# Integração de Detecção de Intrusão em Redes Definidas Por Software

**Laura Dalpissol[1], Cleverton J. A. Vicentini [1,2], Altair Olivo Santin[1]**

[1]Pontifícia Universidade Católica do Paraná (PUCPR)
Programa de Pós-Graduação em Informática (PPGIa)
Curitiba – Paraná – Brasil

[2]Instituto Federal do Paraná (IFPR)
Curitiba – Paraná – Brasil

{laura.dalpissol,santin,cleverton}@ppgia.pucpr.br,
cleverton.vicentini@ifpr.du.br

***Abstract.*** *This work describes the Snort Intrusion Detection System integration within the Software Defined Network architecture through the mininet network environment simulation. The objective is to identify malicious network traffic and provide this information to the administrator in real-time. The resuls presentes the Snort integration with Mininet and the malicious network traffic identification of ICMP Flooding and DoS.*

***Resumo.*** *Este trabalho descreve a integração do Sistema de Detecção de Intrusão Snort na arquitetura de Redes Definidas por Software através do ambiente de simulação de redes Mininet. O objetivo é identificar tráfego malicioso e disponibizar essa informação ao administrador em tempo real. Os resultados apresentam a integração do Snort com Mininet e a identificação do tráfego malicioso ICMP Flooding e DoS.*

## 1. Introdução

O paradigma intitulado Redes Definidas por Software (*Software Defined Networks - SDN*) [Kreutz, D., et. al., 2015], tem como principal diferencial a separação do plano de dados e plano de controle. O plano de controle é responsável pela lógica de encaminhamento da rede através da instalação de regras no plano de dados que é formado pelos dispositivos de encaminhamento. A comunicação entre estes planos ocorre tipicamente através do protocolo de comunicação OpenFlow [N. McKeown, et al. 2008]. De maneira geral quando não existe um fluxo para um novo pacote, esse é encaminhado ao controlador para devido processamento.

Em sua forma nativa as Redes Definidas por Software (RDS/SDN) não possuem ferramentas de detecção de intrusão, o que pode tornar o serviço indisponível de forma breve se o controlador sofrer um ataque de negação de serviço, por exemplo. Neste sentido esse trabalho propõem a integração de ferramenta de detecção de intrusão para monitorar o tráfego em nível de rede, mas externamente ao ambiente *Mininet*, onde a arquitetura das Redes Definida por Software está configurada. Esta abordagem traz a vantagem de integração de RDS com soluções legadas, mais complexas, de detecção de intrusão.

Este trabalho tem seu foco no monitoramento do tráfego malicioso em nível de

rede, Network Intrusion Detection System (NIDS). A integração das RDS com NIDS em sistema legado torna a solução de detecção de intrusão mais robusta, porque aumenta as fontes de dados, levando à detecção mais precisa e flexibiliza a arquitetura uma vez que RDS pode facilmente redirecionar pacotes no caso de detecção de um ataque por exemplo, livrando a vítima do mesmo.

O artigo está organizado da seguinte forma. A Seção 2 apresenta a fundamentação do trabalho. A Seção 3 detalha a metodologia utilizada para a realização do trabalho. A Seção 4 apresenta as experimentações realizadas e a discussão. E por fim a Seção 5 apresenta a conclusão do trabalho.

## 2. Fundamentação

Esta seção apresenta os conceitos relacionados às tecnologias de Redes Definidas por Software e de Sistema de Detecção de Intrusão.

### 2.1. SDN

O modelo SDN realiza o desacoplamento do plano de controle do plano de dados da rede [Kreutz, D., et. al., 2015]. O plano de controle contém a lógica responsável pelo estabelecimento das regras de encaminhamento de pacotes, enquanto o plano de dados aplica em tempo real as regras previamente programadas no plano de controle que são aplicadas para todos os pacotes recebidos pelos comutadores da rede.

