Definição de Novas Regras para o IDS Snort em Redes Definidas por Software

Thiago Oliveira Garcia[1], Charles V. Neu[1]

[1]Universidade de Santa Cruz do Sul (UNISC)

thiago.garcia@gaz.com.br,charles1@unisc.br

Abstract. *In this work, we intend to develop new rules for identifying intrusion attacks that can occur in an SDN network through intrusion detection tools that help identify these vulnerabilities, we use the Open Source SNORT system for attack detection, the Mininet emulator to emulate the SDN environment together with the Ryu driver. The results show that the integration between the Ryu controller and Snort was successful, the rules developed in Snort were able to detect attacks against Telnet and FTP, and this same tool sent the alerts to the Ryu controller inside an SDN environment.SDN.*

Resumo. *Neste trabalho buscamos desenvolver novas regras de identificação de ataques de intrusão que podem ocorrer em uma rede SDN através de ferramentas de detecção de intrusão que auxiliem na identificação destas vulnerabilidades, utilizamos o sistema Open Source SNORT para a detecção dos ataques, o emulador Mininet para emular o ambiente SDN juntamente com o controlador Ryu. Os resultados mostram que a integração entre o controlador Ryu e o Snort foi bem sucedida, as regras desenvolvidas no Snort foram capazes de detectar ataques contra Telnet e FTP, e esta mesma ferramenta enviou os alertas ao controlador Ryu dentro de um ambiente SDN.*

1. Introdução

As redes de computadores, tanto tradicionais quanto as Redes Definidas por Software ou *Software Defined Networking* - SDN necessitam de mecanismos de segurança. No modelo tradicional de redes de computadores, a administração dos equipamentos que compõem estas redes é realizada de maneira individual. Este modelo dificulta o trabalho do administrador da rede e torna mais complicada a sua análise para a resolução de um problema ou melhoria na configuração de uma política [Bitencourt 2014]

As redes SDN apresentam uma alternativa para um gerenciamento logicamente centralizado e trazem maior flexibilidade aos administradores de rede, assim facilitando o seu dia a dia. O conceito de redes SDN é proporcionar um novo modelo para administração das redes de computadores, permitindo que um único equipamento possa ter a visibilidade e o controle de toda a rede e uma administração centralizada [Kim 2013]. A utilização deste modelo proporciona também maior flexibilização no controle das redes, onde dividir tipos de tráfego, empregar controles diferentes para cada elemento da rede e analisá-los separadamente é a questão-chave deste novo modelo [Mckeown 2008].

Ao mesmo tempo em que a arquitetura SDN apresenta-se promissora, existem alguns desafios de segurança a serem vencidos nesta nova tecnologia. A centralização do plano de controle traz inúmeras vantagens, como lógica centralizada e visão global da rede [Nadeau 2013]. Tais características representam significativos benefícios, porém aumentam a

exposição do controlador e suas aplicações a ataques na rede e interceptação de fluxos, assim causando um grande transtorno e prejuízo à empresa que foi afetada.

Embora a adoção desta nova tecnologia esteja em fase inicial, através deste trabalho busca-se desenvolver novas regras de identificação de ataques de intrusão que podem ocorrer em uma rede SDN totalmente centralizada, através de ferramentas de detecção de intrusão que auxiliem na identificação destas vulnerabilidades. É utilizado o sistema *Open Source* Snort e o *Dataset* Darpa 99, que contém uma vasta base de ataques com as principais características de cada intrusão. Foi desempenhada uma análise dos dados dos ataques contra os protocolos Telnet e *File Transfer Protocol* - FTP, onde foi determinada uma faixa de tempo de cada ataque e análise dos pacotes de redes baseado em informações que determinam suas características.

O restante deste trabalho está organizado da seguinte maneira: o capítulo 2 apresenta os trabalhos estudados com relação a métodos de segurança em SDN encontradas na literatura. O trabalho desenvolvido é apresentado no capítulo 3. O capítulo 4 mostra os testes realizados e os resultados obtidos. E finalmente, a conclusão e considerações finais são apresentadas no capítulo 5.

2. Trabalhos Relacionados

Mattos *et* al. [Mattos, Ferraz and Duarte 2013] sugere o isolamento da comunicação entre redes virtuais que utilizam uma mesma infraestrutura física que visa garantir a confidencialidade da rede virtual de cada host nesta estrutura física, onde são isolados os recursos de cada host e do tráfego desta rede. Todavia, garantir o isolamento entre estas redes virtuais irá prevenir somente ataques de uma rede virtual em outra rede virtual em uma mesma estrutura física, mas não foca na detecção de ocorrências dos ataques de intrusão dentro do ambiente virtual.

