em seguida, é preciso adicionar os recursos que serão utilizados. Desta forma, é preciso clicar em “*Add an Aggregate Manager to the current slice*”, como pode ser observado na Figura 2.8, e selecionar os recursos que serão utilizados (Figura 2.9).

Figura 2.8. Adição de recursos ao *slice.*

Add an Aggregate for Slice Slice Equipe 2

Name	Type	Location	Description	Size	Managers	Status	Actions
UFBA Openflow	OpenFlow Aggregate	Salvador - BA	TBD	24	fibre	✔	Select
UFBA Virtualization	Virtualization Aggregate	Salvador - BA	Virtualization Aggregate	44	fibre	✔	Select

Done

Figura 2.9. Seleção dos recursos que devem ser adicionados no slice.

Uma vez alocados os recursos do projeto que serão utilizados, é preciso instanciá-los através da criação das VMs.

2.4.3 Criar as máquinas virtuais

O processo de criação de uma máquina virtual é demonstrado nas figuras abaixo. Nesse processo é preciso escolher o servidor que será utilizado na criação da VM e suas configurações, como nome, capacidade de memória, entre outros, ambos processos podem ser observados nas Figuras 2.10 e 2.11, respectivamente. Essas configurações irão depender, diretamente, do cenário escolhido e das características e funcionalidades que os dispositivos irão atuar.

Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
UFBA_Whitebox1	XEN	GNU/Linux Debian (7.0)	None	None	None	Update
UFBA_Whitebox2	XEN	GNU/Linux Debian (7.0)	None	None	None	Update
UFBA_Whitebox3	XEN	GNU/Linux Debian (7.0)	None	None	None	Update
UFBA_Whitebox4	XEN	GNU/Linux Debian (7.0)	None	None	None	Update

Actions: Create a new virtual machine in server: UFBA_Whitebox1 Create VM

Figura 2.10. Criação de uma máquina virtual.

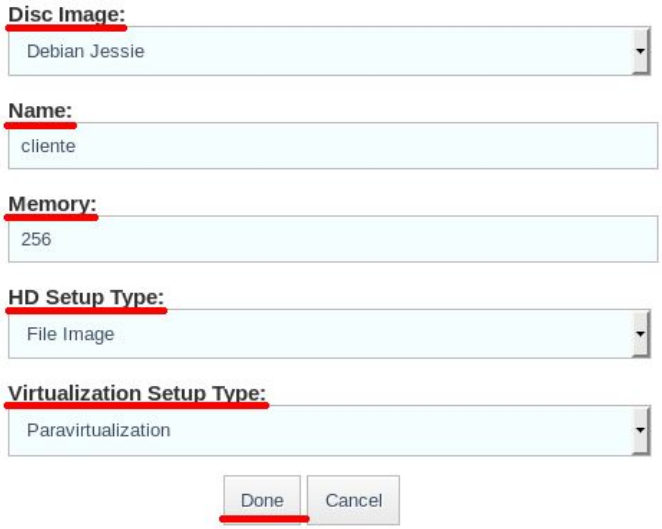


Figura 2.11. Configuração da máquina virtual.

Para o cenário da oficina, deve-se criar as máquinas virtuais de acordo com as configurações apresentadas no Quadro 2.1.

Quadro 2.1. Configurações das máquinas virtuais

Nome	**Whitebox**	**Imagem**	**Função**	**Memória**
Cliente	Whitebox1	Debian Jessie	Cliente	256MB
Controlador	Whitebox2	Debian Jessie	Controlador	1024MB
WebServer	Whitebox2	Debian Jessie	Servidor de páginas Web	256MB
Atacante	Whitebox3	Debian Jessie	Atacante	512MB
IDS	Whitebox4	Debian Jessie	Sistema de Detecção de Intrusão	512MB

Ao final desta etapa, deve ser possível observar as características das VMs criadas (nome, status, sistema operacional, memória e **endereço IP**) e são disponibilizados botões que servem para iniciar/parar, deletar e atualizar a VM, conforme Figura 2.12.

Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
UFBA_Whitebox1	XEN	GNU/Linux Debian (7.0)	None	None	None	Update

VM Name	State	Operating System	Memory	Mgmt IP	Actions	Update Status
Cliente	created (stopped)	GNU/Linux Debian (8.0)	256	10.144.12.47	Start Delete	Update

SSH access: ~# ssh @rnp@10.144.12.47 (password: your user password)

*SSH common details: to access as root just type **su** inside (password: **openflow**)*

Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
UFBA_Whitebox2	XEN	GNU/Linux Debian (7.0)	None	None	None	Update

VM Name	State	Operating System	Memory	Mgmt IP	Actions	Update Status
Controlador	created (stopped)	GNU/Linux Debian (8.0)	1024	10.144.12.48	Start Delete	Update
WebServer	created (stopped)	GNU/Linux Debian (8.0)	256	10.144.12.44	Start Delete	Update

SSH access: ~# ssh @rnp@10.144.12.48 (password: your user password)

SSH access: ~# ssh @rnp@10.144.12.44 (password: your user password)

*SSH common details: to access as root just type **su** inside (password: **openflow**)*

Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
UFBA_Whitebox3	XEN	GNU/Linux Debian (7.0)	None	None	None	Update

VM Name	State	Operating System	Memory	Mgmt IP	Actions	Update Status
Atacante	created (stopped)	GNU/Linux Debian (8.0)	512	10.144.12.49	Start Delete	Update

SSH access: ~# ssh @rnp@10.144.12.49 (password: your user password)

*SSH common details: to access as root just type **su** inside (password: **openflow**)*

Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc	Update
UFBA_Whitebox4	XEN	GNU/Linux Debian (7.0)	None	None	None	Update

VM Name	State	Operating System	Memory	Mgmt IP	Actions	Update Status
IDS	created (stopped)	GNU/Linux Debian (8.0)	512	10.144.12.56	Start Delete	Update

SSH access: ~# ssh @rnp@10.144.12.56 (password: your user password)

*SSH common details: to access as root just type **su** inside (password: **openflow**)*

Figura 2.12. Representação das máquinas virtuais configuradas de acordo com a topologia que será utilizada na oficina.

Depois de alocar os recursos e criar as VMs, é preciso instanciar a topologia do experimento.

2.4.4 Instanciar a topologia do experimento

Para instanciar a topologia, é necessário fazer configurações em relação a conectividade dos nós. Para isso, é preciso clicar em "*Define flowspace (virtual topology)*", conforme Figura 2.13.

Slice AMs and resource details

+ Add an Agregate Manager to the current slice

Network resources

- **OpenFlow Aggregate: UFBA Openflow**

Name: **UFBA Openflow**
Status:
Automatic approval:
Physical location: **Salvador - BA**
Resources:

Requested Flowspace (1) More information

Flowspace has not been granted yet, which means slice controller will NOT receive any packet. Consider contacting Island Manager(s) involved if grant of the flowspace does not happen in a reasonable time-frame.

Openflow controller: **Not set** Set controller

Actions: Define flowspace (virtual topology) Your Flowspace can only be granted if your slice is started

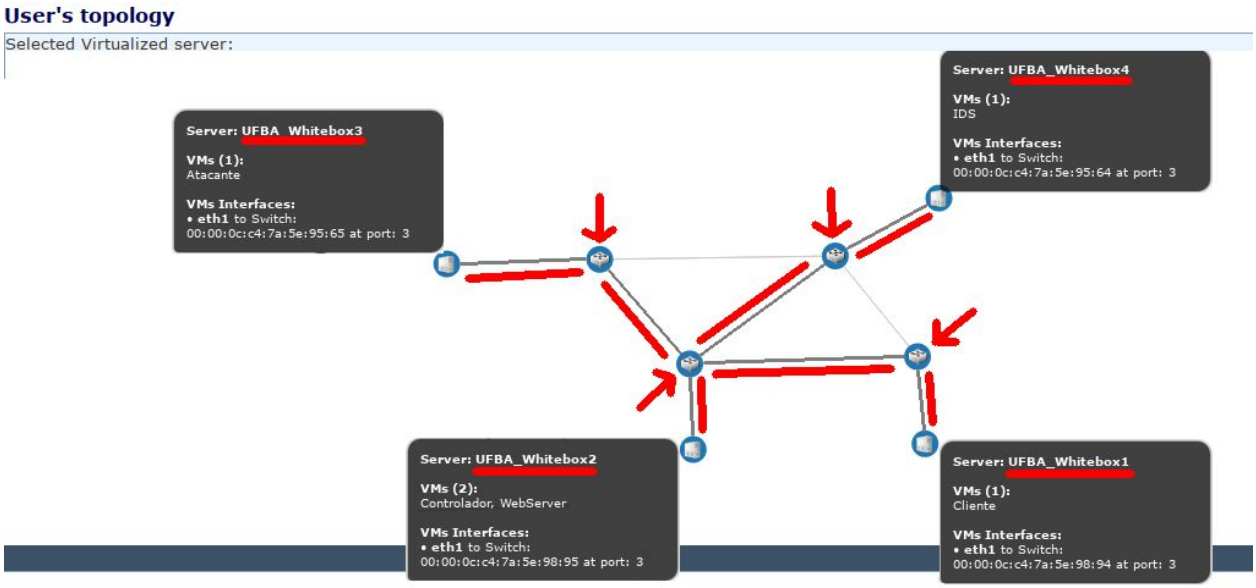
Remove from slice: Remove AM

2.16. Definiçao do *Flowspace*.

Em seguida, na aba "*Select Openflow Resources*" selecione a opção *Simple* e selecione, no mínimo, duas VLANs (Figura 2.17).

2.17. Seleção do número de VLANs.

Uma vez estabelecidas as VLANs, é preciso marcar as portas de conexão de cada equipamento, de forma que a topologia reflita a arquitetura proposta na Figura 2.3. Em particular, selecione apenas os switches que estão ligados ao Whitebox1, Whitebox2, Whitebox3, e Whitebox4 conforme ilustrado na Figura 2.18 (observe que os links e switches selecionados ficam com cor cinza escuro):



2.18. Topologia instanciada de acordo com a arquitetura proposta.

Uma vez selecionadas todas as portas, basta clicar em *Next* (rodapé da página) e aceitar as configurações selecionadas, ainda que apareça uma mensagem de "Existem *loops* na topologia selecionada". Feito isso você terá finalizado a configuração da topologia de rede e retornará à página inicial do OCF. Na página principal, na seção "Network resources", é possível observar os recursos de rede que foram alocados, conforme Figura 2.19.

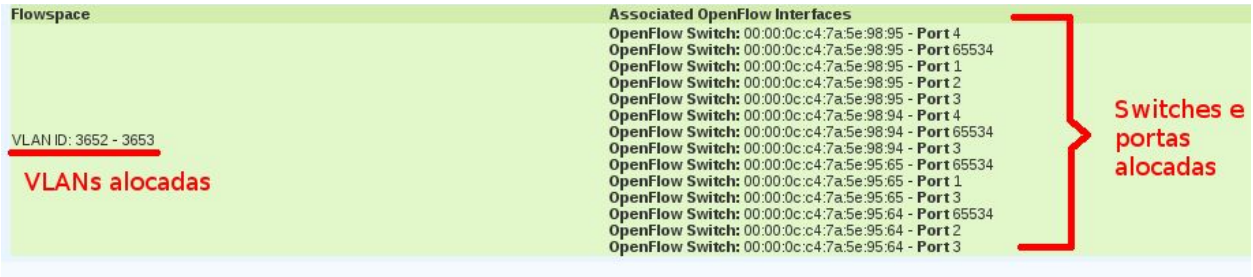
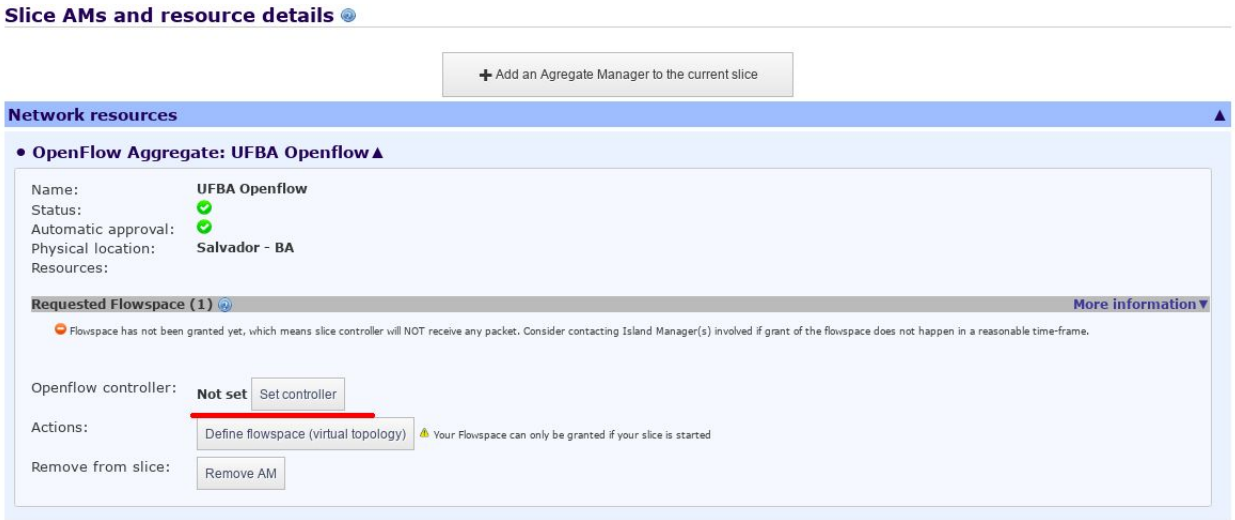


Figura 2.19. Visualização dos recursos de rede alocados.

Depois de configurar a topologia, é preciso definir o IP do controlador. De acordo com a topologia definida no início desta seção, o controlador será a máquina criada no Whitebox2. Assim você deve clicar em "Set Controller" e depois escolher a máquina Controlador, conforme ilustrado nas Figura 2.20 e 2.21.



2.20. Definição do controlador.

2.21. Escolha do IP e porta do controlador.

Finalizadas todas as configurações, é possível iniciar o experimento e ligar as máquinas virtuais. Para isso, basta clicar em "*Start slice*" e em seguida, clicar em "*Start*" para cada máquina virtual, conforme Figuras 2.22 e 2.23.

Figura 2.22. Inicialização de um *slice*.

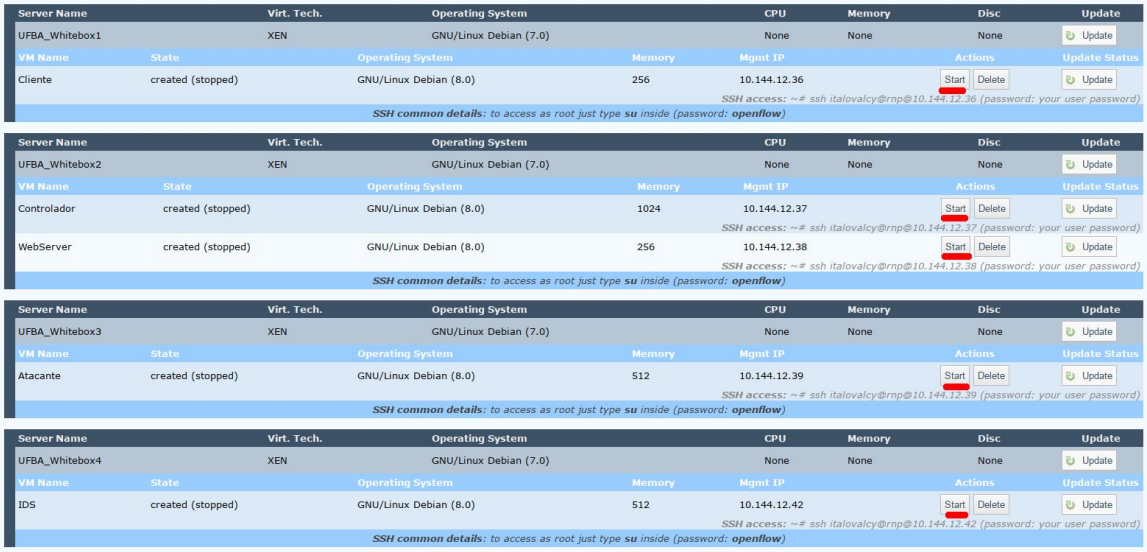


Figura 2.23. Inicialização das máquinas virtuais.