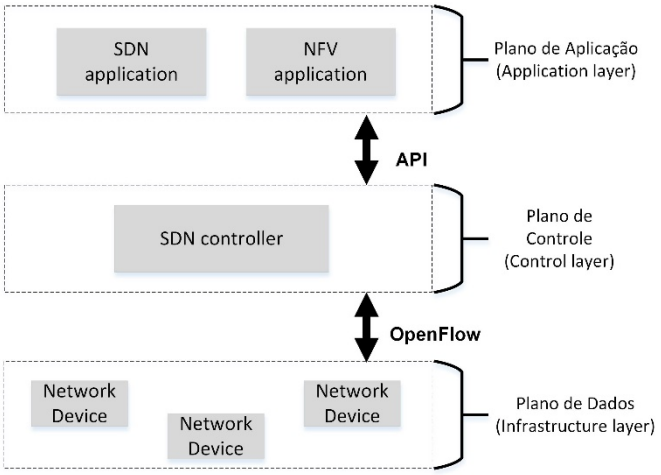


Figura 1 - Abstração do Modelo Redes Definidas por Software. Adaptado de ONF [ONF, 2014]

A Figura 1 representa uma abstração do modelo SDN através de 3 camadas: (i) Plano de Aplicação: em que rodam os sistemas e aplicativos dos usuários. (ii) Plano de Controle: responsável pela lógica nas definições das regras de encaminhamento dos pacotes na rede. (iii) Plano de Dados: representada pelos dispositivos (hardware) de rede, responsáveis pelo encaminhamento de pacotes baseados nas regras definidas no plano de controle.

O objetivo deste trabalho é integrar um Sistema de Detecção de Intrusão em nível de rede para monitorar todo o trafego do plano de dados da arquitetura de Redes Definidas por Software, a fim de identificar e informar ao administrador da rede a ocorrência de atividades maliciosas.

### 2.2. Sistema de Detecção de Intrusão

Sistema de detecção de intrusão (Intrusion Detection System - IDS) tem o objetivo de permitir ao administrador identificar possíveis quebras de políticas de segurança**,** permitindo que as ações necessárias sejam tomadas e a quebra de política não comprometa o restante do sistema [Corona, I., et. al., 2013].

Um IDS pode ser baseado em rede (NIDS – *Network-based Intrusion Detection System*), que monitora o tráfego da rede a fim de identificar possíveis atividades maliciosas na rede, tal como ataques de *DoS* ou *Probing*. E também pode ser baseado em host (HIDS – *Host-based Intrusion Detection System*) que monitora atividades maliciosas em um sistema específico, normalmente sobre chamadas de sistemas ou aplicações.

## 3. Metodologia

Para imitar um ambiente com as características das Redes Definidas por Software, foi adotada a ferramenta *Mininet* [B. Lantz, et. al., 2010]. A ferramenta *Mininet* permite a criação de infraestruturas de rede, de forma virtual que possuem todas as características intrínsecas do modelo de Redes Definidas por Software. Ainda segundo [Prete et al., 2014], um dos objetivos do *Mininet* é fornecer de maneira trivial e com baixo custo, subsídios aos pesquisadores para o desenvolvimento de aplicações com diferentes topologias do modelo de Redes Definidas por Software, sem a necessidade de uma rede física.

Definido o ambiente que será responsável pelo tráfego das Redes Definidas por Software (*Mininet*), é necessário definir qual ferramenta fará a fiscalização do tráfego de rede, a fim de cumprir a proposta deste trabalho de iniciação científica. Para isto a ferramenta escolhida foi o Sistema de Detecção de Intrusão intitulado *Snort*.

O *Snort* [Roesch, M. and Green, C, 2001] possui a metodologia de análise de tráfego de rede em tempo real. Seu objetivo é analisar os protocolos dos pacotes IP, e desta forma através dos padrões de conteúdo detectar diversos tipos de ataques. Neste sentido para este trabalho após a integração das duas ferramentas (*Mininet* e *Snort*), quando um determinado padrão de ataque for identificado no ambiente de Redes Definidas por Software, a ferramenta *Snort* gera um alerta e informa o ataque ao administrador.

