CURSO DE CIÊNCIA DA COMPUTAÇÃO

Thiago Oliveira Garcia

DEFINIÇÃO DE NOVAS REGRAS PARA O IDS SNORT EM REDES DEFINIDAS POR SOFTWARE

Santa Cruz do Sul

2016

CURSO DE CIÊNCIA DA COMPUTAÇÃO

Thiago Oliveira Garcia

DEFINIÇÃO DE NOVAS REGRAS PARA O IDS SNORT EM REDES DEFINIDAS POR SOFTWARE

Prof. Me. Charles Varlei Neu

Orientador

Santa Cruz do Sul

2016

CURSO DE CIÊNCIA DA COMPUTAÇÃO

Thiago Oliveira Garcia

DEFINIÇÃO DE NOVAS REGRAS PARA O IDS SNORT EM REDES DEFINIDAS POR SOFTWARE

Trabalho de Conclusão II apresentado ao Curso de Ciência da Computação da Universidade de Santa Cruz do Sul – UNISC, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof Me. Charles Varlei Neu

Santa Cruz do Sul

2016

AGRADECIMENTOS

Agradeço primeiramente a toda minha família pelo apoio constante. Aos meus pais Clademir e Maria Lucia por tudo o que me ensinaram durante a vida, pelo incentivo em todos os momentos e todo o apoio que me deram possibilitando a realização desse sonho. A minha namorada Scheila Muller, que sempre me deu forças nos momentos mais difíceis, de desânimo e cansaço, não me deixando jamais desistir, com todo seu carinho e determinação. Graças a eles eu tinha certeza que não estava sozinho nessa caminhada.

Agradecimento especial ao meu orientador, Prof Me. Charles Varlei Neu, pelos conselhos, indicando o caminho correto a ser seguido durante este trabalho. Não deixando de agradecer a todos os professores da UNISC, que contribuíram com o ensinamento necessário para chegar até aqui.

Também agradeço a todos meus parentes, colegas e amigos que contribuíram de uma forma ou de outra, para eu estar aqui hoje, firme e forte.

A todos, meus eternos e sinceros agradecimentos.

RESUMO

Redes Definidas por Software – SDN é um novo conceito de redes de computadores, pois proporciona a dissociação entre o plano de controle e o plano de dados. A partir dessa dissociação, a arquitetura das redes permite que a inteligência seja centralizada em um único ponto. Esse ponto possui uma visão global da rede, assim permitindo que os administradores de redes gerenciem todos os serviços de uma forma mais simples sem depender dos fabricantes de equipamentos de rede. Com uma centralização de serviço e uma segurança contra ataques de intrusão frágil, um ataque sendo bem sucedido pode vir a prejudicar uma rede inteira e gerar grandes problemas ao administrador. Por isso, Sistemas de Detecção de Intrusão ou *Intrusion Detection System* - IDS são essenciais para a verificação do tráfego em tempo real, em busca de novas intrusões que acontecem diariamente em uma rede, possibilitando que o administrador consiga protegê-la através da detecção provida por tais sistemas. Neste trabalho de conclusão 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*. A ferramenta THC Hydra foi utilizada para a geração do tráfego com os ataques direcionados para os serviços Telnet e FTP e seus pacotes analisados pelo *Wireshark* para verificar algumas características dos ataques dentro do ambiente de teste. Utilizamos o *Dataset* Darpa 99 que contém uma vasta base de ataques com as principais características de cada intrusão para a realização dos testes. 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.

Palavras Chaves: SDN, Segurança, Sistemas de Detecção de Intrusão, IDS**,** Regras de identificação, *Ryu*, Snort, *Dataset* Darpa 99.

ABSTRACT

Software Defined Networking - SDN is a new concept of computer networks as it provides the dissociation between the control plane and the data plane. From this dissociation, the architecture of the networks allows the intelligence to be centralized in a single point. This point has a global view of the network, thus allowing network administrators to manage all services in a simpler way without relying on network equipment manufacturers. With a centralized service and security against fragile intrusion attacks, a successful attack can damage a whole network and give the administrator big problems. Therefore, Intrusion Detection Systems (IDS) are essential for verifying real-time traffic in search of new intrusions that occur daily in a network, allowing the administrator to be able to protect it through these detections found by the system. 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 THC Hydra tool was used to generate traffic with targeted attacks on the Telnet and FTP services and their packages analyzed by Wireshark to verify some characteristics of the attacks within the test environment. We use the Dataset Darpa 99 that contains a wide base of attacks with the main characteristics of each intrusion to carry out the tests. 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.