O trabalho desenvolvido por Nagahama [Nagahama 2013], apresenta o IPSFlow, que é uma arquitetura de Sistema de Prevenção de Intrusão que utiliza o SDN e o protocolo OpenFlow para a construção de um sistema de prevenção com ampla cobertura na rede. Nesta arquitetura IPSFlow, de acordo com o resultado da análise feita por um determinado IDS, o controlador OpenFlow teve como bloquear o fluxo de forma automática no switch que está mais próximo da origem do tráfego, impedindo assim, que o tráfego malicioso circule livremente pela rede.

No artigo de Le *et* al. [Le *et* al. 2015], propõem um deslocamento de rede dentro da abordagem de SDN realizando experiências típicas contra os ataques de intrusão, que busca a redução do custo de hardware e software comparado a uma rede tradicional, mantendo o mesmo desempenho. Já no trabalho realizado por Chen *et* al. [Chen and Chen 2015], utiliza um método que pressupõe que o atacante vai usar ferramentas de verificação para fazer um reconhecimento da rede. O autor cria dois algoritmos, o primeiro gera falsos hosts, que espera para o estabelecimento de conexão completa. O segundo algoritmo gera informações falsas, como portas abertas que irão responder ao atacante, sendo uma isca virtual para atrair atacantes em potenciais.

Tabela 1. Comparativo entre trabalhos estudados e trabalho proposto

Trabalho	Snort	Integração	Controlador	Detecção intrusão
Mattos *et* al.	X	x	X	x
Nagahama	✓	x	✓	✓
Le *et* al.	X	x	✓	✓
Chen *et* al.	X	x	X	✓
Este Trabalho	✓	✓	✓	✓

Como pode-se observar na Tabela 1, mesmo aquele que apresenta uma solução entre a ferramenta Snort e o Controlador *Openflow*, o mesmo desenvolveu uma solução própria utilizando os conceitos e características destas mesmas ferramentas. Desta forma, percebe-se que não houve nenhuma tentativa de integrar essas tecnologias sem a necessidade de se criar uma terceira solução para fim de pesquisas.

Este trabalho propõe a integração entre o IDS Snort e o Controlador *Openflow* sem a necessidade de desenvolvimento de uma nova ferramenta. Também é proposta a criação de novas regras de identificação de ataques de intrusão que serão criadas na ferramenta Snort contra ataques de intrusão direcionados aos serviços de Telnet e FTP.

3. Trabalho Desenvolvido

O desenvolvimento deste trabalho foi divido em três etapas. A primeira delas refere-se a forma de como foi realizado a integração entre o IDS Snort e SDN. A segunda etapa foi realizada a coleta e análise dos dados das intrusões destinadas a cada tipo de serviço e por fim, a terceira etapa é a criação detalhada de cada regra conforme alguns métodos escolhidos.

3.1. Integração IDS Snort e SDN

A arquitetura desta integração compreende uma das decisões mais importantes deste trabalho. Diante das limitações apresentadas como a falta de integração entre controladores existentes com dispositivos de segurança, mas especialmente com o IDS Snort para a detecção de ataques de intrusão, foi escolhido então controlador *Ryu.*

Este controlador oferece duas formas para a integração com o Snort, na primeira opção, tanto *Ryu* quanto o IDS trabalham em uma mesma máquina, onde o controlador recebe os pacotes de alertas via *Unix Domain Socket*, que é um mecanismo de comunicação entre processos que permite a troca de dados bidirecional entre processos em execução em uma mesma plataforma. A segunda opção é o controlador e o IDS trabalhar em máquinas diferentes, pois o Snort exige um poder computacional muito alto e o desempenho no ambiente de teste não seria afetado. O *Ryu* recebe os alertas via *Network Socket,* onde é o ponto final da comunicação entre processos e a rede, como por exemplo, a conexão sendo direcionado para uma porta 1234 em um computador localizado no endereço 192.168.2.2 [Ryu 2016].

Entre as duas opções, decidimos como a solução mais adequada sendo a de executar o controlador e o IDS em máquinas diferentes.