Um serviço de monitoramento (SM) foi implementado na máquina que opera o IDS com objetivo de monitorar os alertas de ataques gerados pelo *Snort*. Isso permite informar em tempo real ao controlador SDN os eventos desta natureza. Esse serviço foi implementado como um web service RESTful através da biblioteca JAX-RS API[1].

A Figura 2, apresenta uma abstração do ambiente simulado e onde o *Snort* deve atuar, desta forma é possível fiscalizar todos os pacotes que trafegam internamente na rede *Mininet*.

Adicionalmente o trabalho realiza a fiscalização do tráfego externo ao *Mininet*, isso é possível através do *plugin hwintf.py*[2], que permite integrar uma rede virtual em operação no *Mininet* para se comunicar com o ambiente externo, ou seja possibilita a comunicação entre rede física e rede virtual.

---

[1] www.jax-ws.java.net (último acesso em junho de 2017)

[2] https://github.com/mininet/mininet/blob/master/examples/hwintf.py (último acesso em junho de 2017)

Neste sentido, o fluxo ilustrado na cor azul corresponde ao fluxo interno na rede *Mininet*, já o fluxo ilustrado na cor vermelha corresponde ao fluxo externo da rede *Mininet*, ou seja, através da rede física.

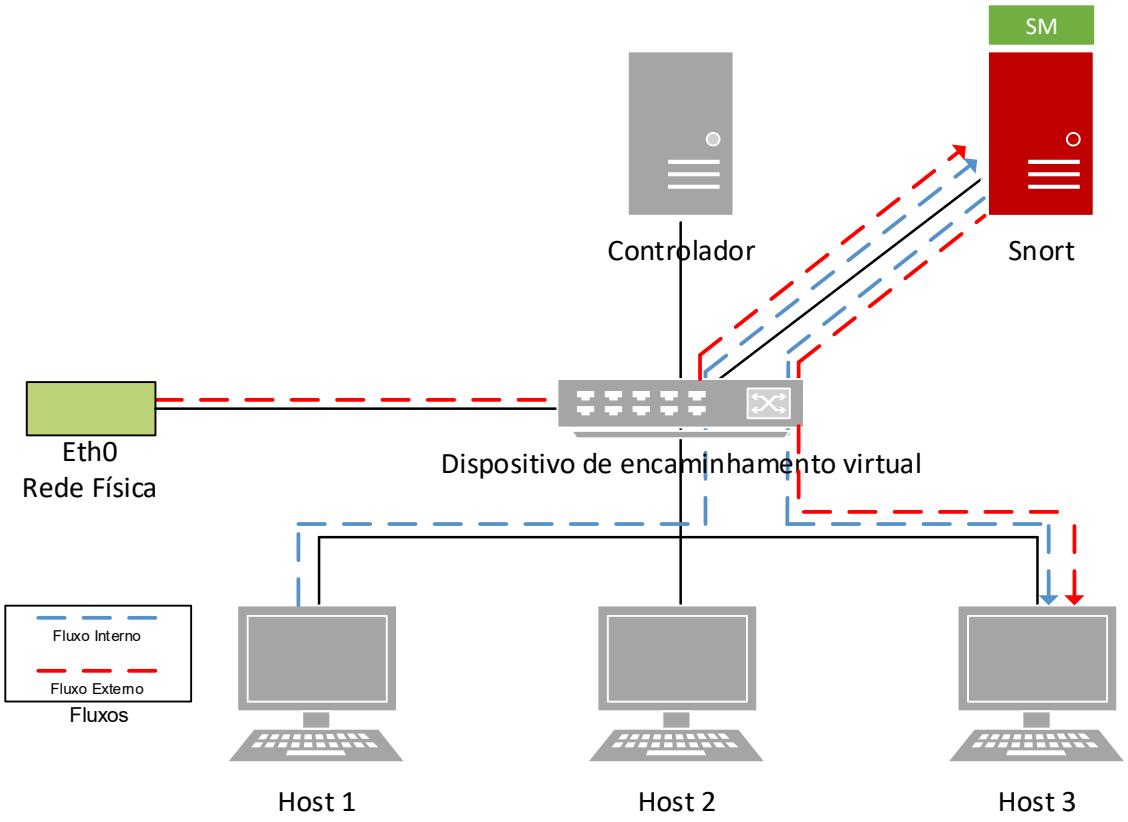


Figura 2 - Abstração do ambiente de teste.