Keywords: SDN, Security, Intrusion Detection System, IDS, identification rules, Ryu, Snort, Dataset Darpa 99.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
CAIS	*Centro de Atendimento a Incidentes de Segurança*
CLI	*Command Line Interface*
CPU	*Central Processing Unit*
DoS	*Denial of Service*
FTP	*File Transfer Protocol*
HIDS	*Host Intrusion Detection System*
ICMP	*Internet Control Message Protocol*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
NIDS	*Network Intrusion Detection System*
SDN	*Software Defined Networking*
SSH	*Secure Shell*
SSL	*Secure Socket Layer*
TCP/IP	*Transmission Control Protocol / Internet Protocol*
TTL	*Time To Live*

SUMÁRIO

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 de conclusão 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 de conclusão está organizado da seguinte maneira: o capítulo 2 apresenta os conceitos necessários para a correta compreensão do mesmo e os trabalhos estudados com relação a métodos de segurança em SDN encontradas na literatura. O capítulo 3 refere-se às principais tecnologias utilizadas no trabalho. O trabalho desenvolvido é apresentado no capítulo 4. O capítulo 5 mostra os testes realizados e os resultados obtidos. E finalmente, a conclusão e considerações finais são apresentadas no capítulo 6.

2. FUNDAMENTAÇÃO TEÓRICA

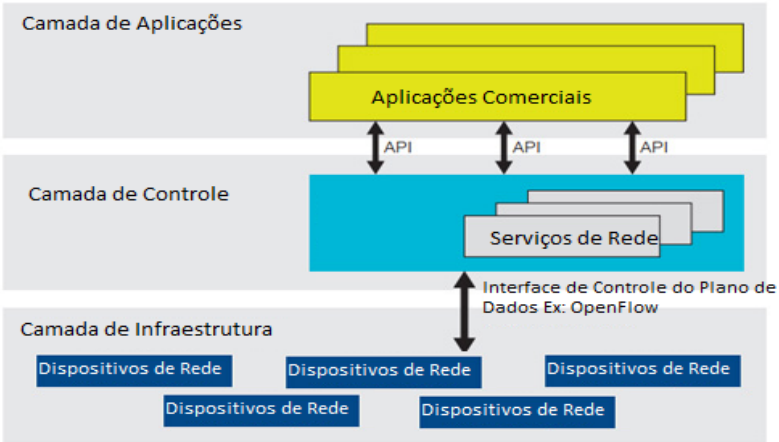
Neste capítulo são apresentados os conceitos básicos para a compreensão deste trabalho. O capítulo está organizado da seguinte forma: na Seção 2.1 são abordados os conceitos de SDN e o protocolo *OpenFlow*, na Seção 2.2 são analisados alguns aspectos de ataques de intrusão e na Seção 2.3 sistema de detecção de intrusão

2.1 SDN e o protocolo Openflow

Redes Definidas por Software constituem um novo paradigma em redes de computadores, pois proporcionam a dissociação entre o plano de controle e o plano de dados. A partir dessa dissociação, a arquitetura das SDNs permite que a inteligência da rede seja centralizada logicamente em um único ponto. Esse ponto possui uma visão global da rede, assim permitindo que os administradores de redes gerenciem todos os serviços da rede de uma forma mais simples, de uma maneira autônoma sem depender dos fabricantes de equipamentos de rede (KREUTZ et al, 2014).

A inteligência lógica da rede é centralizada em controladores. O controlador é o software que toma as decisões e que adiciona ou remove entradas na tabela de fluxos. Também trabalha com uma abstração da infraestrutura física, o que facilita o desenvolvimento de aplicações e serviços que controlem as entradas de fluxos na rede. Assim o controlador atua como um software de gerenciamento e controle da rede (ROTHENBERG et al, 2011).