2.4.4 Instalação do controlador Ryu

Além da configuração da topologia, é preciso fazer a instalação do software de controle *Openflow* na máquina Controlador. O software utilizado na oficina é o Ryu (versão 4.15), e para instalá-lo, basta seguir os passos listados abaixo.

1) Na máquina Controlador, autenticar como usuário administrador (root) e fornecer a senha (verifique a senha na configuração da VM):

```
su
```

2) Instalar algumas dependências via pacotes Debian (apenas uma linha):

```
apt-get install -y tcpdump git curl libxslt1-dev libffi-dev python-msgpack python-setuptools
python-nose python-pip python-dev libssl-dev python-jsonpickle
```

3) Atualizar algumas bibliotecas do Python via pip:

```
pip install --upgrade cffi
```

4) Instalar dependências Python via pip (apenas uma linha):

```
pip install ipaddr bitarray netaddr 'cryptography==1.9' oslo.config routes webob paramiko
mock xml_compare pyflakes pylint 'debtcollector==1.16.0' oslo.i18n rfc3986 greenlet tinyrpc
ovs 'networkx==1.11' 'eventlet>=0.18.2,!=0.18.3,!=0.20.1,<0.21.0' 'six>=1.9.0'
```

5) Obter o código do Ryu:

```
git clone https://github.com/osrg/ryu.git
```

6) Entrar no diretório do ryu e executar:

```
cd ryu
git checkout v4.15
python setup.py install
```

2.4.5 Execução da aplicação SDN-IPS

Depois de instalar o controlador Ryu, vamos executar a aplicação SDN-IPS e estabelecer um serviço de conexão e-Line para nosso experimento. É preciso configurar um e-Line entre o controlador e o atacante e o controlador e o cliente. Essas etapas são listadas logo abaixo.

Antes de prosseguir, você deve mapear os seguintes parâmetros:

- **DPID-SW1**: de acordo com a topologia criada, qual o identificador do switch SW1? De acordo com os passos anteriores esse é o switch conectado ao Whitebox1 (no exemplo foi 00:00:0c:c4:7a:5e:98:94 - atente-se para identificar o seu). O mesmo se aplica para SW2 e assim por diante.
- **VLAN_A**: essa é a VLAN que será usada na conexão entre o Cliente e o Controlador. Utilize a VLAN de <u>menor</u> número (no exemplo acima [passo 4 da prática 2.4.3] foi VLAN 3652 - atente-se para identificar a sua)
- **VLAN_B**: essa é a VLAN que será usada na conexão entre o Controlador e o Atacante. Utilize a VLAN de <u>maior</u> número (no exemplo acima [passo 4 da prática 2.4.3] foi VLAN 3653 - atente-se para identificar a sua)

1) Acessar a máquina virtual do Controlador e fazer o clone da aplicação

```
cd /root
git clone https://nuvem.pop-ba.rnp.br/gitlab/ufba/fibre-2a-opencall-sdn-ips.git
```

2) Iniciar a aplicação

```
cd /root/fibre-2a-opencall-sdn-ips
ryu-manager --observe-links fibre-2a-opencall-sdn-ips.py
```

Ao iniciar a aplicação, observa-se a conexão dos switches e a descoberta de links:

```
root@Controlador:~/fibre-2a-opencall-sdn-ips# ryu-manager --observe-links fibre-2a-opencall-sdn-ids.py
loading app fibre-2a-opencall-sdn-ids.py
Generating grammar tables from /usr/lib/python2.7/lib2to3/Grammar.txt
Generating grammar tables from /usr/lib/python2.7/lib2to3/PatternGrammar.txt
loading app ryu.controller.dpset
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
creating context wsgi
instantiating app ryu.topology.switches of Switches
instantiating app fibre-2a-opencall-sdn-ids.py of SDNIPSApp
Fail to load SDN-IPS config. Error: [Errno 2] No such file or directory: 'sdn-ips-config.json'
instantiating app ryu.controller.dpset of DPSet
instantiating app ryu.controller.ofp_handler of OFPHandler
(472) wsgi starting up on http://0.0.0.0:8080
OFPStateChange switch entered: dpid=00000cc47a5e9894
OFPStateChange switch entered: dpid=00000cc47a5e9564
OFPStateChange switch entered: dpid=00000cc47a5e9565
OFPStateChange switch entered: dpid=00000cc47a5e9895
Update graph edges:
==> 00000cc47a5e9894:4 <-> 00000cc47a5e9895:4
==> 00000cc47a5e9895:4 <-> 00000cc47a5e9894:4
Update graph edges:
==> 00000cc47a5e9895:2 <-> 00000cc47a5e9564:2
==> 00000cc47a5e9564:2 <-> 00000cc47a5e9895:2
==> 00000cc47a5e9565:1 <-> 00000cc47a5e9895:1
==> 00000cc47a5e9895:1 <-> 00000cc47a5e9565:1
```

Conexão dos switches

Descoberta de links

3) Abrir um segundo *console* da máquina virtual do Controlador para listar os switches, via API rest da aplicação:

```
curl -s http://localhost:8080/sdnips/switches | python -m json.tool
```

A saída desse comando pode ser observada logo abaixo:

```
root@Controlador:~# curl -s http://localhost:8080/sdnips/switches | python -m json.tool
[
    "00000cc47a5e9564",
    "00000cc47a5e9894",
    "00000cc47a5e9895",
    "00000cc47a5e9565"
]
```

4) Em seguida, é preciso listar as portas de cada switch. Para isso, basta substituir <ID-SWITCH-n> pelo id de cada switch listado no passo anterior.

```
curl -s http://localhost:8080/sdnips/switches/<ID-SWITCH-1>/ports | python -m json.tool
curl -s http://localhost:8080/sdnips/switches/<ID-SWITCH-2>/ports | python -m json.tool
curl -s http://localhost:8080/sdnips/switches/<ID-SWITCH-3>/ports | python -m json.tool
curl -s http://localhost:8080/sdnips/switches/<ID-SWITCH-4>/ports | python -m json.tool
```

```
root@Controlador:~# curl -s http://localhost:8080/sdnips/switches/00000cc47a5e9564/ports | python -m json.tool
[
    3
]
root@Controlador:~# curl -s http://localhost:8080/sdnips/switches/00000cc47a5e9894/ports | python -m json.tool
[
    2,
    3
]
root@Controlador:~# curl -s http://localhost:8080/sdnips/switches/00000cc47a5e9895/ports | python -m json.tool
[
    3
]
root@Controlador:~# curl -s http://localhost:8080/sdnips/switches/00000cc47a5e9565/ports | python -m json.tool
[
    2,
    3
]
```

5) O próximo passo é criar um e-Line entre a UNI do Controlador e a UNI da máquina Cliente via VLAN_A (substituir as coisas entre <>):

```
curl -s -X POST -d '{"uniA" : "<ID-SWITCH-Controlador>:<PORTA-ACESSO>", "uniB" : "<ID-SWITCH-Cliente>:<Porta-Acesso>", "vlanid": <VLAN_A>}' http://localhost:8080/sdnips/e-line/create | python -m json.tool
```

Caso o e-Line seja configurado com sucesso, será exibida a seguinte mensagem:

```
root@Controlador:~# curl -s -X POST -d '{"uniA" : "00000cc47a5e9895:3", "uniB" :
 "00000cc47a5e9894:3", "vlanid": 1688}' http://localhost:8080/sdnips/e-line/crea
te | python -m json.tool
[
    "Success"
]
```

Ao olhar no *console* que está executando a aplicação, é possível observar a criação do e-Line através do algoritmo do controlador.

```
(498) accepted ('127.0.0.1', 56078)
==> create_eline(UNI-A=14038006143125:3, UNI-B=14038006143124:3, vlanid=1688): [
14038006143125, 14038006143124]
==> add_flow sw=14038006143125 (->) in_port=3 vlanid=1688 action_out_port=4
==> add_flow sw=14038006143125 (<-) in_port=4 vlanid=1688 action_out_port=3
==> add_flow sw=14038006143124 (->) in_port=4 vlanid=1688 action_out_port=3
==> add_flow sw=14038006143124 (<-) in_port=3 vlanid=1688 action_out_port=4
127.0.0.1 - - [01/Dec/2017 09:51:37] "POST /sdnips/e-line/create HTTP/1.1" 200 1
19 0.011249
```

6) Criar um e-Line entre a UNI do Controlador e da máquina Atacante via VLAN B (substituir as coisas entre <>):

```
curl -s -X POST -d '{"uniA" : "<ID-SWITCH-Controlador>:<PORTA-ACESSO>", "uniB" :
"<ID-SWITCH-Atacante>:<Porta-Acesso>", "vlanid": <VLAN_B>}'
http://localhost:8080/sdnips/e-line/create | python -m json.tool
```

2.4.6 Configuração e teste de conectividade nas máquinas

Depois da configuração dos e-Lines é preciso fazer as configurações de VLAN na máquina Cliente, no Controlador, no WebServer e na máquina Atacante.

1) Configurar a máquina Cliente na VLAN_A, IP 192.168.100.10 e ajustar as rotas da rede interna. Para isso, é necessário adicionar ao arquivo */etc/network/interfaces* (atente-se para alterar a VLAN) as seguintes informações:

```
auto eth1
iface eth1 inet manual
        pre-up ifconfig $IFACE up
        post-down ifconfig $IFACE down

auto vlan3652
iface vlan3652 inet static
        vlan-raw-device eth1
        address 192.168.100.10
        netmask 255.255.255.0
        post-up route add -net 192.168.0.0/16 gw 192.168.100.254
        pre-down route del -net 192.168.0.0/16 gw 192.168.100.254
```

2) Uma vez configurada a VLAN, é necessário ativar os serviços. Para isso, digite no terminal:

```
ifup eth1
ifup vlan3652
```

3) Na máquina Atacante, configurar a vlan VLAN_B e IP 192.168.1.2. Para isso, é necessário editar o arquivo */etc/network/interfaces* (atente-se para alterar a VLAN). Nesse caso também é preciso adicionar um endereço à interface de loopback:

```
auto lo:0
iface lo:0 inet static
        address 192.168.66.1
        netmask 255.255.255.0
        post-up iptables -t nat -A POSTROUTING -d 192.168.100.0/24 -j SNAT --to
192.168.66.1

auto eth1
iface eth1 inet manual
        pre-up ifconfig $IFACE up
        post-down ifconfig $IFACE down

auto vlan3653
iface vlan3653 inet static
        vlan-raw-device eth1
        address 192.168.1.2
        netmask 255.255.255.252
```

4) Para ativar os serviços, digite no terminal:

```
ifup eth1
ifup vlan3653
ifup lo:0
```

5) Na máquina Controlador, configurar a VLAN_A com IP 192.168.100.254 e a VLAN B com IP 192.168.1.1. Para isso, é necessário editar arquivo */etc/network/interfaces* (atente-se para alterar a VLAN):

```
auto eth1
iface eth1 inet manual
        pre-up ifconfig $IFACE up
        post-down ifconfig $IFACE down
```

```
auto vlan3652
iface vlan3652 inet static
	vlan-raw-device eth1
	address 192.168.100.254
	netmask 255.255.255.0

auto vlan3653
iface vlan3653 inet static
	vlan-raw-device eth1
	address 192.168.1.1
	netmask 255.255.255.252
```

6) Para ativar os serviços, digite no terminal:

```
ifup eth1
ifup vlan3652
ifup vlan3653
```

7) Na máquina WebServer, configurar a VLAN_A e IP 192.168.100.200. Para isso, é necessário editar arquivo */etc/network/interfaces* (atente-se para alterar a VLAN):

```
auto eth1
iface eth1 inet manual
	pre-up ifconfig $IFACE up
	post-down ifconfig $IFACE down

auto vlan3652
iface vlan3652 inet static
	vlan-raw-device eth1
	address 192.168.100.200
	netmask 255.255.255.0
	post-up route add -net 192.168.0.0/16 gw 192.168.100.254
	pre-down route del -net 192.168.0.0/16 gw 192.168.100.254
```

8) Na máquina WebServer, digite os seguintes comandos para ativar as interfaces:

```
ifup eth1
ifup vlan3652
```

9) Na máquina Controlador, após realizar as configurações acima deve-se realizar um teste de conectividade. Para isso, tente realizar um *ping* para as máquinas Cliente e Atacante:

```
ping -c 2 192.168.100.10
ping -c 2 192.168.1.2
```

Saída esperada:

```
root@Controlador:~# ping -c 2 192.168.100.10
PING 192.168.100.10 (192.168.100.10) 56(84) bytes of data.
64 bytes from 192.168.100.10: icmp_seq=1 ttl=64 time=0.555 ms
64 bytes from 192.168.100.10: icmp_seq=2 ttl=64 time=0.392 ms

--- 192.168.100.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.392/0.473/0.555/0.084 ms
root@Controlador:~# ping -c 2 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.624 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.382 ms

--- 192.168.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.382/0.503/0.624/0.121 ms
```

10) Na máquina Cliente, tente realizar um *ping* para a máquina Atacante. Esse *ping* não deve funcionar, visto que ainda não há roteamento Inter-AS:

```
ping 192.168.66.1
```

Saída esperada:

```
root@Cliente:~# ping -c 2 192.168.66.1
PING 192.168.66.1 (192.168.66.1) 56(84) bytes of data.

--- 192.168.66.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1008ms
```

Assim, no final deste experimento, observa-se que a topologia está configurada e que há conectividade entre os dispositivos e o controlador. Na próxima seção serão expostos os aspectos relacionados ao roteamento Inter-AS via BGP utilizando SDN/Openflow e um roteiro para verificação prática desses conceitos.

3. ROTEAMENTO INTER-AS VIA BGP EM SDN/OPENFLOW

Esta seção apresenta uma visão geral sobre os conceitos e funcionamento do roteamento inter-domínio (entre diferentes Sistemas Autônomos) utilizando o protocolo BGP e sua integração com a arquitetura SDN/Openflow.

3.1 Roteamento Inter-AS

Uma forma de analisar a Internet é observá-la como um conjunto de redes autônomas interconectadas, sob a mesma gerência técnica, compartilhando uma política de roteamento interno e externo bem definida (HAWKINSON; BATES, 1966). A esse conjunto de redes dar-se o nome Sistema Autônomo, ou AS (do inglês *Autonomous System*), e cada AS é identificado por um número de 16 ou 32 *bits* (também conhecido como ASN, do inglês *AS Number*). A partir desse conceito de AS, é possível dividir o roteamento na Internet em dois tipos: roteamento interior ao AS, cujos protocolos mais utilizados são OSPF, RIP e ISIS, e roteamento externo, ou roteamento entre AS's, cujo protocolo utilizado é o BGP (do inglês, *Border Gateway Protocol*).

Quando um par de sistemas autônomos concorda em trocar informações de roteamento, cada um precisa designar um roteador para conversar com o roteador do outro AS; neste caso, os dois roteadores se tornam "vizinhos BGP" um do outro. Em geral, o processo de roteamento em cada AS tem origem no armazenamento das rotas de redes pertencentes ao próprio AS, aprendidas através de protocolos de roteamento interno. Posteriormente, o roteador de borda (ou roteadores de borda) de cada AS inicia uma sessão BGP entre si. Neste caso, temos uma sessão BGP externa, ou simplesmente, *eBGP*. Caso a sessão BGP seja feita entre dois roteadores de um mesmo AS, então temos uma sessão BGP interna ou sessão *iBGP*. A Figura 3.1 mostra a troca de informações de roteamento entre AS's: o AS1 envia suas informações para os AS's vizinhos (AS2 e AS3) e essas informações são encaminhadas até chegar ao AS5. O AS5 armazena os caminhos para o AS1 e envia o melhor caminho como resposta.