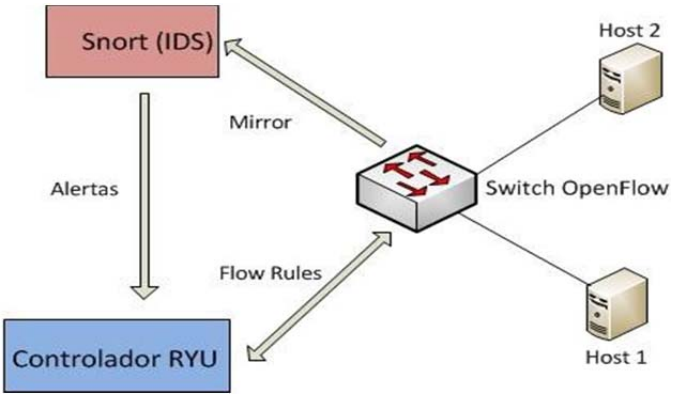

Figura 1. Arquitetura proposta para a Integração entre *Ryu* e Snort

Como ilustrado na Figura 1, para monitorar os pacotes entre os hosts, foi necessário realizar uma configuração no *openflow switch* denominada *Interface Mirror*. Desta forma é possível encaminhar uma cópia de todo tráfego que é transmitido dentro da rede para a interface do Snort, onde o processamento e a verificação das regras acontecerão. Caso alguma regra detecta alguma anormalidade na rede, o IDS enviará alertas ao controlador via *Network Socket*. Depois disso, o *Ryu* poderá tomar decisões na sua rede, assim tendo uma visão mais ampla dos acontecimentos dentro dela. Na figura 1, é ilustrada esta arquitetura.

3.2. Coleta e Análise de Tráfego

Nesta seção, apresentamos a fase da coleta dos dados que foram gerados na rede e a extração das informações que foram necessárias para elaborar as novas regras de intrusão. Para isso, utilizamos duas ferramentas para obter os dados necessários, a ferramenta *THC Hydra* para efetuar ataques contra os protocolos mencionados anteriormente e *Wireshark,* que é uma ferramenta que auxilia na análise do tráfego gerado na rede, e os organiza por protocolos através de uma interface, com a possibilidade de utilizar filtros conforme a necessidade do usuário. A Figura 2 apresenta o ambiente de teste com as ferramentas mencionadas.

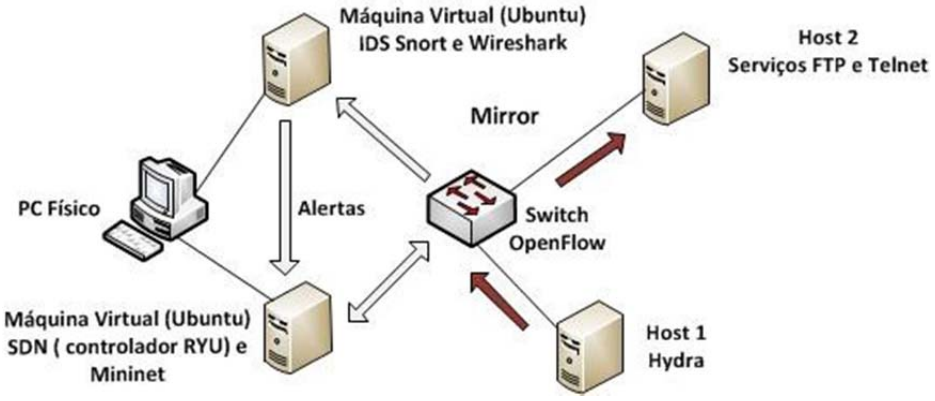

Figura 2. Ambiente de teste

Como mostra na Figura 2, em um primeiro momento, para verificar como um ataque de intrusão se comporta no exato instante em que o mesmo acontece e assim verificar seu funcionamento e seus padrões, utilizou-se a ferramenta *Hydra*, onde se efetuou ataques entre os hosts dentro do emulador Mininet com direção para os protocolos FTP e Telnet.

Para cada tipo de ataque existem alguns padrões, mas para verificar mais afundo cada tipo, foi necessária a utilização do *Wireshark*. Essa ferramenta captura todo o tráfego de uma interface de rede sem ter qualquer interferência no ambiente no qual esta

monitorando. Todo tráfego gerado durante os testes de intrusão realizados pelo *Hydra* e o tráfego normal do ambiente de teste, ficam gravados nesta ferramenta. Utilizando os filtros que a ferramenta fornece, filtramos cada tipo de ataque no qual pretendemos estudar e criamos um repositório para tipo de ataque, utilizando o próprio *Wireshark* para a verificação dos detalhes das intrusões no qual pretendemos criar as regras no Snort.