Figura 1: Arquitetura de Rede Definida por Software (SDN)



Fonte: FOUNDATION (2016)

Como pode ser visto na Figura 1, a arquitetura SDN é dividida em 3 camadas. No topo está a camada de aplicação, que é a interface de gerenciamento do controlador, que em

conjunto com a Interface de Programação de Aplicativos ou *Application Programming Interface* - API de comunicação permite acesso aos parâmetros de configuração da camada de controle. A camada de controle é a camada intermediária e é o elemento mais crítico da SDN, pois nessa camada está toda a inteligência da rede. O controlador se comunica com a camada de aplicação através de aplicações de comunicação, de forma transparente. É responsável em fornecer os serviços pelas APIs e em abstrair os detalhes da rede. Na parte inferior, está a camada de infraestrutura, que opera um módulo SDN para receber as "ordens" da camada de controle e determinar seu funcionamento (FOUNDATION, 2016).

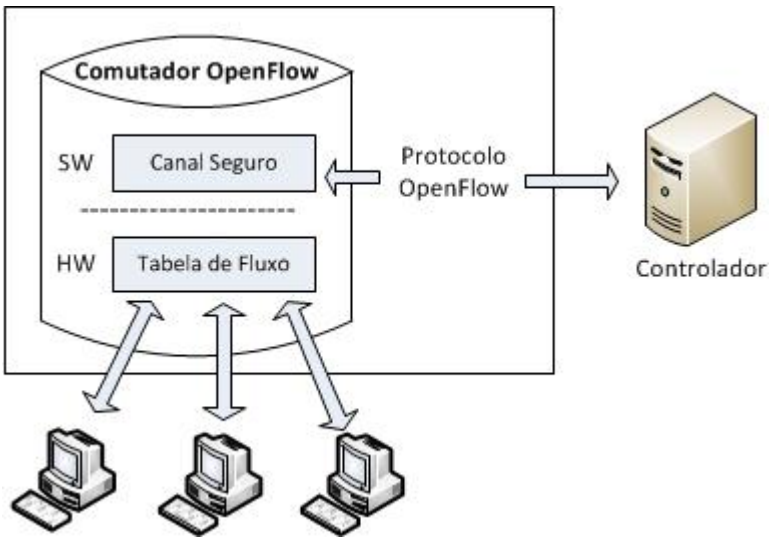
Para que o funcionamento de uma SDN seja possível, é necessário que exista um protocolo de comunicação entre os diferentes elementos de rede e o controlador, de forma que este último seja capaz de manipular as tabelas de fluxo dos elementos de redes. O protocolo com estágio de desenvolvimento mais avançado com essa finalidade é o protocolo aberto *OpenFlow* (ONF, 2016;MCKEOWN, 2008).

O protocolo *OpenFlow* é a base para a criação de um SDN que tem como principal objetivo permitir a utilização de equipamentos de redes comerciais para pesquisas, sem a necessidade de solicitar aos fornecedores o funcionamento interno de seus equipamentos. Ele permite a manipulação e o acesso direto do plano de encaminhamento dos pacotes de dispositivos de rede, como switches e roteadores, tanto virtuais como físicos (ONF, 2016).

O protocolo é implementado em ambos os planos, onde o equipamento do plano de dados possui uma tabela que armazena regras de encaminhamento de dados, chamada tabela de fluxos. Esta tabela armazena regras de encaminhamento de dados que determinam o fluxo, contadores e as ações a serem aplicadas. Estes contadores armazenam informações sobre os fluxos, como quantos pacotes trafegaram naquele fluxo, quantidade de bytes, *Internet Protocol - IP* de origem/destino, entre outros.

Este protocolo também prevê um canal seguro para comunicação entre switch e controlador, fazendo com que a comunicação não sofra com ataques mal intencionados. Para realizar esta tarefa, o protocolo *Secure Socket Layer* – SSL é utilizado (MCKEOWN, 2008). A Figura 2 mostra a arquitetura geral de um switch *OpenFlow* simples, suas ligações com o Controlador e os hosts. Também é possível observar que o switch está conectado a quatro hosts e ao controlador e possui uma tabela de fluxos. A ligação entre o switch e o controlador é realizada por meio do canal seguro, no qual é executado o protocolo *OpenFlow*.

Figura 2: Arquitetura Openflow



Fonte: COUTO (2016)

2.2 Ataques de Intrusão

Um ataque de intrusão pode ser definido como um conjunto de ações que tentam comprometer recursos de um sistema de computador ou inúmeras tentativas de explorar informações de qualquer natureza, independente de ter sucesso ou não destas ações. Estes ataques têm como objetivo corromper três dos componentes básicos da segurança de informação (FERREIRA, 2003;VACCA, 2013):

• **Integridade:** este princípio estabelece que toda informação só possa sofrer acréscimos, reduções ou atualizações por pessoas previamente autorizadas, o que mantém suas características originais e que, portanto, é integra.