### 3.1. Monitoração dos Ataques

A fim de realizar avaliações preliminares da integração das ferramentas (*Snort* e *Mininet*) e com o objetivo de manter o ambiente controlado, foram instaladas três assinaturas na ferramenta *Snort*, de forma a fiscalizar tráfegos e identificar tráfego ICMP e ataques de *ICMP Flooding* e *DoS*.

Inicialmente foi instalada e regra de fiscalização de pacotes ICMP, de forma nativa. O segundo passo foi alterar para identificação do *ICMP Flooding*, adicionando um contador de pacotes de modo que quando uma determinada quantidade de um mesmo pacote for contabilizada é gerado o alerta. E finalmente foi instalada a regra que fiscaliza ataques de negação de serviço (DoS), de modo que quando não tratados podem indisponibilizar rapidamente o controlador da rede ou o *host* alvo, por exemplo.

A Figura 3, representa as regras instaladas na ferramenta *Snort* a fim de testar a integração e monitoramento dos ataques. Para isto é necessário editar o arquivo de regras da ferramenta (*/etc/snort/rules*), e inserir as regras que deseja-se utilizar para fiscalizar atividades maliciosas na rede.

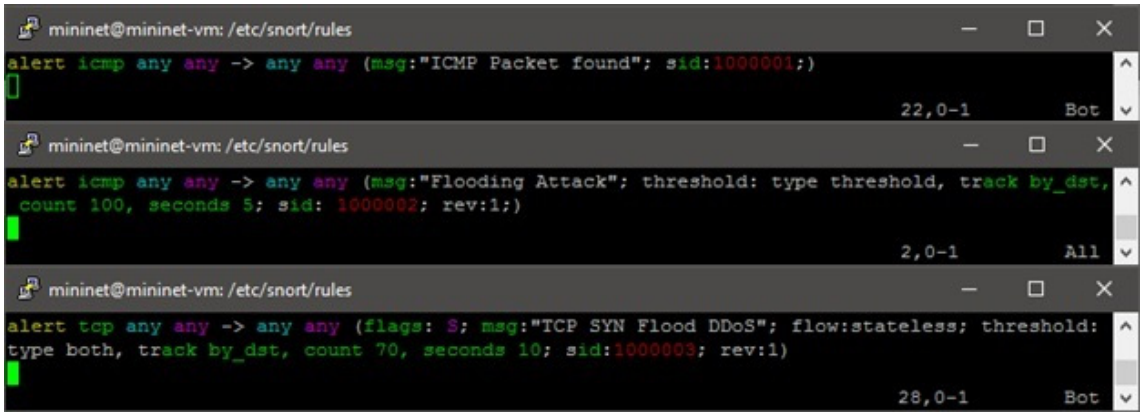


Figura 3 – Regras instaladas a fim de fiscalizar a rede.

As regras avaliam tráfego de a rede na *Mininet* e também externamente, fiscalizando a interface física que faz comunicação com o ambiente de teste.

## 4. Resultados Alcançados e Discussão

Para realização dos testes, inicialmente foi necessário criar uma topologia de rede na ferramenta *Mininet*. Desta forma, foi criada uma topologia simples, composta por três *hosts*, um *switch* e um controlador todos instanciados através do *Mininet*, com o seguinte comando: "*mn --custom mininet/examples/hwintf.py --topo single,3*".