Tendo estas informações, já é possível a criação de regras no IDS Snort para a prevenção de ataques. Na próxima seção, é explicado como estas informações foram utilizadas na criação das regras dentro do Snort.

3.3. Desenvolvimento de Regras no Snort

Uma das vantagens de integrar Snort com SDN é a flexibilidade na criação das regras que a ferramenta oferece, assim podendo alertar o controlador sobre diversas tentativas de exploração e o mesmo podendo agir conforme sua necessidade. Nesta seção é apresentado o desenvolvimento das regras conforme os métodos propostos, no qual esta dividida em subseção: o desenvolvimento da regra na ocorrência de intrusão direcionada ao protocolo *telnet* esta na subseção 4.3.1. Já na subseção 4.3.2 está o desenvolvimento da regra que verifica a ocorrência de intrusão direcionada ao protocolo FTP.

3.3.1. esenvolvimento Regra na Ocorrência de Tentativa de Intrusão ao Telnet

Para o desenvolvimento da primeira regra, é analisado o tráfego gerado pelo ataque com direção ao protocolo Telnet para descobrir algum padrão que possa ser utilizado para a criação de tal regra dentro do Snort. Foi identificado que o usuário root, no corpo do pacote tanto no valor hexadecimal quanto em ASCII, tem algumas semelhanças que levamos em consideração. Analisando os dados, temos na grande maioria os hexadecimais: 72 6f 6f 74 0d 00. Então utilizamos os seguintes hexadecimais: 72 6f 6f 74 0d 00 no qual foi utilizado dentro da regra.

Também foi verificado o número de *Acknowledgement – ACK*, onde no cabeçalho TCP contém um campo com o número de reconhecimento deste pacote. O campo mostra o próximo número de sequência que o remetente do pacote TCP está esperando para receber. Analisamos qual o valor que mais tem repetição dentro do tráfego gerado pelo ataque, e definimos o valor do ACK em 78.

Após as analises realizadas, definiu-se a seguinte regra:

alert tcp any any -> any 23 (msg:"Possivel ataque Telnet"; flow:to_server; flags: A; ack: 78; content:"|72 6f 6f 74 0d 00|"; rawbytes; offset:0; depth: 6; classtype:bad-unknown; sid:3658011; rev:8;)

Onde **alert tcp any any** é de qualquer origem, **-> any 23** para qualquer destino em direção a porta 23. A definição **(msg:"Possivel ataque Telnet"** é a mensagem no qual o Snort irá mostrar na detecção de alguma intrusão. Para definir o valor de ACK é usado **flow:to_server; flags: A; ack: 78** e que o pacote enviado contenha o hexadecimal **content:"|72 6f 6f 74 0d 00|".** Para que o IDS não verifique todo o pacote é usado o limitador **rawbytes; offset:0; depth: 6**. A regra é classificado como **classtype:bad-unknown** sendo ordenada dentro do snort por **sid:3658011** . **S**ua revisão é indicada como **rev:8;)**.

Então, a regra desenvolvida tem como função identificar qualquer *host* de qualquer rede que tente acessar algum *host* da rede monitorada através da porta 23, em que o ACK seja 78 e o final do cabeçalho contenha os hexadecimais | 72 6f 6f 74 0d 00 |.

3.3.2. esenvolvimento Regra na Ocorrência de Intrusão Direcionada ao FTP

Para o desenvolvimento desta regra, analisamos o tráfego gerado pelo ataque com direção ao protocolo FTP, descobrir o tempo que leva cada tentativa de intrusão e verificar outro padrão que possa ser utilizada para a criação de tal regra dentro do Snort.

Foi verificado que o tempo em que levou cada intrusão é inferior a 1 segundo, independente do usuário no qual esta tentando acesso no *host* alvo. Como a ferramenta utiliza uma base de dados com possíveis chaves de acesso, terá muitas tentativas de conexão em um curto espaço de tempo, menor que 1 segundo.

Foi analisado também o valor do bit PSH flag, onde utilizando o bit 1 ativa esta função, onde informa ao TCP origem que ele deve enviar todos os pacotes, inclusive os que estiverem em sua memoria ao destinatário.