• **Confidencialidade:** princípio que estabelece a garantia que somente pessoas previamente autorizadas tenham acesso à informação. Além disso, estabelece os casos em que a divulgação da existência da informação é proibida.

• **Disponibilidade:** garantir que toda informação deve estar sempre acessível às pessoas previamente autorizadas, no momento em que necessitarem utilizá-la. Sua eficiência esta atrelada aos princípios vistos anteriormente, pois no momento em que a informação está disponível é necessário que sejam garantidas a confidencialidade e a integridade do material a ser acessado.

Um ataque de intrusão pode explorar diversas falhas de segurança de sistemas, protocolos, aplicações ou ainda falhas de configurações. Eles também podem ocorrer nas diferentes camadas do modelo *Transmission Control Protocol / Internet Protocol* - TCP/IP e, de acordo com a camada que atuam, esses ataques são classificados em quatro grupos (BHARDWAJ, 2007;ZAMAN, 2009):

• **Negação de Serviços ou *Denial of Service* - DoS:** esse tipo de ataque tem como objetivo gera indisponibilidade de recursos em um determinado sistema ou equipamento, como consumo excessivo de memória, processador ou rede. Para isso, um atacante gera muitas requisições de modo que os recursos do sistema fiquem sobrecarregados, ao ponto de não conseguir mais atender as requisições de usuários legítimos, impedindo os acessos aos recursos ou informações no local onde está alocado o sistema.

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

• **Remoto para Usuário:** ataque onde o intruso possui conexão com a máquina vítima, que sem possuir uma conta de usuário e explorando alguma vulnerabilidade existente no sistema, consegue obter acesso local à máquina.

• **Usuário para Superusuário:** engloba todos os ataques em que o intruso tem acesso ao sistema como usuário normal e que realiza diversas tentativas de adquirir privilégios de Superusuários do sistema, possibilitando assim realizar qualquer tipo de operação.

Os ataques de intrusão ocorrem das mais diversas formas e em todas as camadas do Modelo TCP/IP. Contudo, existem sistemas que possuem componentes que desempenham funções como sensores e analisadores de eventos, que provêm a capacidade de detectar, analisar e identificar cada tipo de evento, os Sistemas de Detecção de Intrusão, ou *Intrusion Detection System* – IDS.

2.3 Sistemas de Detecção de Intrusão

Segundo Nobre (NOBRE, 2007), os IDS são sistemas automáticos que funcionam como sniffers, onde monitoram em tempo real o tráfego na rede e detectam tentativas não autorizadas de acesso à infraestrutura lógica. São ferramentas de segurança que, como antivírus e firewalls,

se destinam a reforçar a segurança de um ambiente de rede, de modo que os requisitos básicos de segurança em sistemas de informações não sejam corrompidos.

Desta forma, os IDS são considerados como uma das principais ferramentas de defesa contra ataques a sistemas, onde se torna um componente essencial para qualquer sistema de segurança em rede nos dias atuais. Seu monitoramento se baseia nos tipos conhecidos de ataques e também verificando alterações de comportamento no tráfego de dados, onde ocorrendo uma tentativa de intrusão, o sistema é capaz de gerar um alerta informando o ocorrido aos responsáveis pela segurança (NOBRE, 2007),

Para identificar um ataque, um IDS pode utilizar duas formas de detecção:

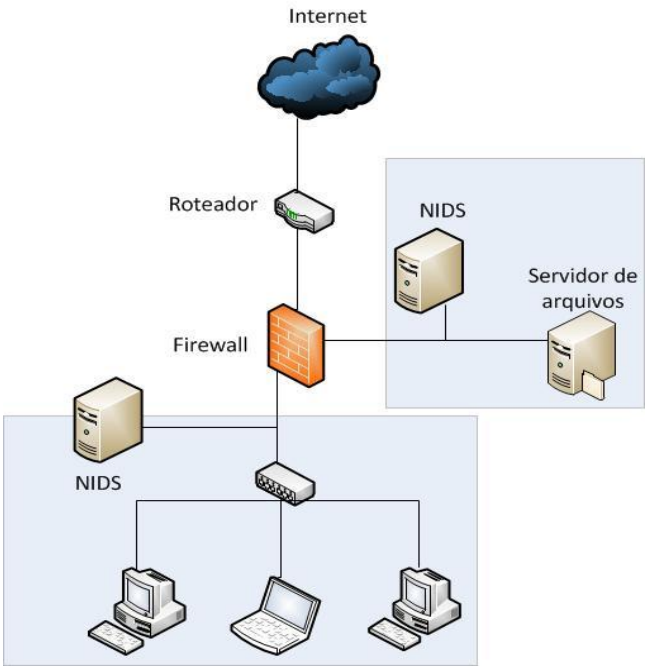
- **Baseado em regras ou assinaturas:** Neste tipo, o IDS visa à detecção de intrusos através da utilização de uma base de dados de ataques conhecidos. Esta base de dados é composta por um conjunto de regras que caracteriza determinado intruso. Quando um ataque for identificado, é realizado um processo simples de comparação de um pacote contra a assinatura do intruso especificado. Sistemas deste tipo são muito eficientes na identificação de ataques e vulnerabilidades conhecidas, mas são muitos frágeis na identificação de novas ameaças (SABAHI, 2008).

- **Baseado em anomalias:** O método de detecção por anomalia, por outro lado, busca identificar um ataque na ocorrência de algum evento fora do padrão usual, como anomalias da rede, tráfego em portas incomuns e elevado volume de tráfego, tentativa de acesso não autorizado a um recurso do sistema, comportamentos anormais do sistema que possam indicar atividade maliciosa na rede. Com muitas variáveis, este método se torna mais trabalhoso e menos seguro, pois é realizada uma coleta de informações da atividade da rede e forma uma base de dados. A partir daí, o sistema faz comparações das ocorrências da rede com esta base de dados e alerta sobre atividades que estão fora do comum na rede do sistema. No entanto, torna-se difícil configurar um sistema especificando o que é comum e o que é incomum em se tratando de tráfego de rede e comportamento do sistema (SABAHI, 2008).

Em uma arquitetura de rede tradicional, existem alguns modos para que os IDS atuem em uma rede, e uma delas atua na identificação de ataques dentro de uma rede de dispositivos em uma arquitetura de rede tradicional que é classificado como (AL-JARRAH, 2014) (BHUYAN, 2014):

- ***Network Intrusion Detection System* – NIDS**: Tem seu funcionamento baseado no monitoramento de um segmento de rede e busca identificar atividades suspeitas que possam ocorrer. Nesta arquitetura, o IDS é responsável por analisar atividades que envolvam todos os dispositivos presentes nesta rede. Na Figura 3 é ilustrado a arquitetura de NIDS.

Figura 3: Posicionamento do NIDS em segmentos de redes



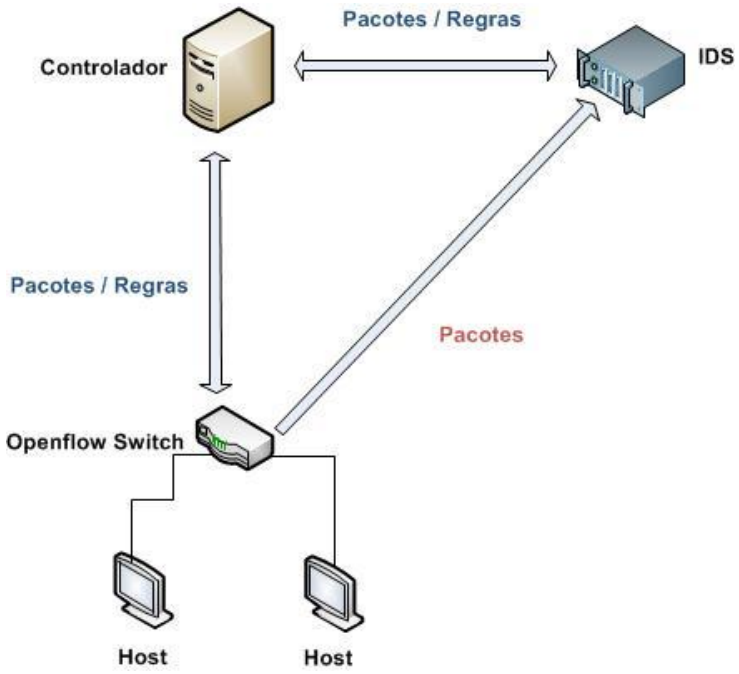
Fonte: BHUYAN (2014)