Posteriormente a ferramenta *Snort* irá fiscalizar o *switch* da topologia instanciada no *Mininet*, de forma a avaliar o tráfego interno a esta rede, através do comando: "*snort -i (interface a ser fiscalizada) -c /etc/snort/snort.conf -A console*". Esse comando utilizado para fiscalizar o *switch*, que recebe como parâmetro a interface virtual que o *switch* opera, e também pode receber uma interface física, porém somente quando o *plugin hwintf.py* é ativado na inicialização da topologia.

Conforme já descrito na sub-seção 3.1, os primeiros pacotes fiscalizados são o de ICMP, para isto foi realizada uma operação simples de comunicação interna a topologia do *Mininet* através do seguinte comando: "*h1 ping h2*", que realiza uma comunicação entre o *host 1* e o *host 2*. A Figura 4 ilustra os pacotes capturados.

```
Commencing packet processing (pid=1655)
06/16-17:55:47.932207  [**] [1:1000001:0] ICMP Packet found [**] [Priority: 0] {ICM
P} 10.0.0.1 -> 10.0.0.2
06/16-17:55:47.932722  [**] [1:1000001:0] ICMP Packet found [**] [Priority: 0] {ICM
P} 10.0.0.2 -> 10.0.0.1
06/16-17:55:48.933018  [**] [1:1000001:0] ICMP Packet found [**] [Priority: 0] {ICM
P} 10.0.0.1 -> 10.0.0.2
06/16-17:55:48.933425  [**] [1:1000001:0] ICMP Packet found [**] [Priority: 0] {ICM
P} 10.0.0.2 -> 10.0.0.1
```

Figura 4 – Tráfego ICMP, gerado internamente ao Mininet e identificado pelo *Snort*.

O segundo teste tem o objetivo de monitorar o *ICMP Flooding*, para isso a regra do *ICMP* padrão foi alterada para que a partir de 100 pacotes em um intervalo de tempo de 5 segundos é gerado um alerta de *ICMP Flooding*. O comando utilizado entre os hosts é o seguinte: "*h2 ping –f –s 56500 h1*", a Figura 5 ilustra os pacotes capturados.

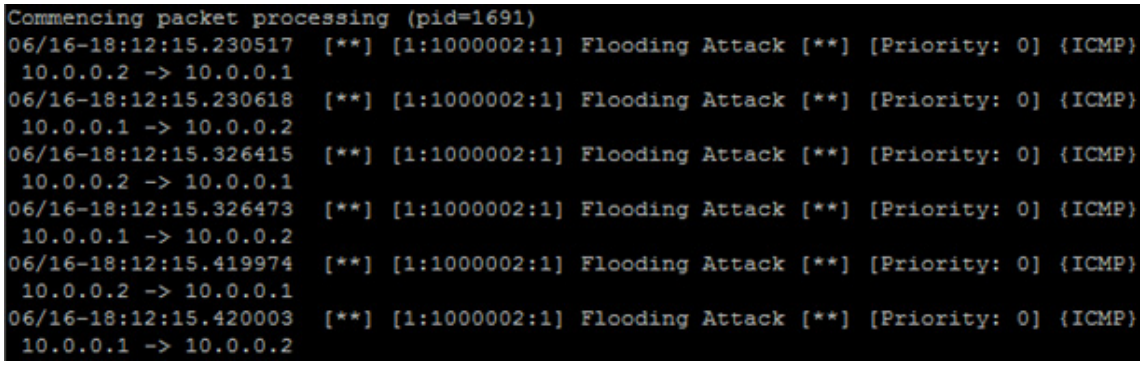


Figura 5 – Tráfego *ICMP Flooding,* gerado internamente ao Mininet e identificado pelo *Snort*.

Adicionalmente foi realizado um teste de fora da rede virtual, utilizando a interface física para a geração de um ataque *ICMP Flooding,* ilustrado na Figura 6 e os Pacotes capturados na Figura 7.