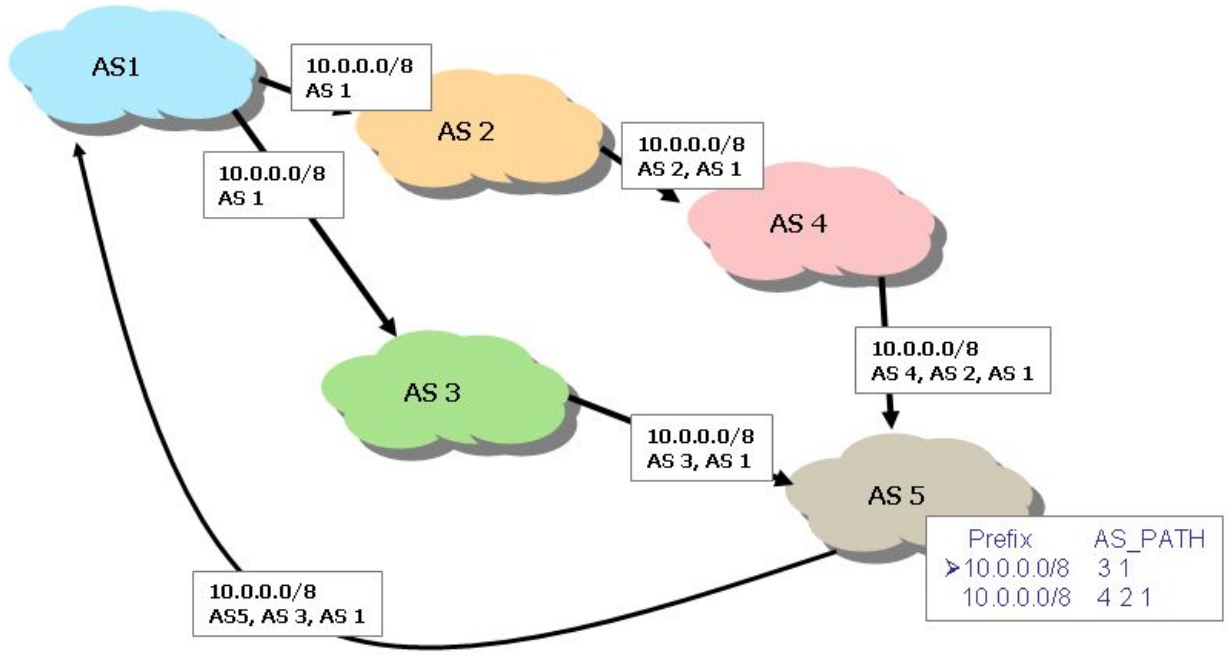


Figura 3.1 - Troca de informações de roteamento entre ASs

O BGP é um protocolo de roteamento bastante flexível e robusto, que foi projetado para ser escalável e evitar *loops* de roteamento em topologias arbitrárias. Ele destaca-se pela facilidade de definir roteamento baseado em políticas e por ser fundamentado em um conjunto de regras não técnicas definidas pelos Sistemas Autônomos.

Um roteador BGP constrói sua tabela de roteamento a partir de informações trocadas pelos "vizinhos BGP" (*BGP neighbors*), cuja atualização é feita de forma incremental. A atualização completa da tabela de roteamento é realizada apenas quando se estabelece a sessão entre *neighbors*, ou seja, apenas nesse primeiro momento a tabela completa é enviada entre os vizinhos. A flexibilidade e possibilidade de roteamento baseado em políticas do BGP estão intimamente ligadas ao conjunto de atributos que ele disponibiliza, bem como ao algoritmo de tomada de decisão na escolha de rotas. Alguns atributos do BGP são:

- AS-Path: seqüência de ASNs que uma rota cruza para alcançar uma determinada rede de destino - rotas com menor AS-path são preferidas;
- Local-Pref: atributo usado para influenciar na definição do caminho preferencial de saída para uma determinada rota, usado apenas entre roteadores iBGP;

- MED: o atributo MED (do inglês, *Multi Exit Discriminator*) tem por objetivo informar para os vizinhos BGP externos qual o melhor caminho para uma determinada rota do próprio AS, influenciando assim o tráfego de entrada do AS;
- Community: atributo utilizado para marcar as rotas com alguma característica que as agrupa (ex: rotas aprendidas a partir do cliente X, rotas que não devem ser repassadas, etc);
- Weight: apesar de não ser propriamente um atributo BGP, é utilizado para decisão local da rota de saída em um roteador BGP;
- Next-Hop: este atributo recebe o endereço IP da interface do próximo roteador, cujo valor varia de acordo com o tipo da sessão: eBGP ou iBGP.

A configuração do protocolo BGP pode ser dividida em configurações básicas e avançadas, e depende da plataforma em que se está configurando (Cisco, Juniper, Extreme, Brocade, Quagga, BIRD, GoBGP, etc). Considerando o escopo deste curso, serão abordadas apenas as configurações básicas demonstradas na plataforma Quagga. As tarefas de configuração básica consistem em:

- **Habilitar o roteamento BGP**: este passo é necessário para habilitar o processo do BGP no roteador em questão, indicando qual o ASN do provedor, o IP que identifica o roteador em que se está configurando e a rede que ele irá anunciar. Para isso use os seguintes comandos no modo de configuração global no console do roteador:
    - router bgp <asn> (ex: router bgp 100)
    - bgp router-id <ip-addr> (ex: bgp router-id 10.0.0.2)
    - network <prefix/mask> (ex: network 192.168.0.0/16)
- **Configurar vizinhos BGP (neighbors)**: é necessário configurar os vizinhos BGP manualmente, sejam eles eBGP ou iBGP. Na configuração do vizinho é preciso especificar o endereço IP e o número AS do roteador remoto, através do seguinte comando:
    - neighbor <ip-addr> remote-as <asn> (ex: neighbor 10.0.0.1 remote-as 200)

Outras configurações possíveis são: BGP Soft Reconfiguration (que permite alteração em configurações sem reiniciar a sessão BGP); BGP In/Out Policies (permite a configuração de políticas de filtragem BGP de entrada ou saída); etc.

3.2 Integração entre BGP e SDN/Openflow

Uma outra maneira de implantar a comunicação Inter-AS via BGP é através da arquitetura SDN/Openflow. A implementação de BGP através do *Openflow* tem sido bem sucedida em diversos casos demonstrados na indústria e na academia, permitindo a ampliação das capacidades do BGP, facilidade no gerenciamento de políticas e até mesmo escalabilidade. Alguns exemplos de iniciativas que têm sido conduzidas nesse sentido são listadas a seguir:

- RouteFlow[8]: é um projeto *open source* que tem o objetivo de prover serviços de roteamento IP virtualizados sobre qualquer *hardware* que tenha *Openflow* habilitado. A intenção é criar uma arquitetura de roteamento que possa ser executada em computadores de propósito geral, combinando o desempenho dos *hardwares* comerciais com a flexibilidade de uma camada de roteamento de código aberto. Assim, espera-se fornecer: migração de protocolos da rede legada para SDN; um *framework open-source* para virtualização de rede; modelos de rede do tipo RaaS (do inglês, *Routing-as-a-Service*); além de uma forma simples de fazer roteamento intra e inter-domínio que tenha interoperabilidade entre equipamentos legados.
- SIR – SDN Internet Router[9]: é um agente que pode ser adicionado ao roteador para tornar visível e disponibilizar informações que não poderiam ser expostas por si só, como tabelas BGP e tráfego por prefixo BGP ou por ASN. Essas informações podem ser utilizadas para fazer engenharia de tráfego, planejamento de capacidade, melhorar a decisão das rotas instaladas na FIB, entre outras soluções, e podem ser utilizadas em diferentes equipamentos pois utilizam BGP e netflow/sflow/ipfix na obtenção dos dados.
- Software Defined Internet Exchange Point[10] (SDX): Tenta trazer os benefícios de SDN para o roteamento interdomínio, principalmente em IXPs (do inglês, *Internet Exchange Point*), devido à sua capacidade de conectar diversas redes e de deixar os conteúdos mais populares mais próximos dos usuários. Com a utilização de SDN, é possível oferecer um controle direto sobre as regras de processamento de pacotes, considerando diversos campos dos cabeçalhos de rede e uma grande variedade de

[8] http://routeflow.github.io/RouteFlow/
[9] https://github.com/dbarrosop/sir
[10] http://sdx.cs.princeton.edu/
https://noise-lab.net/projects/software-defined-networking/sdx/

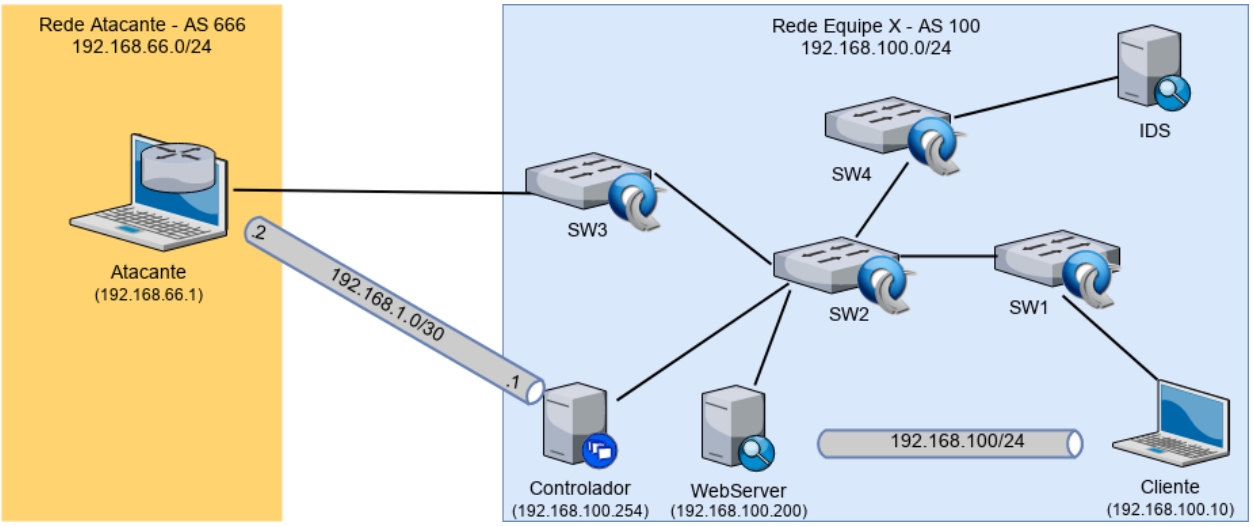
ações. Assim, a proposta de SDX tem a capacidade de habilitar novas aplicações, tais como: troca de tráfego específico por aplicação, por exemplo com um *peering* apenas para tráfego de streaming de vídeo; escalabilidade, em relação ao tamanho da tabela e à sobrecarga computacional; engenharia de tráfego de entrada, inclusive utilizando políticas administrativas; redirecionamento de tráfego para *middleboxes*; servidores de balanceamento de carga; bloqueio de tráfego indesejado, etc.

3.3 Exercícios de fixação

1. O que é um AS e como os AS's se comunicam na Internet?
2. Quais são os principais atributos para seleção de rotas BGP?
3. Como SDN pode ser empregada no roteamento Inter-AS? Quais os benefícios?
4. Como o Controlador pode configurar as rotas BGP nos comutadores OpenFlow?

3.4 Roteiro de laboratório

O objetivo deste laboratório é construir e utilizar uma aplicação SDN/Openflow para estabelecimento de sessões BGP no ambiente de teste FIBRE entre o AS 100 (rede do aluno) e AS 666 (rede hipotética do atacante). A topologia utilizada neste laboratório é a mesma anterior, conforme Figura 3.2.



3.2. Topologia proposta para os experimentos da Oficina.

Grande parte das configurações desta prática serão executadas através do console de alguma das máquina virtuais (Controlador, Atacante ou Cliente), através uma conexão SSH. Para saber o endereço IP da máquina você deverá verificar a interface do OCF.

3.4.1 Integrando BGP na aplicação SDN-IPS

A aplicação SDN-IPS é baseada no controlador Ryu versão 4.15, sendo divida em dois principais componentes: i) SDNIPSApp, classe responsável pelo tratamento dos eventos Openflow, por armazenar e gerenciar a topologia da rede e por atender a requisições de caminho na instalação de fluxos através da API sul baseada em Openflow 1.0; ii) SDNIPSWSGIApp, classe responsável pela interface REST da aplicação, que recebe as requisições da API norte do usuário e requisita serviços à aplicação SDNIPSApp. Nesta prática o aluno deverá alterar a classe SDNIPSApp e implementar as funções que são utilizadas para estabelecimento da sessão BGP, a saber: bgp_create(), bgp_add_neighbor() e bgp_add_prefix(). Ambas as funções farão uso da biblioteca BGP Speaker do Ryu, conforme demonstrado a seguir.

1) O primeiro passo é editar a função bgp_create() a fim de programá-la para que ela utilize a biblioteca do Ryu para configurar os parâmetros básicos do BGP (número AS e ID do roteador). Modifique a função bgp_create() no arquivo fibre-2opencall-sdn-ips.py complementando-a para que ela fique como mostrado abaixo (altere a parte que possui um comentário “# MUDAR AQUI - INICIO” até “# MUDAR AQUI - FIM”):

```
def bgp_create(self, as_number, router_id):
    # MUDAR AQUI - INICIO
    try:
        self.bgp_speaker = BGPSpeaker(as_number=as_number, router_id=router_id,
                    best_path_change_handler=self.best_path_change_handler,
                    peer_down_handler=self.peer_down_handler,
                    peer_up_handler=self.peer_up_handler)
    except:
        return (False, 'Failed to create BGP speaker')
    # MUDAR AQUI - FIM
```

```
    ...
```

Atente-se para respeitar as indentações do código original e do trecho acima. Em linhas gerais esse código apenas cria uma instância da classe BGPSpeaker(), configurando o número AS e o ID do roteador. Deve-se notar também a presença de três funções para tratar eventos do BGP, a saber: best_path_change_handler() - chamada sempre que uma rota é inserida ou removida (parâmetro is_withdraw=True) no processo de aprendizagem do BGP; peer_down_handler() - chamada sempre que uma vizinhança BGP é estabelecida; e peer_up_handler() - sempre que uma vizinhança BGP é desfeita. Essas funções são importantes para que a aplicação SDN faça alguma modificação no comportamento padrão do BGP (conforme exemplos apresentados na seção de aprendizagem 3.2). Estas funções já foram previamente definidas no código, recomenda-se que o leitor analise-as para entender sua lógica de funcionamento.

2) Em seguida, é necessário definir a função de configuração de prefixos BGP. Para isso, modifique a função bgp_add_prefix() no arquivo fibre-2opencall-sdn-ips.py complementando-a para que ela fique como mostrado abaixo (altere a parte que possui um comentário "# MUDAR AQUI - INICIO" até "# MUDAR AQUI - FIM"):

```
def bgp_add_prefix(self, prefix):
    # MUDAR AQUI - INICIO
    try:
        self.bgp_speaker.prefix_add(prefix)
    except:
        return (False, 'Failed to add prefix')
    # MUDAR AQUI - FIM
    ...
```

Atente-se para respeitar as indentações do código original e do trecho acima. O leitor irá notar que a função acima faz uso da instância do BGP Speaker criada anteriormente e apenas insere um novo prefixo aos anúncios BGP do roteador atual. Vale salientar que os prefixos anunciados via BGP nesta biblioteca podem ser de qualquer família de endereços (IPv4, IPv6, L2VPNv4, etc), embora este laboratório limitar-se-á à IPv4.