SDN tem a habilidade de efetuar alterações nos pacotes de saída, permitindo assim aos administradores da rede pré-determinar ações de resposta a serem efetuadas em certas intrusões ou eventos de confiabilidade. Um modo no qual o IDS pode atuar em uma SDN é mostrado na Figura 4 onde a ferramenta IDS faz a inspeção profunda de pacotes conectada à interface do controlador de fluxo e também ao *Openflow Switch*, e faz o seguinte processo (KREUTZ et al, 2014):

- O *Openflow Switch* define uma entrada de fluxo e duas portas de saída: a primeira porta será de encaminhamento de pacotes para *Openflow Switch* e a segunda porta é a conexão com o IDS
- Todo tráfego é encaminhado para o *Openflow Switch* e ao IDS
- O IDS examina os pacotes;

- O IDS informa o controlador sobre o que fazer com o fluxo de pacotes (descartar/ aceitar);
- O controlador transmite a ação para o *Openflow Switch* ;
- O *Openflow Switch* executa a ação (descartar/aceitar o tráfego)

Figura 4: Posicionamento do IDS em uma SDN



Fonte: KREUTZ (2014)

2.4 Trabalhos Relacionados

Durante as pesquisas, foram encontradas algumas propostas que visam à utilização de Redes Definidas por Software para a detecção de intrusões em ambientes virtualizados. A seguir, são apresentados alguns dos mais relevantes a este trabalho de conclusão.

No trabalho proposto por Mattos *et al*. (2013), o mesmo 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, prevenindo ataques de bisbilhotamento (*eavesdroping*). 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.

Vale apresentar também o trabalho desenvolvido por Nagahama (2013), onde o mesmo 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. Também permite a captura seletiva e distribuída de tráfego nos switches para análise de diversos IDSs. 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*. (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. O autor se baseia em algumas características dos ataques de intrusão, que se refere ao intervalo de tempo que acontece cada ataque, onde alguns podem acontecer dentro de um curto período de tempo como nos ataques de negação de serviço, enquanto outros podem ter uma duração mais longa, como uma varredura no sistema por minutos ou por horas. Ele também utiliza uma base onde armazena informações dos pacotes de rede, como cabeçalho de IP e *Internet Control Message Protocol - ICMP* e utiliza o *dataset* Darpa 99 para efetuar os testes de intrusão na rede. Após, essas informações são comparadas em uma base de dados onde existe uma classe de resposta, e utilizando um algoritmo de decisão, ele informa o resultado e, caso detecte alguma anomalia, ele informa o controlador SDN para tomar as ações que foram programadas para ele.

Já no trabalho realizado por Chen *et al*. (2015), utiliza um método que pressupõe que o atacante vai usar ferramentas de verificação para fazer um reconhecimento da rede, pois é muito fácil e rápido identificar os serviços que são executados em um host remoto, e ate mesmo em outros host que estão na mesma sub-rede. O autor cria dois algoritmos, o primeiro gera falsos hosts, que espera para o estabelecimento de conexão completa. O objetivo deste é permitir que o invasor não veja que sua tentativa foi vista pela vitima. O segundo algoritmo gera informações falsas, como portas abertas que irão responder ao atacante onde deixa um objeto livre para ser atacado com muitas informações abertas, sendo uma isca virtual para atrair atacantes em potenciais. Uma vez o atacante enviando algum pacote ou tentando estabelecer uma conexão, estas informações são salvas em uma lista, e com base nesta lista cria-se uma

regra de detecção, e até mesmo uma politica de segurança a fim de proteger a rede de novos ataques.

Na Tabela 1 é possível visualizar uma comparação entre os trabalhos relacionados e este trabalho de conclusão de curso. Onde o mesmo mostra se os trabalhos foram estudados utilizando o IDS Snort integrando com algum controlador SDN. Também verificou se os trabalhos focaram-se na detecção de intrusão dentro de SDN. Observou-se que o foco principal de estudo da maioria destes está relacionado a novos métodos de melhoria na detecção de intrusão.