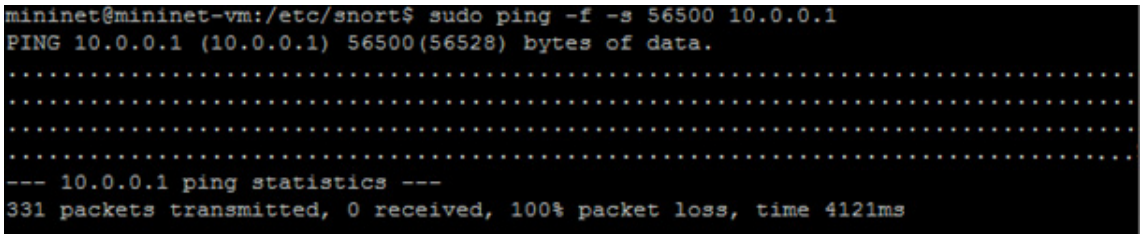


Figura 6 – Ataque *ICMP Flooding,* gerado externamente ao Mininet.

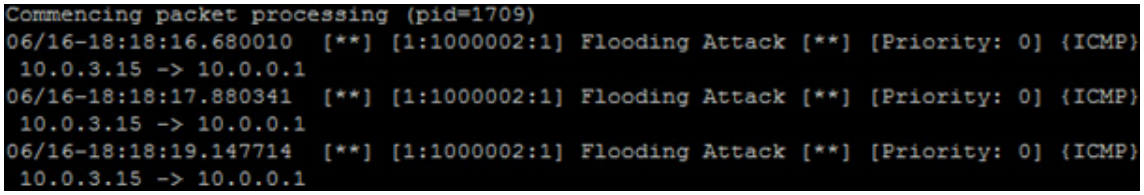


Figura 7 – Tráfego *ICMP Flooding* identificado pelo *Snort*.

O último teste realizado foi a identificação do ataque *DoS* no ambiente simulado de uma Rede Definida por Software. Para isso de forma externa, foi executado o seguinte comando: "*hping3 –S –flood –rand-source 10.0.0.1*", para gerar um ataque de negação de serviço. Os pacotes capturados são ilustrados na Figura 8.

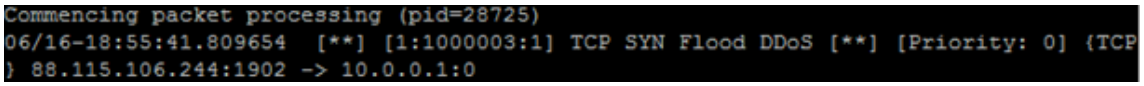


Figura 8 – Tráfego DoS identificado pelo *Snort*.

A partir do momento que a ferramenta *Snort* realiza o monitoramento do ambiente RDS e gera os *logs* de alerta quando ataques são detectados, o serviço de monitoramento (SM) informa ao controlador tais atividades, e assim torna-se possível realizar alterações nos fluxos de rede de forma a mitigar o ataque, para isso é necessário redirecionar o tráfego para um *host* preparado para suportar a demanda do ataque. Neste sentido o controlador RSD deve monitorar o *log* gerado pela ferramenta de NIDS (*Snort*), e ao identificar os alertas deve realizar a alteração nas tabelas de fluxo do controlador para migrar o ataque para outro *host*.

Para essa situação, este trabalho considerou para fins de teste que o *host* preparado para receber os ataques e trata-los é o *host2*, implementado num *honeypot*. Desta forma

o controlador irá alterar as tabelas de fluxo do dispositivo de encaminhamento de forma que esse nó (*host2*) receba os pacotes do ataque em andamento. A Figura 9, ilustra tal cenário.