3) Por fim, será definida a função que faz o registro de novos vizinhos BGP. A partir das informações do vizinho (endereço IP e número AS remoto), a biblioteca BGP Speaker tenta estabelecer a sessão BGP. Para isso é preciso modificar a função bgp_add_neighbor() no arquivo fibre-2opencall-sdn-ips.py complementando-a para que fique conforme mostrado abaixo (altere a parte que possui um comentário "# MUDAR AQUI - INICIO" até "# MUDAR AQUI - FIM"):

```
def bgp_add_neighbor(self, address, remote_as):
    # MUDAR AQUI - INICIO
    try:
        self.bgp_speaker.neighbor_add(address, remote_as)
    except:
        return (False, 'Failed to add BGP neighbor')
    # MUDAR AQUI - FIM
    ...
```

Atente-se para respeitar as indentações do código original e do trecho acima. O leitor irá notar que a função acima faz uso da instância do BGP Speaker criada anteriormente e apenas insere uma chamada à função de adição de vizinhos do BGP Speaker. Vale destacar que esta função possui diversos parâmetros adicionais (ex: modo de conexão passivo ou ativo, senha MD5, valor do atributo MED, conjunto de *address families* suportados, etc), recomenda-se a leitura da API do BGP Speaker para mais informações.

3.4.2 Configurando a sessão BGP do AS 100 na aplicação SDN-IPS

1) Esta prática pressupõe que o roteiro anterior do capítulo tenha sido executado com sucesso. Portanto, caso tenha desligado o ambiente ao final da prática anterior, é necessário religar o controlador Ryu. Ele irá recarregar as configurações que foram realizadas anteriormente a partir do arquivo **`sdn-ips-config.json`** na mesma pasta da aplicação.

2) Na máquina Controlador (em outra aba), vamos utilizar a API REST da aplicação SDN-IPS para fazer a configuração da sessão BGP. O primeiro passo é configurar o ASN e Router-ID da sessão BGP. Para isso execute o seguinte comando:

```
curl -s -X POST -d '{"as_number" : 100, "router_id" : "192.168.1.1"}' \
  http://localhost:8080/sdnips/bgp/create | python -m json.tool
```

3) O próximo passo é configurar as redes que serão anunciadas na sessão BGP, a saber o prefixo de rede do AS 100 (192.168.100.0/24). Para isso execute o seguinte comando:

```
curl -s -X POST -d '{"prefix" : "192.168.100.0/24"}' \
  http://localhost:8080/sdnips/bgp/add_prefix | python -m json.tool
```

4) Por fim, deve-se adicionar o vizinho BGP com o qual deseja-se comunicar, ou seja, o roteador do AS 666. É necessário especificar, portanto, o endereço IP do roteador vizinho (192.168.1.2) e o número do AS remoto (666). Para isso execute o seguinte comando:

```
curl -s -X POST -d '{"remote_as" : 666, "address" : "192.168.1.2"}' \
  http://localhost:8080/sdnips/bgp/add_neighbor | python -m json.tool
```

5) É possível acompanhar nos logs do Ryu que a aplicação SDN-IPS está tentando estabelecer a sessão BGP com o peer remoto, porém sem sucesso (de fato essa sessão será configurada na etapa seguinte). Para visualizar esse comportamento, volte ao console em que a aplicação do Ryu foi iniciada (passo 1) e observe as mensagens de "*Will try to reconnect to 192.168.1.2 after 30 secs*":

```
(460) accepted ('127.0.0.1', 51248)
API method core.start called with args: {'router_id': '192.168.1.1', 'label_range': (100, 100000), 'waiter': <ryu.lib.hub.Eve
: 179, 'local_as': 100, 'allow_local_as_in_count': 0, 'refresh_stalepath_time': 0, 'cluster_id': None, 'local_pref': 100, 're
127.0.0.1 - - [30/Nov/2017 15:54:32] "POST /sdnips/bgp/create HTTP/1.1" 200 119 0.029547
(460) accepted ('127.0.0.1', 51249)
API method network.add called with args: {'prefix': '192.168.100.0/24'}
127.0.0.1 - - [30/Nov/2017 15:54:32] "POST /sdnips/bgp/add_prefix HTTP/1.1" 200 119 0.003864
(460) accepted ('127.0.0.1', 51250)
API method neighbor.create called with args: {'connect_mode': 'both', 'cap_mbgp_evpn': False, 'remote_as': 666, 'cap_mbgp_vpn
r_octet_as_number': True, 'cap_mbgp_ipv6': False, 'is_next_hop_self': False, 'cap_mbgp_ipv4': True, 'cap_mbgp_ipv4fs': False,
ipv6fs': False, 'is_route_server_client': False, 'cap_enhanced_refresh': False, 'peer_next_hop': None, 'password': None, 'ip_
lse, 'cap_mbgp_vpnv4': False, 'cap_mbgp_vpnv6fs': False}
127.0.0.1 - - [30/Nov/2017 15:54:32] "POST /sdnips/bgp/add_neighbor HTTP/1.1" 200 119 0.003354
Will try to reconnect to 192.168.1.2 after 30 secs: True
Will try to reconnect to 192.168.1.2 after 30 secs: True
```

3.4.3 Configuração a sessão BGP do AS 666 com o Quagga

O Quagga é uma plataforma de roteamento em software bastante utilizada que provê implementação de diversos protocolos como OSPF, RIP e BGP na plataforma Unix (Linux, BSD, Solaris, etc). A arquitetura do Quagga consiste em diversos daemons, em particular:

zebra, o daemon principal do Quagga que atua como uma camada de abstração com a pilha de rede do Kernel Linux, permitindo por exemplo instalação das rotas na FIB; bgpd, que é o daemon utilizado para estabelecer as sessões BGP.

Neste laboratório o Quagga será executado na máquina Atacante, portanto deve-se acessar essa máquina via SSH para executar os comandos abaixo.

1) Na máquina Atacante, o primeiro passo é instalar o Quagga. Para isso, na máquina Atacante execute o seguinte comando:

```
su
apt-get install quagga telnet tcpdump
```

2) Em seguida deve-se criar a configuração mínima para funcionamento do daemon zebra. Para isso, modifique o arquivo /etc/quagga/zebra.conf com o seguinte comando:

```
cat >/etc/quagga/zebra.conf <<EOF
hostname as666
password zebra
log file /var/log/quagga/zebra.log
ip forwarding
ipv6 forwarding
line vty
EOF
```

3) Deve-se criar, também, a configuração mínima para funcionamento do daemon bgpd. Para isso, modifique o arquivo /etc/quagga/bgpd.conf com o seguinte comando:

```
cat >/etc/quagga/bgpd.conf <<EOF
hostname as666
password zebra
log file /var/log/quagga/bgpd.log
line vty
EOF
```

4) Em seguida deve-se habilitar os daemons do quagga que serão executados (zebra e bgpd). Para isso, modifique o arquivo /etc/quagga/daemons deixando-o da seguinte maneira:

```
zebra=yes
bgpd=yes
```

5) Agora deve-se ajustar as permissões dos arquivos e iniciar o serviço Quagga:

```
chown quagga:quagga /etc/quagga/*
chmod 640 /etc/quagga/*
/etc/init.d/quagga restart
```

6) O próximo passo é fazer a configuração do roteador BGP do AS 666, através do console do quagga. Para acessar o console do bgpd no Quagga, execute o seguinte comando (ao ser questionado pela senha, informe a senha configurada anteriormente “zebra” - sem aspas):

```
telnet localhost 2605
```

7) Para fazer a configuração BGP do AS 666 é necessário configurar o Router-ID, número AS local, rede anunciada e vizinhos BGP. Deve-se fazer isso a partir do console do bgpd aberto no passo anterior. No console do bgpd execute os seguintes comandos:

```
enable
configure terminal
 router bgp 666
  bgp router-id 192.168.1.2
  network 192.168.66.0/24
  neighbor 192.168.1.1 remote-as 100
  neighbor 192.168.1.1 soft-reconfiguration inbound
  exit
 exit
write memory
```

A saída esperada é algo como ilustrado abaixo:

```
root@Atacante:/home/rnp/[redacted]# telnet localhost 2605
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.23.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
as666> enable
as666# configure terminal
as666(config)#  router bgp 666
as666(config-router)#  bgp router-id 192.168.1.2
as666(config-router)#  network 192.168.66.0/24
as666(config-router)#  neighbor 192.168.1.1 remote-as 100
as666(config-router)#  neighbor 192.168.1.1 soft-reconfiguration inbound
as666(config-router)#  exit
as666(config)#  exit
as666# write memory
Configuration saved to /etc/quagga/bgpd.conf
as666#
```

3.4.4 Testando a conectividade entre o AS 100 e AS 666

1) É possível visualizar nos logs da aplicação Ryu SDN-IPS que a sessão BGP foi estabelecida. Para isso, volte ao console do Controlador SDN e observe uma saída similar ao ilustrado abaixo:

```
Will try to reconnect to 192.168.1.2 after 30 secs: True
Connection to peer: 192.168.1.2 established
Peer up: 192.168.1.2 666
the best path changed: remote-as=666 prefix=192.168.66.0/24 next-hop=192.168.1.2 action=add
net.ipv4.ip_forward = 1
```

2) É possível verificar também as sessões BGP ativas no console do Quagga. Para isso, volte ao console do Atacante e execute os seguintes comandos:

```
show ip bgp summary
show ip bgp neighbors 192.168.1.1 received-routes
```

A saída esperada é algo como ilustrado abaixo:

```
as666# show ip bgp summary
BGP router identifier 192.168.1.2, local AS number 666
RIB entries 3, using 216 bytes of memory
Peers 1, using 2528 bytes of memory

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
192.168.1.1     4   100      20      23        0    0    0 00:04:15        1

Total number of neighbors 1
as666#
```

```
as666# sh ip bgp neighbors 192.168.1.1 received-routes
BGP table version is 0, local router ID is 192.168.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.100.0    192.168.1.1                            0 100 i

Total number of prefixes 1
```

3) Por fim, a partir da máquina Cliente deve-se executar um teste de conectividade através de PING. Acesse o console da máquina cliente via SSH e execute os seguintes comandos:

```
ping -c 2 192.168.66.1
traceroute -n 192.168.66.1
```

A saída esperada é algo como ilustrado abaixo:

```
root@Cliente:~# ping -c 2 192.168.66.1
PING 192.168.66.1 (192.168.66.1) 56(84) bytes of data.
64 bytes from 192.168.66.1: icmp_seq=1 ttl=63 time=1.47 ms
64 bytes from 192.168.66.1: icmp_seq=2 ttl=63 time=0.840 ms

--- 192.168.66.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.840/1.158/1.477/0.320 ms
root@Cliente:~# traceroute -n 192.168.66.1
traceroute to 192.168.66.1 (192.168.66.1), 30 hops max, 60 byte packets
 1  192.168.100.254  0.578 ms  0.556 ms  0.538 ms
 2  192.168.66.1  1.097 ms  1.084 ms  1.034 ms
root@Cliente:~#
```

Com isso finaliza-se o módulo a conectividade BGP entre os dois AS's de forma integrada ao SDN/Openflow. Na próxima seção serão expostos os principais aspectos relacionados ao uso de sistemas de detecção de intrusão e como ativá-lo de forma integrada ao ambiente SDN ora proposto.

4. SISTEMAS DE DETECÇÃO DE INTRUSÃO

Esta seção apresenta os conceitos relacionados com as soluções de Detecção de Intrusos, que funcionam como um mecanismo de defesa adicional na proteção da segurança da informação nas organizações. Serão apresentados os principais tipos de sistemas IDS e opções de ferramenta para cada tipo, com foco especial na ferramenta Suricata, que será utilizada no laboratório prático deste curso. Por fim, aborda-se em linhas gerais o gerenciamento de regras de assinaturas de ataques.

4.1 Visão geral sobre IDS

A complexidade dos protocolos de rede, a evolução na complexidade dos ataques e o aumento significativo nas atividades cibernéticas maliciosas têm preocupado as organizações em termos de defesas tecnológicas (Mahurin, 2017). Uma das proteções mais utilizadas para segurança de redes e sistemas é a segmentação da rede em zonas (ex: interna, internet e DMZ) e a instalação de soluções de proteção de perímetro como Firewalls. Considerando que nenhum sistema é 100% seguro, existe um risco de que alguns ataques não sejam bloqueados pela solução de perímetro, portanto fazem-se necessárias outras barreiras de proteção.

Desta forma, tecnologias como Sistemas de Detecção de Intrusos (IDS) tem emergido como mecanismo adicional de segurança contra diferentes ataques: proteção contra *malwares* modernos, intrusão de redes, ameaças avançadas persistentes e controle de aplicações (Stuart & Beaver, 2013). Além de características clássicas como análise de assinaturas e análise de comportamento da rede e das aplicações, os sistemas de IDS modernos são caracterizados por incorporarem mecanismos de Inteligência contra Ameaças (*Threat Intelligence*) para bloqueio automático de URL/DNS/IP em sites de baixa reputação, inspeção profunda de pacotes, análise de aplicações e arquivos, etc.

4.2 Tipos de IDS

A maioria dos sistemas IDS utiliza um modelo baseado na captura de pacotes, normalização do tráfego, pré-processamento de aplicações e análise pelo motor de inspeção. O motor de inspeção, por sua vez, compara o tráfego observado com um conjunto de assinaturas

pré-definidas para aquele tipo de aplicação e verifica se o comportamento está em conformidade com o desejado. Outros motores de inspeção fazem a comparação com um perfil de tráfego previamente mapeado (análise baseada em comportamento). Através desse monitoramento constante do tráfego, pode-se tomar uma ação caso alguma atividade suspeita seja detectada. Esta ação pode ser desde um alerta ao administrador até o bloqueio temporário ou permanente do atacante. Em relação às informações coletadas, monitoramento, avaliação e tomada de decisões, os IDS podem ser classificados da seguinte forma:

- **IDS de rede**: sistemas de detecção de intrusão baseados em rede (NIDS) coletam informações através do tráfego de rede da organização, entre seus diferentes segmentos (seja o tráfego norte/sul - entrada e saída da internet - seja o tráfego leste/oeste - comunicação entre VLANs internas). O IDS de rede checa os ataques ou "comportamentos maliciosos" através da inspeção dos cabeçalhos (headers) e do conteúdo (payload) dos pacotes. A checagem ocorre através da correspondência com "assinaturas de ataque" que são regras que descrevem o padrão de tráfego associado a um comportamento malicioso específico. Geralmente, a maioria das soluções de NIDS oferece a opção de definir regras customizadas com assinatura de ataques ou comportamentos particulares da organização, por exemplo: uma regra customizada que bloqueie o acesso a sites de *phishing* dirigidos aos usuários da organização, ou ainda uma regra que detecta tentativas de intrusão em uma aplicação web da organização que possui uma vulnerabilidade específica. Cabe, portanto, ao NIDS comparar as assinaturas de ataque com os pacotes capturados para identificar tráfego hostil. A inspeção de tráfego pode ocorrer de duas maneiras: i) espelhamento de tráfego para o sistema NIDS ou ii) posicionando o equipamento NIDS *inline* nos segmentos de rede que se deseja monitorar. Cada abordagem possui vantagens e desvantagens, por exemplo: em ambientes com espelhamento o desempenho da rede não é prejudicado pela sobrecarga de inspeção de pacotes, por outro lado, ataques que consistem no envio de mensagens especialmente criadas para colapsar a aplicação não são barradas até que o NIDS execute a ação de contenção. Exemplos de NIDS são: Suricata[11], Snort [12] e Bro IDS[13];