Após as análises realizadas, definiu-se a seguinte regra:

alert tcp any any -> any 21 (msg:"Possivel ataque FTP"; flags: P; threshold: type threshold, track by_dst, count 18, seconds 1; classtype:attempted-recon; sid:3648012; rev:1;)

Onde **alert tcp any any** é de qualquer origem, **-> any 23** para qualquer destino em direção a porta 23. A definição **(msg:"Possivel ataque FTP"** é a mensagem no qual o Snort irá mostrar na detecção de alguma intrusão quando o flag PSH estiver ativo **flags: P;** e se ocorrer 18 tentativas **threshold: type threshold, track by_dst, count 18** dentro de 1 segundo **seconds 1**. A regra é classificado como **classtype:attempted-recon** sendo ordenada dentro do snort por **sid:3648012. S**ua revisão é indicada como **rev:10;)**

Então, a regra desenvolvida tem como função identificar qualquer *host* de qualquer rede que tente acessar algum *host* da rede monitorada através da porta 21, em que a flag PSH esta ativa e que envia mais de 18 requisições no intervalo de 1 segundo.

4. Resultados

Para a validação deste trabalho, foi produzido um ambiente para testar e validar este trabalho, desde a integração entre controlador *Ryu* e o IDS Snort, passando pela análise de tráfego gerado pelas intrusões na rede ate a criação das regras dentro da ferramenta Snort. Estas regras foram desenvolvidas para identificar intrusões dentro de um ambiente de rede controlada, conforme o ambiente de teste apresentado no capítulo anterior.

Para simular os acessos na rede de teste foram utilizados computadores virtuais que receberam endereços IP do range 192.168.41.0/24. Já os hosts criados dentro do emulador Mininet receberam endereço IP do range 192.168.47.0/24. Após, foram realizados diferentes tipos de tentativa de intrusão aos serviços de Telnet e FTP, conforme representado pela Figura 3.

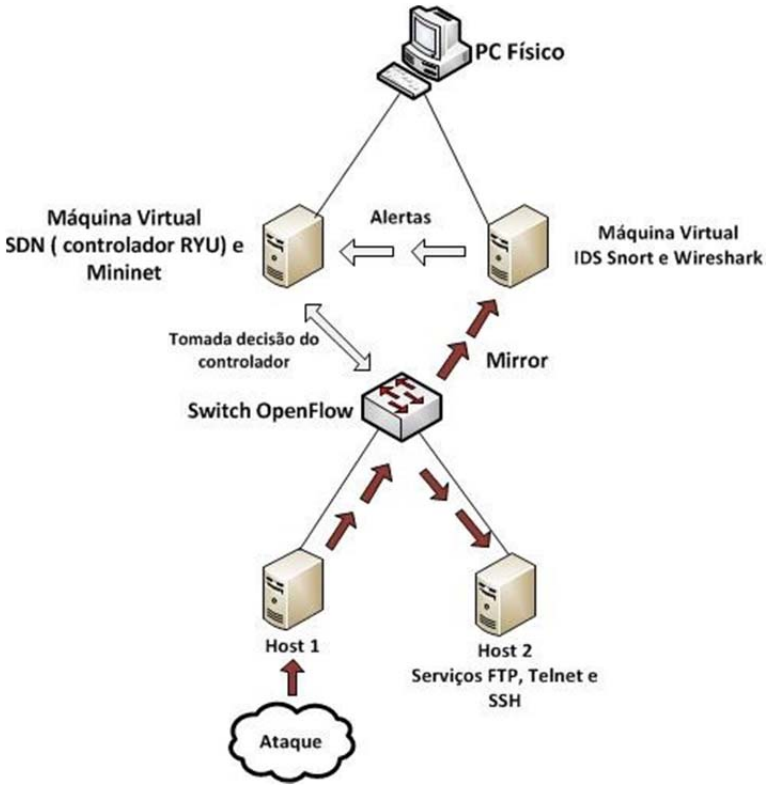

Figura 3. Ambiente desenvolvido para validação e resultados

O ambiente ilustrado pela Figura 3 apresenta o ambiente de teste desenvolvido, onde é identificado o host capaz de realizar algum tipo de intrusão que desrespeite as políticas impostas à rede. Neste cenário o ataque pode ocorrer somente dentro do mesmo ambiente de rede e a atividade é direcionada somente ao host 2.

Na validação das regras de detecção de intrusão, primeiramente utilizamos a ferramenta *Hydra* juntamente com a base de dados que contem diversas senhas para efetuar ataques entre os hosts dentro da rede SDN. Então para efetuar o ataque ao serviço Telnet, utilizamos a porta 23 e para o serviço FTP, a porta 21, no qual é o padrão destes dois serviços. Utilizamos um arquivo no qual continha 3 usuários: root, admin e administrador.