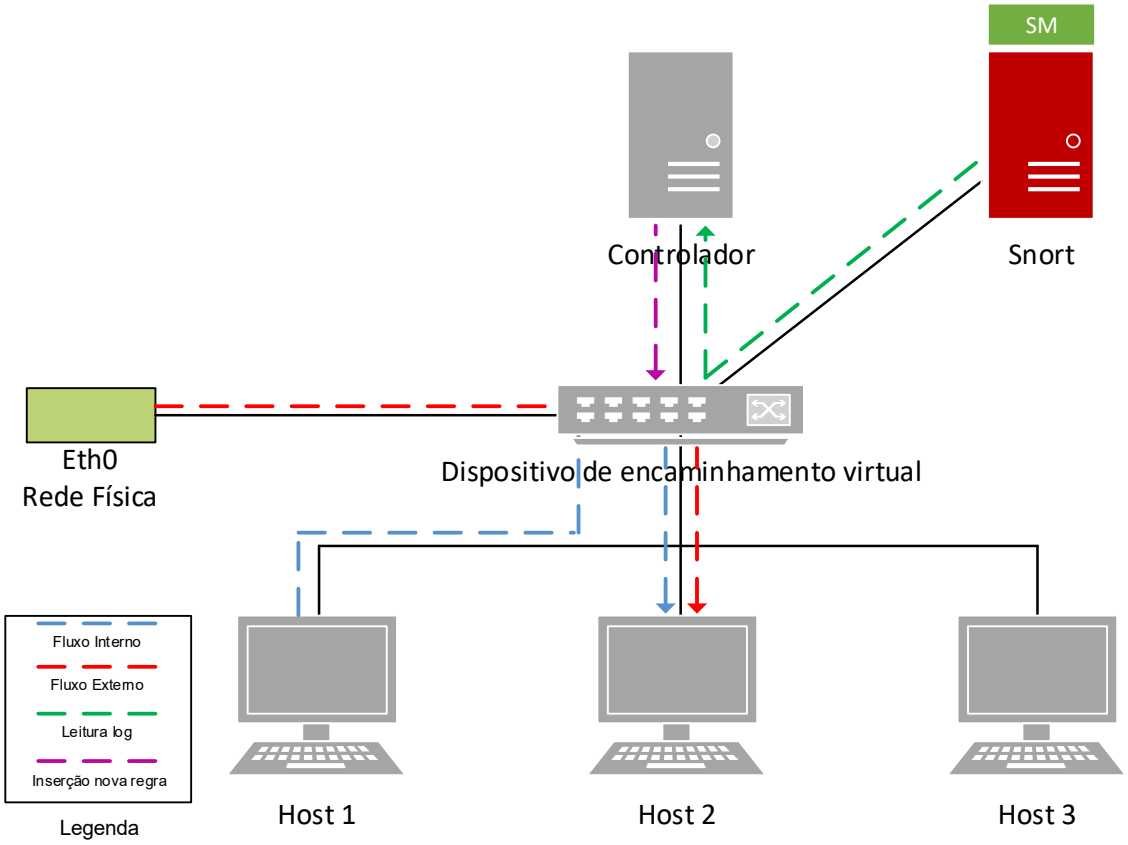


Figura 9 – Leitura de log e inserção de nova regra de fluxo.

Para o exemplo ilustrado na Figura 9 o controlador deve inserir um novo fluxo para o tráfego em andamento. Para listar fluxos em operação é executado o comando: "*dpctl dump-flows tcp:IP:6634"*, onde campo IP representa o endereço IP do controlador e a porta 6634 é o número utilizado por padrão pelos controladores das Redes Definidas por Software. Para o caso entre uma comunicação ICMP simples entre o *host1* e *host3* a saída do comando "*dpctl dump-flows*" será:

*stats_reply (xid=0xe789281e): flags=none type=1(flow)*

*cookie=0, duration_sec=0s, duration_nsec=33000000s, table_id=0, priority=65535, n_packets=1,n_bytes=42,idle_timeout=60,hard_timeout=0,arp,**in_port=1**,dl_vlan=0xffff,dl_src=0e:39:88:8a:48:f5,dl_dst=3a:10:37:48:85:fc,**nw_src=10.0.0.1,nw_dst=10.0.0.3**,nw_proto=1,actions=output:3*