[11] https://suricata-ids.org/
[12] https://www.snort.org/
[13] https://www.bro.org/

- **IDS de host**: sistemas de detecção de intrusão baseados em host (HIDS) atuam coletando informações e atividades de segurança em um sistema específico, geralmente através de um agente instalado no sistema. O termo “host” se refere à um computador individual, de forma que faz-se necessário um sensor para cada sistema que será monitorado. Um HIDS coleta dados a partir de eventos que ocorrem no sistema operacional, processos executando, arquivos, portas de rede e, principalmente, trilhas de auditoria (logs). Algumas vantagens dos HIDS em relação a NIDS incluem: capacidade de auditar e correlacionar eventos internos do sistema; de analisar tráfego que a nível de rede poderia estar criptografado; e de mapear os eventos sendo executados com usuários do sistema, a fim de identificar intrusos internos (insiders). Exemplos de HIDS são OSSEC [14]e Tripware[15];
- **IDS baseado em anomalia**: o IDS baseado em anomalia é um tipo especial do NIDS em que, invés de analisar o tráfego de rede em comparação com “assinaturas”, o sistema mapeia o perfil de rede considerado “normal” e emprega técnicas como redes neurais, aprendizado de máquina, modelos estatísticos, dentre outros, para identificar um tráfego anômalo. Faz-se necessário, portanto, uma fase de monitoramento e aprendizagem para traçar o perfil de acesso “normal” para que o sistema passe a identificar o comportamento malicioso. Uma vantagem desse tipo de IDS é sua capacidade de identificar ataques “zero-day”, ou seja, ataques que não possuem qualquer tipo de informação antes de sua exploração. Não obstante, uma desvantagem é a geração de um grande número de alarmes falsos devido ao comportamento imprevisível de usuários e do próprio sistema. Um exemplo de solução de IDS baseado em anomalia é o Hogzilla[16], além de inúmeros outros projetos acadêmicos;
- **IDS de kernel**: nesse tipo de IDS a detecção de comportamento malicioso é feita através da análise de chamadas de sistema, bibliotecas do S.O., movimentação de registradores, enfim, de um conjunto de rotinas internas ao kernel do sistema. Embora pouco comuns, esse tipo de IDS vem ganhando atenção recentemente com os ataques de Ransomware, no qual os arquivos do usuário são criptografados e o atacante solicita resgate. Como a execução de funções criptográficas são legítimas no sistema, uma análise mais aprofundada a nível do kernel se faz necessária para identificar tal

[14] https://ossec.github.io/
[15] https://github.com/Tripwire/tripwire-open-source
[16] http://ids-hogzilla.org/

atividade maliciosa. Soluções comerciais como Checkpoint[17], Kaspersky[18], dentre outras, possuem tal funcionalidade. Algumas ferramentas *open source* como SystemTap[19] e strace[20] também se encaixam nessa categoria;

- **NGIDS**: esse é um tipo de IDS considerado de "nova geração", cujas funcionalidades englobam as outras categorias previamente relacionadas. Além disso, tem capacidade de tratar ameaças avançadas persistentes (APTs), possuem correlação de eventos (SIEM) e aplicam técnicas de ciber inteligência contra ameaças (CTI). Ademais, o esforço de configuração nessas ferramentas é reduzido se comparado às outras abordagens, posto que os fabricantes empregam recomendações de configuração que são mais comumente observadas em organizações do mesmo setor. Existem diversos exemplos de soluções comerciais que se enquadram nessa categoria. O leitor pode obter mais informações consultando o quadrante de IPS do Gartner.

Atualmente, são encontradas diferentes soluções de IDS/IPS disponíveis, sendo elas comerciais, software livre e acadêmicas. Muitos fabricantes têm investido em produtos que minimizem a necessidade de customizações e otimizem a capacidade de proteção da solução. Em termos comerciais, o Gartnet mantém uma avaliação das soluções de IDS/IPS[21], que pode ser usada pelas organizações como *baseline* de escolha. Existem também algumas soluções abertas, que são comumente utilizadas e efetivas para as organizações. Entre as principais soluções de IDP/IPS *open source* encontram-se: i) Snort - um NIDS open source, bastante conhecido, baseado em assinaturas e com estrutura modular altamente customizável, de modo que diversos plugins podem ser usados para expandir suas funcionalidades (ex: reagir a um alerta, a atualização automática das suas assinaturas e o gerenciamento de diversos sensores espalhados em uma ou mais redes); ii) Suricata - um NIDS baseado em assinatura, derivado do Snort, cujo objetivo é se tornar um padrão para ambientes de alta demanda de tráfego e desempenho; e iii) Bro - um NIDS que monitora de forma passiva o tráfego de rede em busca de atividades suspeitas, tanto com base em assinaturas quanto em termos de eventos e atividades incomuns (por exemplo, certo host conectar a determinados serviços ou falhas em

[17] https://www.checkpoint.com/
[18] https://www.kaspersky.com.br/
[19] https://sourceware.org/systemtap/
[20] https://strace.io
[21] https://www.gartner.com/reviews/market/intrusion-prevention-systems

tentativas de conexão). Neste curso será utilizada a ferramenta Suricata, que tem apresentado bom desempenho no processamento de grande volume de tráfego.

O Suricata foi desenvolvido pelo OISF (*Open Information Security Foundation*) em 2010 e, apesar de menos difundido que o Snort, por exemplo, possui uma capacidade de processamento de tráfego na ordem de 10 Gbps e um motor capaz de executar funções avançadas de detecção de intrusos. Considerado um IDS de "Nova Geração", as funcionalidades avançadas do Suricata incluem: suporte a multi-threading; suporte a detecção automática de diversos protocolos (ex: IP, TCP, UDP, ICMP, HTTP, TLS, FTP, SMB e tantos outros da camada de aplicação), independente da porta; suporte a descompressão Gzip; biblioteca HTB independente para processamento de tráfego HTTP (mesma biblioteca utilizada pelo Modsecurity - uma importante solução de Firewall Web); consulta e atualização de listas de reputação IP; dentre outros.

4.3. Gerenciamento de Regras do IDS

O IDS funciona com base em assinaturas de ataques, geralmente de forma compatível com as regras utilizadas pelo projeto Snort. Isso permite a utilização de regras desenvolvidas pela comunidade ou por empresas especializadas, além da definição de regras customizadas para a organização, a fim de identificar atividades suspeitas específicas do ambiente. A forma de definição de uma regra genérica é mostrada na Figura 4.1: cada regra é composta por uma ação (em vermelho), cabeçalhos do pacote que definem qual tráfego analisar (verde) e opções de casamento (em azul) que permitem fazer buscas também dentro do pacote. As ações que são aplicadas ao tráfego quando uma regra é acionada podem ser *block*, *alert* e *pass*.

```
alert tcp 10.0.0.31 any -> any 80
(msg:"HTTP Google Access";
content:"google.com"; http_uri;
classtype:web-access; sid:1; rev:1;)
```

Figura 4.1 - Formato de definição de regras do Suricata

É comum o agrupamento das regras do IDS em categorias, a fim de facilitar a busca e resolução de problemas, além de especializar o conjunto de métodos de monitoramento que está relacionado a cada contexto. Uma forma de organizar as regras de assinatura é

baseando-se em categorias como: Sistema Operacional (Windows, Linux, Solaris, etc); protocolos (TCP, UDP, TFTP, RPC, POP, IMAP, etc); Servidores (IIS, Apache, Tomcat, MSSQL, MySQL, Oracle, Samba, DNS, etc); entre outras. Dessa forma o administrador pode facilmente escolher quais regras serão ou não habilitadas para cada subrede de acordo com suas características individuais.

As regras de assinaturas de ataques desempenham um papel fundamental no processamento do Suricata. Através dessas regras é possível identificar ataques que são comuns às diferentes organizações. Existem diversos conjuntos de regras disponíveis para integração à ferramenta de IDS, alguns de acesso livre e outros com restrições comerciais. A seguir uma descrição sobre os conjuntos de regras mais utilizados:

- Regras Snort Community[22] - conjunto de regras com acesso aberto e mantidas pela comunidade Snort;
- Regras Snort Subscriber[23] - conjunto de regras VRT / Talos (Vulnerability Research Team) , mantida pela Sourcefire/Cisco, em que os clientes registrados conseguem acesso às regras com antecedência de até 30 dias em comparação com a versão Community;
- Regras do Emerging Threats[24] - regras comunitárias e gratuitas, desenvolvidas por voluntários. Apesar de gratuitas, as regras ET são muito eficientes e possuem um elevado índice de atualizações;
- Regras do Emerging Threats Pro[25] - versão comercial do ET mantida pela empresa Proofpoint, oferecendo maior grau de precisão e rapidez na distribuição de assinaturas de novos ataques.

Essas regras são mantidas pela comunidade e por empresas como Talos/Cisco, Proofpoint, entre outras, elas são atualizadas frequentemente e fornecem uma boa fonte de assinaturas para ataques conhecidos. Uma das formas de instalar e manter as regras supracitadas é através de ferramentas de Gerência de Regras como o software Oinkmaster[26]. Através da ferramenta Oinkmaster o administrador pode automaticamente ativar ou desativar um conjunto

[22] https://www.snort.org/downloads#rules
[23] https://www.snort.org/products#rule_subscriptions
[24] http://doc.emergingthreats.net/bin/view/Main/AllRulesets
[25] https://www.proofpoint.com/us/threat-insight/et-pro-ruleset
[26] http://oinkmaster.sourceforge.net/

de regras, para evitar falsos positivos, gerenciar assinaturas de conjuntos de regras ou ainda utilizar diferentes fontes de obtenção das assinaturas.

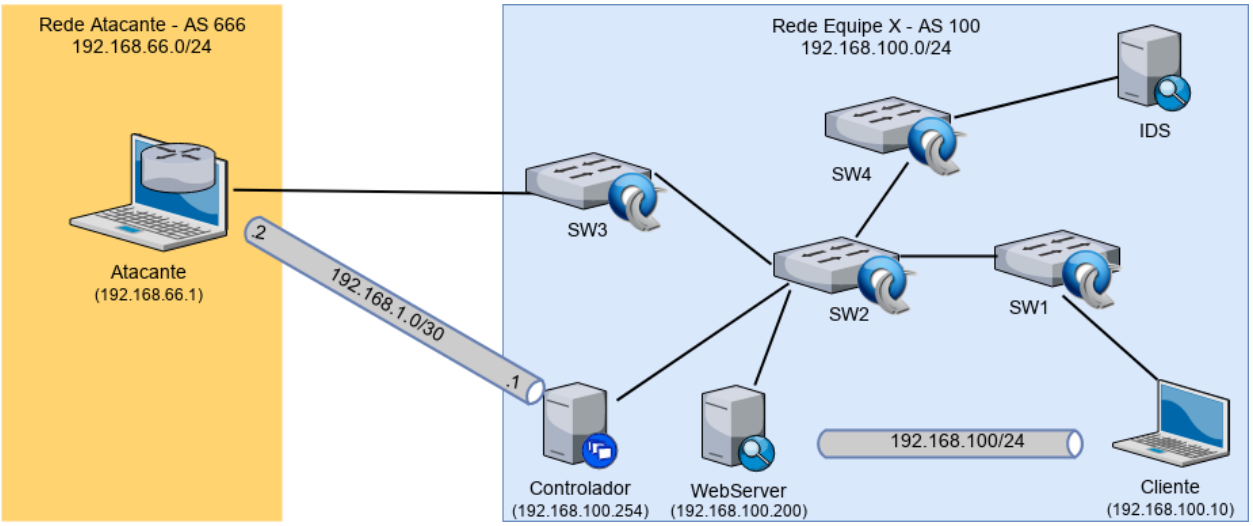
Por fim, as regras são um elemento-chave na configuração do IDS. Um conjunto de regras bem estabelecido pode ser o diferencial para uma maior capacidade de identificação de ataques. Por outro lado, muitas regras podem gerar falsos positivos ou ainda violar a privacidade dos usuários (registrando logs além do necessário), então fazer um ajuste das regras é uma tarefa cansativa, porém indispensável para que os registros de alerta sejam confiáveis e úteis. Um IDS que gera muitos alertas falsos facilmente acaba em desuso e um IDS sem regras atualizadas e relevantes ao perfil de tráfego traz baixo benefício para a organização.

4.4 Exercícios de Fixação

1. O que é um IDS e como ele funciona?
2. Quais os tipos de IDS?
3. Quais os riscos inerentes ao gerenciamento de assinaturas de IDS?
4. Considerando o cenário de rede baseada em SDN proposta neste laboratório, como viabilizar o encaminhamento do tráfego para o servidor IDS?

4.5 Roteiro de Laboratório

O objetivo deste laboratório é instalar e configurar um sistema de detecção de intrusão para monitoramento do tráfego interno e externo do AS 100 através da ferramenta Suricata IDS. Para isso, além da instalação do Suricata, deve-se habilitar o espelhamento de tráfego no switch Openflow para o Servidor IDS. A topologia utilizada neste laboratório é a mesma anterior, conforme Figura 4.2.



4.2. Topologia proposta para os experimentos da Oficina.

4.5.1 Instalação e configuração do Suricata

1) A instalação do Suricata será realizada a partir dos repositórios Debian, evitando a necessidade de compilar seu código fonte. Para isso, utilizaremos a versão "backport" do Debian Jessie, que fornece o Suricata na versão 3.2. Antes de instalar será necessário atualizar a lista de pacotes. Para isso acesse a máquina IDS e execute:

```
su -
echo "deb http://debian.pop-sc.rnp.br/debian/ jessie-backports main contrib non-free" >>
/etc/apt/sources.list
apt-get update
```

2) Em seguida deve-se proceder com a instalação do Suricata:

```
apt-get -t jessie-backports install suricata tcpdump curl python-ipcalc
```

3) Após instalar o suricata o próximo passo é obter as regras que serão utilizadas. Neste laboratório vamos trabalhar com as regras community da Emerging Threats. Para obtê-las, basta executar o seguinte comando:

```
/usr/sbin/suricata-oinkmaster-updater
```

4) Além das regras do Emerging Threats, vamos criar uma regra customizada para identificar ataques de Negação de Serviço (DoS) baseado em TCP Syn Flood para nossa organização. Observe que esse tipo de regra é baseada em limiares (threshold), portanto sua correta configuração depende de um estudo prévio do perfil de tráfego da organização para identificar o limiar mais adequado. Para fins ilustrativos, consideraremos que um limiar de 40 pacotes/seg de TCP-SYN de uma mesma origem externa é considerado ataque. Para isso, crie o arquivo **/etc/suricata/rules/local.rules** com o seguinte conteúdo:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"Possible TCP Syn Flood DoS"; flags:S; flow:to_server; threshold: type both, track by_src, count 40, seconds 5; classtype:attempted-dos; sid:10001; rev:1;)
```

5) Em seguida vamos realizar a configuração do servidor Suricata, editando o arquivo **/etc/suricata/suricata.yaml**. O primeiro passo na edição desta configuração é definir o endereçamento de rede da organização (HOME_NET). Essa definição é importante para que o IDS possa diferenciar o sentido do tráfego (acessos internos *versus* acessos externos, etc). A configuração deverá ficar da seguinte forma:

```
vars:
  address-groups:
    HOME_NET: "[192.168.100.0/24]"
...
```

6) O próximo passo, ainda no arquivo **/etc/suricata/suricata.yaml**, é selecionar quais regras deseja ativar ou desativar. Vamos comentar algumas linhas (adicionando o caracter # no início), descomentar outras (retirando o caracter # do início) e adicionar a regra criada. Ao final das mudanças a configuração deve ficar da seguinte forma:

```
default-rule-path: /etc/suricata/rules
rule-files:
 - botcc.rules
 - emerging-scan.rules
 - local.rules
```

7) O passo seguinte, ainda no arquivo **/etc/suricata/suricata.yaml**, é configurar as opções de captura de pacotes do sistema IDS. Para isso edite o arquivo substituindo a interface eth0 pela eth1 (outro parâmetros não mencionados devem ser mantidos com seus valores originais):