Após, com a ajuda do *Dataset* Darpa 99, inicialmente verificamos os momentos no qual aconteceram os ataques, pois o mesmo disponibiliza um arquivo que detalha todos os acontecimentos desta base, como qual serviço esta sendo atacado e seu tempo de duração. A seguir, com ajuda do *Wireshark* e um editor de texto, separamos os dados em dois diferentes arquivos: rede normal e rede com ataques.

Assim, tendo esses dois cenários, no qual cada um tendo um tráfego diferente foi possível verificar a ocorrência de falsos positivos e falsos negativos em relação às regras criadas no Snort. Desta mesma forma, conseguimos visualizar o Snort tratando o tráfego, analisando estes dados e enviando os alertas ao controlador *Ryu* e assim o mesmo podendo tomar decisões sobre essas requisições dentro do ambiente SDN

Finalizando, é possível afirmar que, dentro do ambiente testado, as regras propostas e seus métodos utilizando Snort, foram capazes de identificar ataques baseado nas regras desenvolvidas e enviar os alertas ao controlador *Ryu* dentro de um ambiente SDN. Da mesma forma, a integração entre as ferramentas propostas cumpriu com seu objetivo, no qual visa o auxilio ao uso futuro de administradores na implantação e gerenciamento dentro do paradigma do SDN.

5. Considerações finais

Através dos resultados obtidos e pelos testes realizados pode-se concluir que as regras criadas através dos métodos pesquisados são capazes de detectar possíveis ataques e assim auxiliar o gestor de rede a protegê-la. Da mesma forma, a integração entre o IDS Snort e o ambiente SDN através do controlador *Ryu* provou ser bem sucedida nos experimentos.

Após monitorar a rede e verificar as regras estabelecidas, o Snort envia os alertas correspondentes para o controlador *Ryu* para que possa tomar a melhor decisão possível em sua rede. A combinação entre estas tecnologias pode revelar-se muito eficaz, uma vez combinadas as vantagens de flexibilidade e controle da rede, podem evitar as vulnerabilidades que esse sistema tem atualmente e proteger contra ataques cibernéticos.

Como melhorias possíveis, utilizando esta integração entre o IDS Snort e o controlador *Ry*u, seria interessante o desenvolvimento de uma aplicação para o efetivo bloqueio destas tentativas de intrusão dentro da rede SDN. Desta forma, mostraria a integração completa entre esses sistemas, desde a geração de alertas através do Snort até o bloqueio deste tráfego malicioso com a utilização do controlador *Ryu* dentro do paradigma SDN.

Outro ponto que pode ser estudado com atenção diz respeito à detecção de outros tipos de intrusão como negação de serviço e a varredura de sistema. Como esta tecnologia ainda existe muitas vulnerabilidades, ela torna-se alvo fácil para um ambiente sem muita proteção, e estes tipos de ataques podem afetar a rede como um todo, pois a negação de serviço tende a deixar um sistema inativo por um período e a varredura de sistema o atacante detectar ainda mais as vulnerabilidades existentes dentro do ambiente de rede.

Referências

Bitencourt, William Lopes (2014). *Implementação de politicas globais em redes SDN utilizando marcação de pacotes*. Unisinos, São Leopoldo.

Chen, P. and Chen, Y. (2015). *Implementation of SDN Based Network Intrusion Detection and Prevention System.* ICCST.

Kim, H. and Feamster, N. (2013) *Improving Network Management with Software Defined Networking.* Communications Magazine, IEEE.

Le, A., Dinh, P., Le, H. and Tran, N. C. (2015). *Flexible Network-based Intrusion Detection and Prevention System on Software-defined Networks*. ICACA.

Mattos, D. M. F., Ferraz, L. H. G. and Duarte, O. B. (2013). *Um mecanismo para isolamento seguro de redes virtuais usando a abordagem hıbrida Xen e OpenFlow*. Em XIII SBSeg, p. 128–141.

Mckeown, N. and Anderson, T. (2008) *OpenFlow:* Enabling Innovation in Campus Networks. ACM SIGCOMM.

Nadeuau, T. D. and Gray, K. (2013). *SDN:* Software Defined Networks. ”O‟Reilly Media, Inc.”, 2013.

Nagahama, Fabio Yu. (2013). *IPSFlow:* um framework para Sistema de Prevenção de Intrusão baseado em Redes Definidas por Software. UFPA.

Ryu (2016). *RYU the Network Operating System.* Disponível em: http://ryu.readthedocs.io/. Acesso em: Setembro de 2016