*cookie=0, duration_sec=0s, duration_nsec=33000000s, table_id=0, priority=65535, n_packets=1,n_bytes=42,idle_timeout=60,hard_timeout=0,arp,**in_port=3**,dl_vlan=0xffff,dl_src=3a:10:37:48:85:fc,dl_dst=0e:39:88:8a:48:f5,**nw_src=10.0.0.3,nw_dst=10.0.0.1**,nw_proto=1,actions=output:1*

Esse log, permite identificar algumas informações como porta do *switch* virtual e os IPs dos *hosts* envolvidos no fluxo em questão. Neste sentido, quando o tráfego

malicioso for detectado o controlador deve inserir um novo fluxo, que redirecione o tráfego oriundo no nó atacante (*host1, port=1*) para o *host* que tem condições técnicas para tratar tal demanda (*host2, port=2*), isso pode ser realizado através da inserção da seguinte regra:

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

O comando anterior, realiza uma chamada para o controlador OpenFlow para solicitar a alteração no fluxo entre *host* atacante (*host 1*) e vítima (*host 3*), para o host que está preparado para tratar o ataque, neste caso o *host 2*. A partir deste momento todos os tráfegos destinados ao *host 3*, serão encaminhados para o *host 2*.

## 5. Conclusão

Este trabalho de iniciação cientifica propôs realizar a integração da ferramenta de IDS *Snort* ao simulador de Redes Definidas por Software *Mininet*. O objetivo foi estudar ambas as tecnologias e encontrar meios para que uma rede virtual *Mininet* fosse fiscalizada por uma entidade externa com o papel de identificar tráfego malicioso e fornecer alertas em tempo real. Através do *plugin hwintf.py*, torna-se possível tal integração e monitoramento da rede, isso pode ocorrer de forma interna e também de forma externa.

Para validar a integração foram realizados alguns testes controlados com três assinaturas na ferramenta *Snort*, fiscalizando tráfego/ataque interno e também tráfego/ataque oriundo da rede externa (física) ao ambiente de simulação.

O objetivo desse trabalho de iniciação científica foi alcançado em virtude da ferramenta *Mininet* permitir a criação de um ambiente completo de Redes Definidas por Software sem a necessidade de uma estrutura física real em operação, o que impacta em alto custo para a pesquisa em questão. Adicionalmente a integração da ferramenta de IDS *Snort* ao ambiente *Mininet* tornou possível a mitigação dos ataques, pois todo alerta gerado é informado em tempo real ao controlador, o que permite o redirecionamento do tráfego para outra máquina da rede.

## 6. Referências


Corona, I., Giacinto, G. e Roli, F., (2013) “Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues,” In: *Information Sciences. (Ny).*, vol. 239, pp. 201–225.

Kendall, K., (1999) “A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems,” In: Darpa Off-Line Intrusion Detection Evaluation, Proceedings Darpa Information Survivability Conference And Exposition (Discex).

Prete, L. R., Shinoda, A. A., Schweitzer, C. M., Oliveira, R. L. S. (2014) “Simulation in an SDN network scenario using the POX Controller”, In: Communications and Computing (COLCOM).

Roesch, M. and Green, C. Snort Users Manual, Release: 1.9.1. www.snort.org/docs/writing_rules/, 2001.

Kreutz, D., Ramos, F. M. V, Verissimo, P. E., et al. (2015). Software-defined networking: A comprehensive survey. In: Proceedings of the IEEE, v. 103, n. 1, p. 14–76.

Open Networking Foundation (ONF). (2014). SDN architecture 1.0, [Online]. Disponível em: opennetworking.org/

B. Lantz, B. Heller, and N. McKeown. (2010) “A Network in a Laptop: Rapid Prototyping for Software-Defined Networks”, In: ACM Hotnets ’10, Monterey, CA.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J. (2008). In: “OpenFlow: enabling innovation in campus networks,” SIGCOMM CCR, vol. 38, no. 2.