```
af-packet:
  - interface: eth1
...
pcap:
  - interface: eth1
```

8) O próximo passo da configuração é ativar a interface de rede em que o tráfego será capturado. Para isso, edite o arquivo **/etc/network/interfaces**, e altere os seguintes parâmetros:

```
auto eth1
iface eth1 inet manual
    pre-up ifconfig $IFACE up
    post-down ifconfig $IFACE down
```

9) É preciso ativar o serviço do IDS suricata durante a inicialização do sistema. Para isso, edite o arquivo **/etc/default/suricata** e altere os seguintes parâmetros:

```
RUN=yes
IFACE=eth1
```

10) Devido a um problema nessa versão do Suricata, um passo adicional precisará ser realizado:

```
sed -i '$i /etc/init.d/suricata restart' /etc/rc.local
```

11) Finalizada essa configuração inicial, podemos iniciar a interface de captura e iniciar o daemon do Suricata para que ele monitore o tráfego de rede. Para isso, execute:

```
ifup eth1
```

```
/etc/init.d/suricata restart
```

4.5.2 Ativando o espelhamento de tráfego

Para que o sistema IDS possa monitorar o tráfego de rede que está passando na organização, é necessário que esse tráfego seja “espelhado” para porta do Suricata. Essa configuração não seria necessária se o IDS fosse implantado no modo “inline” (onde todo o tráfego da organização é forçado a passar pelo IDS), o que não o caso deste laboratório. Dessa forma vamos utilizar a aplicação SDN-IPS que vem sendo utilizada até aqui. Para que esse passo tenha sucesso você precisará identificar três informações: i) o datapath-id do switch em que o espelhamento será realizado (SW2 na Figura do laboratório); ii) a porta na qual o servidor IDS está conectado ao switch SW2 (posicione o mouse sobre o servidor IDS na interface do Ofelia OCF); e iii) o conjunto de regras que representam o fluxo que deseja-se espelhar (nesse laboratório todo o tráfego será espelhado). De posse destas informações, o espelhamento é então configurado.

1) Esta prática pressupõe que o roteiro anterior do capítulo tenha sido executado com sucesso. Portanto, caso tenha desligado o ambiente ao final da prática anterior, é necessário religar o controlador Ryu. Ele irá recarregar as configurações que foram realizadas anteriormente a partir do arquivo **`sdn-ips-config.json`** na mesma pasta da aplicação.

2) Antes de prosseguir será necessário identificar o DatapathID em que o Controlador está instalado, pois é desse switch que o tráfego será espelhado (origem do espelhamento). Para isso busque na topologia do OCF (*Physical topology*) o switch equivalente ao SW2 do desenho da prática. Um exemplo pode ser observado na Figura 4.3:

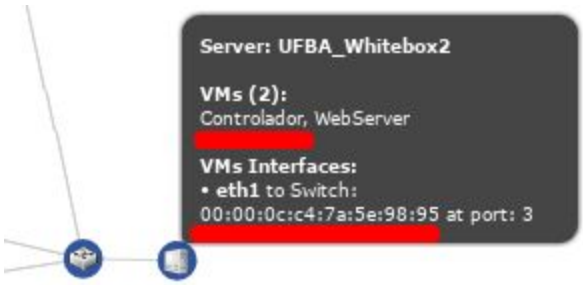


Figura 4.3 Exemplo de identificação do DatapathID do Controlador.

Na Figura 4.3 é possível observar que o servidor Controlador (no exemplo, alocado no Virt. Aggregate “UFBA_Whitebox2”) está conectado ao Switch Openflow **00:00:0c:c4:7a:5e:98:95**.

3) É necessário também identificar o “destino” do espelhamento através do DatapathID e da porta para onde o tráfego será espelhado, ou seja, onde o servidor IDS está conectado. Para isso busque na topologia do OCF (*Physical topology*) o switch equivalente ao SW4 do desenho da prática. Um exemplo pode ser observado na Figura 4.4:

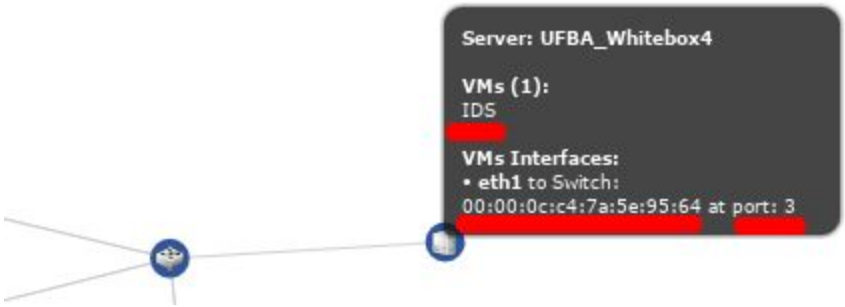


Figura 4.3 Exemplo de identificação do DatapathID do destino do espelhamento .

Na figura acima é possível observar que o servidor IDS (no exemplo, alocado no Virt. Aggregate “UFBA_Whitebox4”) está conectado ao Switch Openflow **00:00:0c:c4:7a:5e:95:64** e na **porta 3**.

4) Em seguida, na máquina Controlador, precisamos definir em qual tráfego esse espelhamento será aplicado. Em ambientes convencionais a configuração de qual tráfego será monitorado pode se dá baseado na VLAN, nos cabeçalhos TCP/IP, etc. Na aplicação SDN aqui utilizada, a definição sobre qual tráfego será monitorado se dá através dos fluxos instalados no switch, ou seja, você precisará dizer quais fluxos deseja espelhar o tráfego. Para listar os fluxos, utilize o seguinte comando substituindo o <DPID> (DatapathID) pelo valor obtido no passo anterior (2):

```
curl -s http://localhost:8080/sdnips/flows/<DPID> | python -m json.tool
```

O comando acima deve gerar como saída quatro fluxos: existem dois e-Line e para cada e-Line há dois fluxos, um para cada sentido. Como pode ser observado abaixo:

```
root@Controlador:~/fibre-2a-opencall-sdn-ips# curl -s http://localhost:8080/sdnips/flows/
00000cc47a5e9895 | python -m json.tool[
    {
        "match": {
            "dl_vlan": 3652,
            "in_port": 3
        },
        "priority": 65533
    },
    {
        "match": {
            "dl_vlan": 3652,
            "in_port": 4
        },
        "priority": 65533
    },
    {
        "match": {
            "dl_vlan": 3653,
            "in_port": 3
        },
        "priority": 65533
    },
    {
        "match": {
            "dl_vlan": 3653,
            "in_port": 1
        },
        "priority": 65533
    }
]
```

A partir do comando acima o administrador poderá selecionar qual fluxo monitorar. Neste laboratório vamos monitorar todo o tráfego, tanto da rede interna quanto da rede externa. Assim podemos usar a palavra chave “all” na escolha dos fluxos do espelhamento, conforme detalhado a seguir.

5) Agora, de fato, o espelhamento será criado a partir dos dados obtidos nos passos anteriores: DPID, porta e lista de fluxos (ou “all” para todos os fluxos). Assim execute o seguinte comando (não esqueça de alterar o <DPID-ORIG> (passo 2) e a <PORTA> e <DPID-DEST> (passo 3)):

```
curl -s -X POST -d '{"flows": "all", "to_target": "<DPID-DEST>:<PORTA>"}'
http://localhost:8080/sdnips/flows/<DPID-ORIG>/mirror | python -m json.tool
```

Exemplo:

```
root@Controlador:~# curl -s -X POST -d '{"flows": "all", "to_target": "00000cc47a5e9564:3"}'  http://localhost:8080/sdnips
/flows/00000cc47a5e9895/mirror | python -m json.tool
[
    "Success"
]
```

6) Para validar se o espelhamento está funcionando corretamente, será executado um teste de PING entre a máquina Controlador e a máquina Cliente, ao passo que a máquina IDS estará com sniffer de tráfego ligado para visualizar as mensagens. Para isso, através do console da máquina IDS, execute (o filtro mostra apenas pacotes icmp ou arp):

```
su
tcpdump -i eth1 -n icmp or arp
```

Agora, no console da máquina Controlador, execute:

```
ping -c 2 192.168.100.10
```

De volta ao console da máquina IDS, você deverá ver o seguinte:

```
root@IDS:~# tcpdump -i eth1 -n icmp or arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
00:16:20.242552 ARP, Request who-has 192.168.100.10 tell 192.168.100.254, length 42
00:16:20.242765 ARP, Reply 192.168.100.10 is-at 00:00:90:00:00:5f, length 42
00:16:20.242989 IP 192.168.100.254 > 192.168.100.10: ICMP echo request, id 1409, seq 1, length 64
00:16:20.243161 IP 192.168.100.10 > 192.168.100.254: ICMP echo reply, id 1409, seq 1, length 64
00:16:21.243555 IP 192.168.100.254 > 192.168.100.10: ICMP echo request, id 1409, seq 2, length 64
00:16:21.243701 IP 192.168.100.10 > 192.168.100.254: ICMP echo reply, id 1409, seq 2, length 64
```

4.5.3 Ferramentas de auditoria do IDS

Nesta prática vamos utilizar algumas ferramentas de auditoria de IDS para simular ataques na rede da organização e verificar se o sistema IDS consegue identificá-los. Aqui no laboratório serão utilizadas ferramentas como HPING, HYDRA e CURL. O leitor deve utilizar ferramentas como HPING e HYDRA com muito cuidado.

1) Na máquina Atacante, o primeiro passo é instalar essas ferramentas de auditoria. Para isso, acesse a máquina Atacante e execute os seguintes comandos:

```
su
apt-get update && apt-get install hping3 hydra
```

2) Agora na máquina WebServer vamos instalar o serviço de páginas web (servidor HTTP) para servir páginas externas e redirecionar os usuários infectados. Este servidor será utilizado como

alvo do teste de auditoria do HPING. Para isso, o primeiro passo é instalar o apache e depois criar o arquivo HTML que será servido. Utilize os seguintes comandos:

```
su
apt-get update && apt-get install apache2 tcpdump
cat >/var/www/html/index.html <<EOF
<h1>SDN-IPS: Sua maquina foi identificada como possivelmente infectada! Procure o
departamento de TI da organizacao!</h1>
EOF
```

3) Na máquina Atacante, execute um ataque de syn-flood contra a máquina WebServer:

```
hping3 --fast -S -p 80 192.168.100.200
```

4) Na máquina Atacante, realize um ataque de brute-force SSH com hydra contra WebServer:

```
seq 1 300 > /tmp/pass.txt
hydra -l root -P /tmp/pass.txt ssh://192.168.100.200
```

5) Na máquina Cliente, vamos simular um acesso a IP malicioso de C&C (servidor de *Command and Control*). Para isso, acesse o site do Tracker do Trojan Feodo (ou qualquer outro da lista da emergint-threats botcc), mantido pelo time de segurança abuse.ch através do site https://feodotracker.abuse.ch/, escolha algum IP que esteja online de C&C para simular o acesso. A partir daí vamos rotear esse IP através da rede do AS100 e simular um acesso. Para isso execute o seguinte comando (atente-se para alterar o <IP-CnC> para o IP escolhido):

```
route add -host <IP-CnC> gw 192.168.100.254
wget http://<IP-CnC>/
```

OBS: apesar do comando acima não conseguir efetuar a conexão HTTP, afinal na rede deste laboratório não habilitamos o acesso à Internet, essa tentativa de acesso já é suficiente para identificar uma máquina possivelmente comprometida no IDS.

6) Por fim, na máquina IDS, observe os logs do Suricata (/var/log/suricata/fast.log) se as tentativas de intrusão foram corretamente identificadas:

```
tail /var/log/suricata/fast.log
```

A saída esperada é algo como:

```
root@IDS:~# tail /var/log/suricata/fast.log
 • • • •
11/30/2017-00:18:56.131529  [**] [1:10001:1] Possible TCP Syn Flood DoS [**] [Classification: Attempted Denial of Service]
 [Priority: 2] {TCP} 192.168.66.1:1288 -> 192.168.100.200:80
11/30/2017-00:19:10.511084  [**] [1:2001219:20] ET SCAN Potential SSH Scan [**] [Classification: Attempted Information Lea
k] [Priority: 2] {TCP} 192.168.66.1:34339 -> 192.168.100.200:22
11/30/2017-00:19:44.865637  [**] [1:2404546:4825] ET CNC Ransomware Tracker Reported CnC Server group 147 [**] [Classifica
tion: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.100.10:33884 -> 95.85.19.195:80
```

Assim, no final deste experimento, observa-se que os ataques já estão sendo detectados pelo IDS. No entanto, nenhuma medida de prevenção está sendo tomada. Desta forma, na próxima seção serão criadas as medidas de prevenção de acordo com o tipo de ataque (interno ou externo).

5 AÇÕES DE CONTENÇÃO DE INTRUSOS COM SDN

Neste capítulo serão estudados os aspectos de contenção de intrusos através de mecanismos SDN. Nas próximas seções o leitor poderá entender as diferenças entre IDS e IPS, tipos de contenção que podem ser empregadas e como SDN poderá apoiar nessas ações.

5.1 Sistemas de Prevenção de Intrusos

No capítulo anterior foi possível ter uma visão geral sobre o funcionamento dos Sistemas de Detecção de Intrusão, cujo objetivo resume-se a monitorar a rede ou o host e identificar comportamentos maliciosos - tipicamente ataques. É possível fazer uma analogia do IDS com um sistema de alarme de um carro, que apenas soa uma sirene quando alguém abre o carro sem autorização. Neste capítulo, será abordado as ferramentas de IPS (Sistema de Prevenção de Intrusão) que tem uma ação mais incisiva ao receber os alertas do IDS. O IPS complementa, portanto, o funcionamento do IDS, uma vez que ele bloqueia a intrusão e impede que um dano maior seja causado à rede. Utilizando a analogia do carro, é como se o IPS, além de disparar o alarme, também trave as rodas para evitar que o invasor leve o carro. É possível pensar sobre o IDS como um sistema passivo (apenas detecta) e no IPS como um sistema ativo (detecta e atua sobre o evento).

Um exemplo de funcionamento do IPS é ilustrado na Figura 5.1. Nessa figura é importante notar que a rede da organização conta com um sistema de Firewall para filtragem dos pacotes e também uma solução de IPS. Ao receber requisições HTTP (porta 80/TCP), por exemplo, o Firewall da organização permite que o tráfego seja encaminhado, porém agora cabe ao sistema IPS analisar esse tráfego e, eventualmente, na presença de requisições maliciosas, tomar ações de prevenção. Além disso, na arquitetura apresentada na Figura 5.1, a comunicação entre diferentes segmentos de rede da organização também é inspecionada pelo IPS, prevenindo a propagação de atividade maliciosa na rede interna.

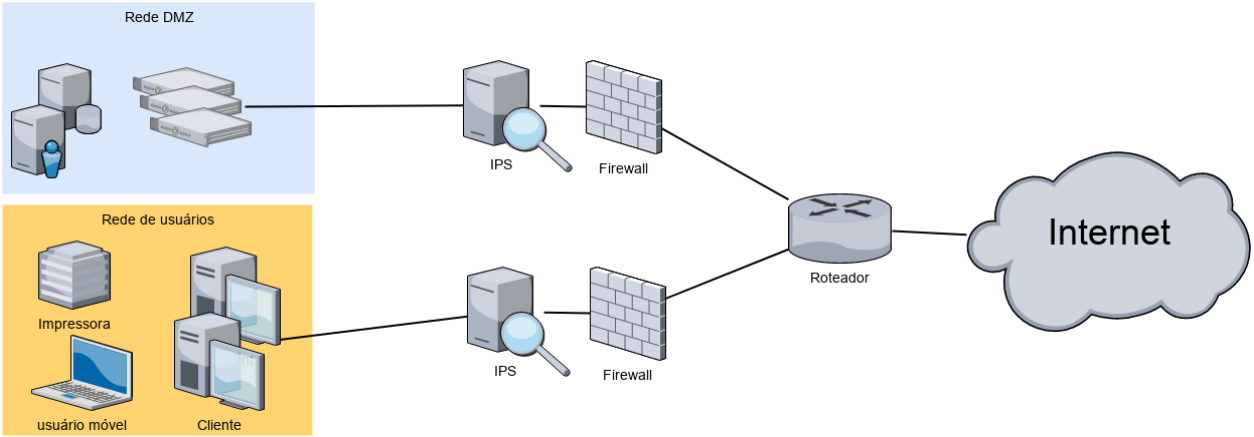


Figura 5.1 - Arquitetura com IPS inline na proteção da rede

É comum que a solução de detecção e prevenção de intrusos seja implantada em um mesmo sistema, levando o nome simplesmente de IPS. Em qualquer caso, esses sistemas são uma barreira importante de segurança para prevenir invasões na organização. Não obstante, dependendo do modelo de implantação que for adotado, alguns pontos de atenção devem ser analisados: 1) como esse sistema tem a capacidade de bloquear ataques, a própria segurança do IPS deve ser cuidadosamente avaliada para evitar comprometimento; 2) deve-se ter muito cuidado com falsos positivos e falsos negativos; 3) deve-se atentar para o desempenho do sistema, principalmente quando “inline”, para não impactar na qualidade da rede da organização; 4) as ações de contenção devem ser comunicadas dentro da organização.

A coleta de evidências constitui-se um importante requisito para auditorias futuras acerca das ações de mitigação de uma invasão. Esse processo é realizado através do registro nas trilhas de auditoria (logs) dos eventos de detecção e prevenção. Os logs geralmente são armazenados em formato SYSLOG, porém podem ser armazenados também em formato JSON ou até mesmo em bancos de dados SQL. Dessa maneira, o administrador da rede poderá não apenas tomar ações quanto ao ataque, mas também estará apto a responder questionamentos futuros quanto à ação executada. Por outro lado, essas evidências devem ser coletadas e armazenadas tendo em vista o respeito à privacidade do usuário.

5.2 Contenção de intrusos

A partir dos alarmes gerados na fase de detecção, o IPS executa ações de contenção para interromper o ataque e evitar maiores danos. Essas ações podem ser das mais variadas naturezas:

- Cancelamento da conexão em andamento (ex: envio de pacotes TCP RST);
- Bloqueio do host atacante através da configuração de regras de Firewall;
- Limitação de banda e requisições do atacante (*rate-limit*);
- Redirecionamento de tráfego para VLAN de quarentena para máquinas internas comprometidas;
- Redirecionamento do tráfego para sistemas de "*honeypot*" para estudar o ataque;
- Limpeza do tráfego removendo partes maliciosas do fluxo de dados;
- Dentre tantas outras possibilidades.

Em verdade, o conjunto de ações suportadas pelo IPS depende diretamente das tecnologias utilizadas (ex: Firewall, ferramentas de NAC - Network Access Control, scripts de bloqueio, etc) e do modelo de implantação adotado (ex: inline versus espelhado). No modelo de implantação inline, geralmente o próprio IPS é responsável pela execução das ações de contenção, portanto há uma flexibilidade maior pela adoção de tecnologias no próprio IPS para realizar os bloqueios. Por outro lado, no modelo de IPS espelhado o sistema depende da interação com outros elementos de rede na tomada da decisão.

Um exemplo desse cenário é ilustrado na Figura 5.2. Nessa figura uma máquina comprometida (ex: usuário clicou em um anexo infectado) realiza acesso a um IP malicioso (1) de servidor de Comando e Controle (servidor malicioso que é usado para controlar máquinas infectadas remotamente como se fossem zumbis). Em seguida, o sistema IPS espelhado identifica aquele tráfego anômalo (2) e notifica o Firewall (3) para redirecionar aquele tráfego ao servidor de quarentena. A partir daí, ao realizar qualquer outro tipo de acesso, a máquina ficará restrita ao ambiente de quarentena (4) até que seja efetuada uma análise com antivírus e antimalware para limpar a máquina.

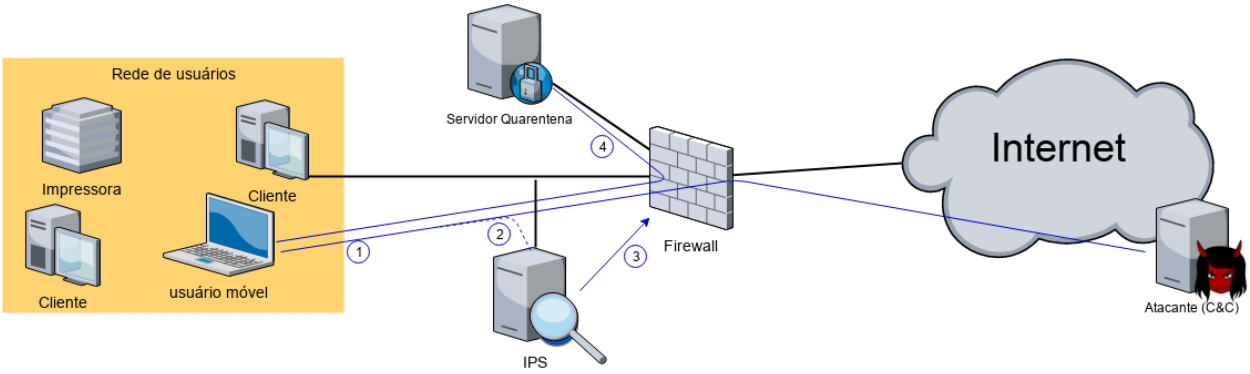


Figura 5.2 - Ilustração de sistema de IPS espelhado com contenção via quarentena.

A fim de suportar redirecionamento de tráfego para uma VLAN de quarentena, a solução de IPS deveria ser capaz de realizar modificações na camada de enlace do pacote ou aplicar alguma técnica de roteamento baseado em política. Em casos de IPS em modo espelhado, esses desafios são ainda maiores ao considerar que o elemento de rede que irá de fato realizar a contenção precisa ter algum suporte a essas tecnologias (ex: *MAC based VLAN*, *Policy Based Routing*, etc), além de disponibilizar um mecanismo para configuração remota (ex: SNMP, SSH, NETCONF, API REST, etc).

5.3 Uso de SDN para execução de ações de contenção

O uso de SDN e Openflow nesse contexto de contenção de ataques do IPS pode potencializar o conjunto de contramedidas adotadas, principalmente através da flexibilidade e programabilidade que são incorporadas. Além disso, Openflow fornece uma API padronizada para comunicação com os switches, o que pode impulsionar estratégias que visam bloquear os ataques mais próximos de sua origem na rede. Diversas aplicações SDN fornecem uma API para integração de sistemas, geralmente através de REST. As APIs de comunicação com switches, também conhecida como API sul, e API de comunicação com aplicações SDN, conhecida como API norte, pode ser visualizada conforme ilustrado na Figura 5.3.

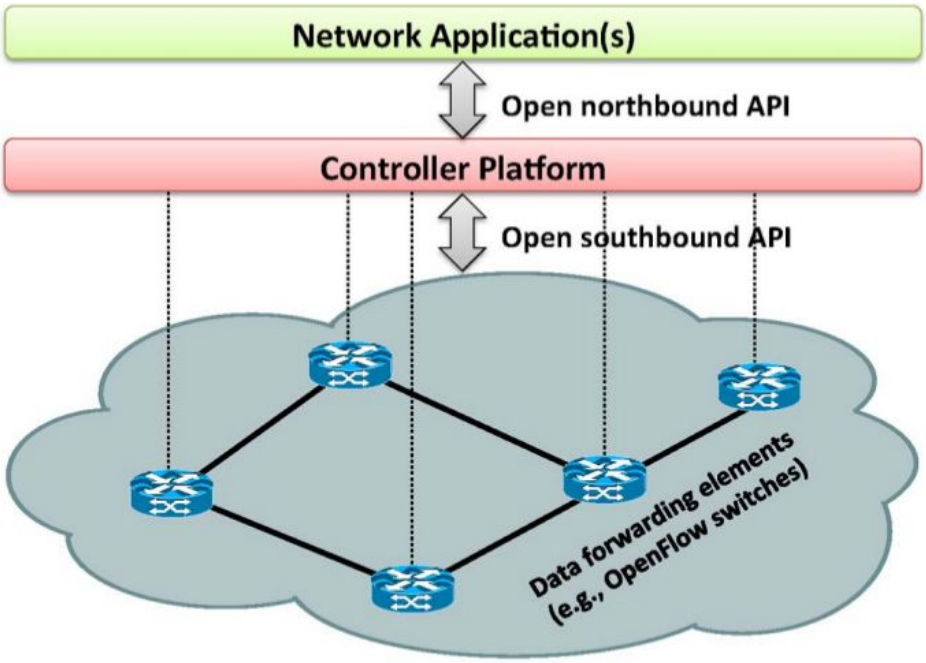


Figura 5.3 - Arquitetura simplificada de SDN e APIs norte e sul [Kreutz et al., 2015]

O protocolo Openflow define um conjunto de ações que podem ser tomadas para determinado fluxo de pacotes. Essas ações dependem da versão do protocolo suportada pelo switch e dependem do fabricante ou tipo de switch Openflow em questão (muitas ações são consideradas opcionais na especificação do protocolo). A versão 1.0 do protocolo Openflow, versão disponível no FIBRE e mais amplamente usada nas organizações, permite executar ações em relação à interfaces físicas, filas e até mesmo protocolos da camada de transporte (L4). É importante notar que ações em protocolos como Ethernet, IPv4, TCP e UDP são opcionais, o que limita de alguma maneira a integração de SDN com IPS. O Quadro 5.1 apresenta um resumo das principais ações do Openflow 1.0.

Quadro 5.1. Principais ações do Openflow 1.0 e comandos do Open vSwitch

Protocolo/Tipo	Comportamento	Alvo
<vazio>	Descarte	Descartar o pacote
Port	Encaminha	(output:port), onde *port* pode ser: **PORT-ID**: encaminha para porta específica **ALL**: encaminha para todas as portas exceto porta de origem

		INGRESS: envia para porta de entrada **CONTROLLER**: envia para o controlador **TABLE**: re-injeta o pacote para processamento novamente **LOCAL**: envia para o S.O. **FLOOD**: envia para todas as portas exceto a porta de entrada **NORMAL**: reprocessa o pacote na pilha de rede convencional
Queue	Modifica	Queue-ID - modifica o pacote de fila (set_queue:queue-id)
Ethernet	Modifica	MAC de origem (mod_dl_src:mac) MAC de destino (mod_dl_dst:mac) VLAN ID (mod_vlan_vid:vlan_vid) Prioridade VLAN (mod_vlan_pcp:vlan_pcp)
	Strip	VLAN ID - Remove a tag de vlan (strip_vlan)
IPv4	Modifica	IP de origem (mod_nw_src:ip) IP de destino (mod_nw_dst:ip) Tipo de serviço ToS (mod_nw_tos:tos)
TCP/UDP	Modifica	Porta de origem (mod_tp_src:port) Porta de destino (mod_tp_dst:port)

A partir desse conjunto de ações é possível implantar diferentes mecanismos de contenção no Sistema de Prevenção de Intrusos, alguns exemplos são:

- Bloqueio de host: para realizar o bloqueio de um host através de regra Openflow, a aplicação pode enviar uma mensagem de *FlowMod* (modificação de fluxo) com o parâmetro *actions* vazio;
- VLAN de quarentena: para realizar a contenção de um host por meio de um isolamento de quarentena via regra Openflow, a aplicação pode enviar uma mensagem de *FlowMod* (modificação e fluxo) com o parâmetro *actions* contendo o comando mod_vlan_vid:vlan_vid;
- Redirecionamento de tráfego: para realizar o redirecionamento de um host via regra Openflow, a aplicação pode enviar uma mensagem de *FlowMod* (modificação e fluxo)

com o parâmetro *actions* contendo o comando mod_nw_dst:ip (redirecionamento em em camada 3) ou mod_dl_dst:mac (para redirecionamento em camada 2).

De toda maneira, a execução de ações de contenção através de regras Openflow apresenta importantes desafios de implantação, a saber: como a tabela de fluxos não armazena estado, o controlador SDN deve dinamicamente mapear esses estados na sua tabela; a tomada de ações através do envio de pacotes de resposta (ex: TCP RST) depende de Packet Out pelo controlador, o que pode degradar o desempenho; Openflow 1.0 tem um conjunto restrito de campos de casamento de ações, o que limita, por exemplo a aplicação de técnicas de rate-limit; dentre outros.

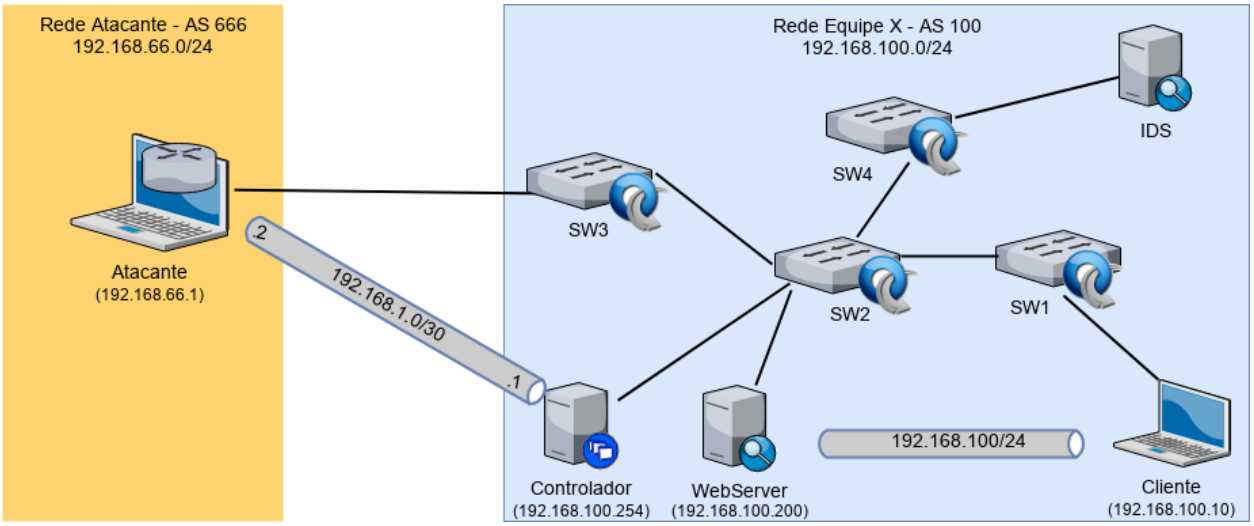
O roteiro de prática deste capítulo apresenta dois exemplos para realizar a contenção de hosts.

5.4 Exercícios de Fixação

1. Quais os tipos de ação de prevenção que podem ser adotadas pelo IPS?
2. Considerando um sistema IDS espelhado, quais os desafios para torná-lo um IPS?
3. Como o Controlador SDN pode implantar uma técnica de contenção baseada em quarentena?
4. Como o Controlador SDN pode implantar uma técnica de contenção baseada em cancelamento da conexão TCP?

5.5 Roteiro de laboratório

Neste laboratório será realizado a configuração de ações de contenção através do controlador SDN a partir dos alertas que são identificados pelo IDS. As ações de contenção ocorrerão de forma diferenciada para hosts intrusos internos e externos: para hosts externos a contenção se dará pelo simples bloqueio, ao passo que para hosts internos a contenção será realizada através de quarentena do host. A topologia utilizada neste laboratório é a mesma anterior, conforme Figura 5.4.



5.4. Topologia proposta para os experimentos da Oficina.

5.5.1 Configurando o script de contenção no IDS

Nesta prática o objetivo é instalar e configurar o Guardian, uma ferramenta que simplesmente monitora os eventos de alertas do IDS e executa ações configuradas, para conter automaticamente os hosts relacionados com atividades maliciosas na rede.

Esta prática pressupõe que o roteiro anterior do capítulo tenha sido executado com sucesso. Portanto, caso tenha desligado o ambiente ao final da prática anterior, é necessário religar o controlador Ryu. Ele irá recarregar as configurações que foram realizadas anteriormente a partir do arquivo **`sdn-ips-config.json`** na mesma pasta da aplicação.

1) Na máquina IDS, baixe o “Guardian Active Response” conforme abaixo:

```
su
wget https://goo.gl/ansjP8 -O guardian-1.7.tar.gz
tar -xzf guardian-1.7.tar.gz
cd guardian-1.7/
```

2) Copie o arquivo de configuração do Guardian para o diretório /etc e edite o arquivo, ajustando os parâmetros AlertFile e RemoteController para indicar o arquivo de alertas do Suricata e o IP da máquina Controlador (<IP-CONTROLADOR>, conforme saída do comando

“ifconfig eth0” no Controlador ou é possível ver também via interface do OCF), respectivamente:

```
cp guardian.conf /etc/
sed -i 's@/var/adm/secure@/var/log/suricata/fast.log@g' /etc/guardian.conf
sed -i 's@X.X.X.X@<IP-CONTROLADOR>@g' /etc/guardian.conf
```

3) Copie o arquivo guardian.pl para o diretório /usr/local/bin e copie os scripts de bloqueio e desbloqueio referentes à aplicação SDN-IPS para o mesmo diretório:

```
cp guardian.pl /usr/local/bin/
cp scripts/sdnips_block.sh /usr/local/bin/guardian_block.sh
cp scripts/sdnips_unblock.sh /usr/local/bin/guardian_unblock.sh
```

4) Antes de iniciar o Guardian vamos zerar as notificações do Suricata para evitar o bloqueio ocasionado pelo teste da seção anterior e zerar os logs do Guardian:

```
echo > /var/log/suricata/fast.log
echo > /var/log/guardian.log
/etc/init.d/suricata restart
```

5) Execute o Guardian com o seguinte comando:

```
/usr/local/bin/guardian.pl -c /etc/guardian.conf
```

5.5.2 Teste com bloqueio de host externo

Nesta prática vamos realizar um ataque a partir da máquina Atacante contra a máquina WebServer e observar que o IDS irá detectar o ataque e requisitar o bloqueio ao Controlador.

1) Na máquina Atacante, vamos realizar um teste de brute-force SSH com hydra:

```
su
hping3 --fast -S -p 80 192.168.100.200
```

2) Na máquina IDS, observe os logs do Guardian que o ataque foi identificado e bloqueado:

```
tail /var/log/guardian.log
```

Deverá ser exibida uma mensagem como mostrado abaixo:

```
root@IDS:~/guardian-1.7# tail /var/log/guardian.log

Guardian process id 565
Thu Nov 30 00:53:10 2017: 192.168.66.1  [1:10001:1] Possible TCP Syn Flood DoS
Running '/usr/local/bin/guardian_block.sh 192.168.66.1 eth0'
```

3) Observe no console do Controlador os logs da aplicação SDN-IPS que o host foi bloqueado:

```
(1482) accepted ('10.144.12.56', 60956)
==> contention_block ipaddr=192.168.66.1 in all switches
10.144.12.56 - - [30/Nov/2017 01:15:32] "POST /sdnips/contention/block HTTP/1.1" 200 119 0.008405
```

4) Finalmente, de volta à máquina Atacante observe que não é possível mais realizar acesso ao servidor WebServer:

```
wget http://192.168.100.200
```

A saída esperada é erro de Timeout, uma vez que a máquina foi bloqueada:

```
root@Atacante:~# wget -T 3 http://192.168.100.200
converted 'http://192.168.100.200' (ANSI_X3.4-1968) -> 'http://192.168.100.200' (UTF-8)
--2017-11-30 01:02:19--  http://192.168.100.200/
Connecting to 192.168.100.200:80... failed: Connection timed out.
Retrying.

--2017-11-30 01:02:23--  (try: 2)  http://192.168.100.200/
Connecting to 192.168.100.200:80... failed: Connection timed out.
Retrying.
```

5.5.3 Teste com quarentena de host interno

Nesta prática vamos realizar um acesso malicioso a partir da máquina Cliente contra um IP de C&C e observar que o IDS irá detectar este comportamento malicioso e requisitar a quarentena ao Controlador.

1) Na máquina Cliente, vamos simular um acesso a IP malicioso de C&C (servidor de *Command and Control*). Para isso, acesse o site do Tracker do Trojan Feodo (ou qualquer

outro da lista da emergint-threats botcc), mantido pelo time de segurança abuse.ch através do site https://feodotracker.abuse.ch/, escolha algum IP que esteja online de C&C para simular o acesso. A partir daí vamos rotear esse IP através da rede do AS100 e simular um acesso. Para isso execute o seguinte comando (atente-se para alterar o <IP-CnC> para o IP escolhido):

```
route add -host <IP-CnC> gw 192.168.100.254
wget -q -O - http://<IP-CnC>/
```

Observe que a requisição será redirecionada automaticamente para o host WebServer e você receberá como retorno a mensagem:

```
root@Cliente:~# wget -q -O - http://95.85.19.195/
<h1>SDN-IPS: Sua maquina foi identificada como possivelmente infectada! Procure o departamento de TI da organizacao!</h1>
root@Cliente:~#
```

3) Observe no console do Controlador os logs da aplicação SDN-IPS que o host foi bloqueado:

```
(1414) accepted ('10.144.12.56', 60955)
10.144.12.56 - - [30/Nov/2017 01:05:16] "POST /sdnips/contention/quarantine HTTP/1.1" 200 119 0.008196
==> create contention_quarantine_redirect in dpid=00000cc47a5e9894 src=192.168.100.10 dst=95.85.19.195 redirect_to=192.168
.100.200
```

Ao final deste experimento, o aluno deve ter sido capaz de criar regras para contenção dos ataques e verificar a atuação dessas regras aplicadas a diferentes contextos (interno e externo à organização).

6 CONCLUSÃO

O crescimento na quantidade e complexidade dos ataques contra a infraestrutura das organizações torna evidente a necessidade de ferramentas e metodologias que visam identificar e conter atividade maliciosa com requisitos de tempo e precisão bastante restritos. A utilização de ferramentas como Sistema de Detecção e Prevenção de Intrusão mostra-se uma abordagem imprescindível para complementar as barreiras de proteção da organização, incorporando princípios de defesa em profundidade e diversidade de defesa. Sobretudo impulsionado pela programabilidade e flexibilidade da rede que foram viabilizadas com o paradigma SDN, a contenção de intrusos internos e externos ajuda a minimizar tais riscos e, em última instância, garantir as propriedades de segurança aos usuários (disponibilidade, confidencialidade, autenticidade e integridade).

Neste laboratório, os alunos tiveram oportunidade de construir um ambiente completo para monitoramento, teste e contenção de ataques cibernéticos, através da configuração de múltiplos sistemas autônomos e diferentes serviços de rede. Foi possível entender e testar diversos componentes de uma arquitetura de Sistema de Prevenção de Intrusão baseado em SDN/Openflow, incluindo:

- Configuração de enlaces L2 para comunicação na rede, através do padrão e-Line do Metro-Ethernet Forum;
- Configuração de roteamento BGP via software router convencional (quagga) e via SDN (controlador Ryu)
- Configuração de Sistema de Detecção de Intrusos (Suricata IDS)
- Configuração de mecanismos e contenção de intrusos integrando SDN e IDS

7 RESUMO E LIÇÕES APRENDIDAS

Cap 2: Visão geral sobre SDN/OpenFlow e testbed FIBRE

- O paradigma SDN permite maior flexibilidade e programabilidade na rede, viabilizando customizações significativas no seu modo de funcionamento, nos algoritmos e nas funcionalidades providas;
- Uma API aberta, como OpenFlow, para controle remoto dos comutadores facilita o desenvolvimento de aplicações de rede que tiram proveito do controle logicamente centralizado para aplicar políticas ou estratégias que propiciem maior ganho global;
- Infraestruturas de experimentação *testbed* como o FIBRE são essenciais para validar aplicações antes de disponibilizá-las em produção;
- O *testbed* FIBRE possui diversos recursos para experimentação, através do framework OCF, permitindo experimentação com máquinas virtuais, switches OpenFlow, etc;
- Utilizando a infraestrutura do FIBRE é possível desenvolver aplicações SDN, por exemplo, através do controlador Ryu em Python e implantar serviços de *Carrier Ethernet* com túneis virtuais ethernet ponto-a-ponto definidos através do padrão e-Line do MEF (Metro-Ethernet Forum);

Cap 3: Roteamento Inter-AS via BGP e SDN/OpenFlow

- Foi possível obter uma visão geral sobre roteamento entre Sistemas Autônomos na Internet através do protocolo BGP;
- O uso de SDN nesse contexto abre novas perspectivas, tanto do ponto de vista de escalabilidade e possibilidade de execução desses protocolos em hardware de propósito geral, quanto do ponto de vista de aplicação de novas políticas de roteamento para definição dos melhores caminhos;
- Foi possível desenvolver e implantar uma solução de roteamento BGP no controlador SDN Ryu, instalar e configurar uma solução de roteamento BGP convencional e viabilizar a comunicação de ambas;
- Os roteiros de laboratório forneceram uma visão geral sobre as atividades de administração de um AS, estabelecimento de sessões BGP e teste de conectividade.

Cap 4: Sistemas de Detecção de Intrusão

- Os sistemas de detecção de intrusão fornecem uma barreira adicional na proteção em profundidade das organizações, seja pela análise de padrões de tráfego ou assinaturas de ataques, o IDS ajuda o administrador a identificar acessos maliciosos na rede tanto externamente quanto internamente;
- Foi possível estudar os diversos tipos de IDS e sua forma de funcionamento, bem como soluções de mercado e *open source* que podem ser utilizadas;

- O gerenciamento de assinaturas do IDS é um ponto de muita atenção na adoção dessas ferramentas, pois ela tem relação direta com a efetividade da solução, com a privacidade dos usuários, ocorrência de falsos positivos e falsos negativos e com o próprio desempenho da rede;
- Através dos roteiros de laboratório, o aluno teve uma experiência prática de instalação e configuração de um IDS baseado no software Suricata;
- As ferramentas de IDS geralmente disponibilizam a possibilidade de criação de regras customizadas, o que permite ao administrador escrever assinaturas para ataques ou comportamentos indesejados específicos da organização. Nos laboratórios, os alunos tiveram oportunidade de escrever regra específica para ataque de negação de serviço, levando em consideração o perfil de tráfego da organização;
- Ferramentas de auditoria de IDS foram utilizadas para validar o funcionamento e efetividade da ferramenta.

Cap 5: Ações de Contenção de Intrusos com SDN

- Ao identificar um ataque através do IDS o administrador pode desejar aplicar alguma ação de contenção sobre o tráfego malicioso: bloqueio, restrição de banda, redirecionamento, etc. Neste capítulo o leitor estudou algumas das estratégias de implantação de ações de contenção;
- Sistemas de Prevenção de Intrusos (IPS) atuam de forma ativa na rede, utilizando como entrada os alertas do IDS e aplicando ações que impedem ou limitam o escopo das atividades maliciosas na rede;
- A utilização de SDN/OpenFlow potencializa o conjunto de ações que podem ser tomadas face à identificação de atividade maliciosa, além de prover uma API única para configuração remota dos equipamentos de rede;
- Por outro lado, as características do protocolo OpenFlow e as limitações dos comutadores impõem desafios ao desenvolvedor de aplicações SDN sobre as metodologias de bloqueio que serão aplicadas. Neste capítulo, o aluno teve oportunidade de estudar três possíveis ações que podem ser aplicadas;
- Através dos roteiros de laboratório foi possível estudar e testar na prática as ações de contenção do IPS integrado com o controlador SDN.

AUTORES

Italo Valcy S Brito é mestre em Ciência da Computação pela UFBA, onde exerce o cargo de Gestor de Segurança da Informação e Coordenador da Equipe de Segurança de Redes. É co-fundador e membro do CERT.Bahia (Grupo de Segurança da Rede Acadêmica da Bahia), pesquisador no Ponto de Presença da RNP e Pesquisa na Bahia e professor das trilhas de redes, segurança e sistemas na ESR. Suas áreas de interesse são: Análise Forense, Web Pentest, Roteamento e Infraestrutura Segura, Phishing, Monitoramento de Atividade Maliciosa e SDN. CV Lattes: http://lattes.cnpq.br/8911162910001065

Adriana Viriato Ribeiro é mestre em Ciência da Computação pela UFBA com ênfase em Redes de Computadores Centradas no Conteúdo, Bacharel em Sistemas de Informação pela Universidade Estadual do Sudoeste da Bahia (2014). Experiência na área de Ensino e Pesquisa iniciada na graduação com as práticas de monitoria e Iniciação Científica e estendidas durante o Mestrado com a atividade de Estágio Docente e participação em projetos de pesquisa. Membro do grupo de pesquisa INSERT (Infraestrutura e Sistemas para Redes e Telecom) sob orientação do professor Leobino Sampaio, onde desenvolve pesquisa nas área de Redes de Computadores e Internet do Futuro. Trabalha no Núcleo de Produção de Dados do Centro de Integração de Dados e Conhecimentos para Saúde (CIDACS) desenvolvendo algoritmos para atividades de ETL em grandes bases. CV Lattes: http://lattes.cnpq.br/1582538985588009

Leobino Nascimento Sampaio é Doutor em Ciência da Computação pela Universidade Federal de Pernambuco (2011), Mestre em Redes de Computadores pela Universidade Salvador - UNIFACS (2004), graduado em Ciência da Computação Com Ênfase em Análise de Sistemas pela UNIFACS (1996) e Administração de Empresas pela Universidade Católica do Salvador (1999). Com mais de 15 anos de experiência em pesquisa e ensino em cursos de graduação, atualmente é professor Adjunto IV da UFBA, membro permanente do programa de Pós-graduação em Ciência da Computação (PGCOMP) do Instituto de Matemática da UFBA e líder do grupo de pesquisa INSERT. É membro do Comitê Técnico em Monitoramento de Redes (CT-Mon) da RNP, membro da Sociedade Brasileira de Computação (SBC) e membro da Câmara Básica de Assessoramento e Avaliação Científico-Tecnológica de Inovação da Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB). Atua nas áreas de Redes de Computadores e Internet do Futuro. CV Lattes: http://lattes.cnpq.br/1952937182023132

AGRADECIMENTOS

Agradecemos à Rede Nacional de Ensino e Pesquisa pelo financiamento da produção deste material através do “Edital 2a Open call FIBRE - Desenvolvimento de material educacional”.

REFERÊNCIAS

Hawkinson, John, and Tony Bates. "Guidelines for creation, selection, and registration of an Autonomous System (AS)." (1996).

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

ONF. Openflow Switch Specification v1.3.0. jun. 2012. ONF Specification.

ONF. Software-Defined Networking: The New Norm for Networks. abr. 2012. White Paper.

Mckeown, N. et al. Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, ACM, v. 38, n. 2, p. 69–74, 2008.

Mckeown, N. Software-defined Networking. abr. 2009. Infocom Keynote Talk.

Stuart, D., & Beaver, K. (2013). Next Generation IPS for Dummies. Hoboken: John Wiley & Sons, Inc.

Mahurin, M. Basic NGIPS Operation and Management for Intrusion Analysts. Aug 2017. SANS InfoSec Reading Room Whitepaper