Tabela 1: Comparativo entre trabalhos estudados e trabalho proposto

Trabalho	Snort	Integração	Controlador	Detecção intrusão
Mattos *et al*. (2013)	X	x	X	x
Nagahama (2013)	✓	x	✓	✓
Le *et al*. (2015)	X	x	✓	✓
Chen *et al.* (2015)	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 de conclusão propõe então, 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. As tecnologias que foram utilizadas para desenvolver este trabalho serão apresentadas no Capítulo 3. Os métodos escolhidos, juntamente com o trabalho proposto e sua arquitetura serão apresentados no Capítulo 4.

3. TECNOLOGIAS ENVOLVIDAS

Este capítulo apresenta as principais tecnologias utilizadas para desenvolver este trabalho. Na Seção 3.1 é abordado o sistema de detecção de intrusão Snort. Já a Seção 3.2 refere-se ao sistema Mininet, suas características e funcionalidades e na seção 3.3 é comentado sobre *Ryu*, que é um controlador SDN que pode ser conectado à rede emulada pelo Mininet.

3.1 Sistema de Detecção de Intrusão Snort

Snort é uma ferramenta IDS de código-fonte aberto, desenvolvida por Martin Roesch sendo muito popular pela sua flexibilidade nas configurações de regras e constante atualização frente às novas técnicas de invasão. Seu código fonte otimizado, é desenvolvido em módulos utilizando a linguagem C possuindo documentação de domínio público (SNORT, 2016). O Snort pode ser configurado para funcionar em três modos diferentes:

- **Modo *Sniffer***: no qual simplesmente lê os pacotes da rede e os mostra como um fluxo contínuo no terminal de saída.
- **Modo *Packet Logger*:** neste modo o Snort além de capturar o tráfego, grava as informações de todos os pacotes em disco;
- **Modo Sistema de Detecção de Intrusão de Rede:** esse modo, o sistema analisa o tráfego da rede comparando os pacotes capturados com regras pré-definidas, a fim de detectar alguma atividade maliciosa.

O Snort faz a sua detecção baseado em assinaturas utilizando uma linguagem flexível de regras para analisar o tráfego coletado. Por ser livre de distribuição, existe uma base de dados com milhares de assinaturas/regras que são disponíveis para download e que são atualizadas diariamente por usuários Snort que estão espalhados pelo mundo, assim permitindo a realização de atualizações constantes e respostas imediatas contra novos ataques (SNORT, 2016).

Durante a escolha da ferramenta IDS, também foi pesquisado a ferramenta Bro. Esta ferramenta, desenvolvida por Vern Paxson, utiliza método de assinaturas para detecção de intrusão. Possui uma grande capacidade para capturar dados de redes de alta velocidade sem acontecer perdas desses mesmos pacotes. Vem configurado com algumas politicas pré-prontas, possui uma linguagem própria para o desenvolvimento das regras, mas sua complexidade é maior que o do Snort (MEHRA, 2012).

Uma das vantagens do Snort é permitir que o usuário produza suas próprias regras. Estas regras constituem a parte mais importante do IDS que podem levar ao sucesso ou não da detecção das intrusões. Esta ferramenta utiliza uma linguagem de descrição simples e fácil de usar, assim permitindo customizar as regras de acordo com as características do ambiente no qual deseja trabalhar, refinando ou ampliando o poder de detecção do Snort (SNORT, 2016).

Uma desvantagem da ferramenta é que ela identifica apenas ataques conhecidos, ou seja, identifica o ataque através de regras definidas dentro do sistema. Assim o IDS deve ser constantemente atualizado diante da rapidez que novos ataques e vulnerabilidades surgem (SNORT, 2016).

O Snort sendo um especialista em detecção de intrusão, permite ainda ser configurado para reagir aos ataques, onde o sistema é configurado em modo de Prevenção de Intrusão ou *Intrusion Prevention System* - IPS. Desta forma identificando a ocorrência de algum ataque reage enviando, por exemplo, respostas de REJECT para o endereço que originou o ataque (SNORT, 2016).

Mas, apesar de ser capaz de reagir a um ataque, este tipo de ferramenta pode permitir ocorrências de falsos positivos e falsos negativos, o que pode impactar diretamente no desempenho da rede caso não tenha nenhum tipo de tratamento para evitar estas ocorrências. Os falsos positivos acontecem quando reconhece um comportamento normal da rede como sendo uma intrusão na rede. Já o falso negativo, ao contrario, é qualquer evento malicioso no ambiente que o IDS identifica como legítimo, e assim não gerando alerta (BUL’AJOUL; JAMES; PANNU, 2013).

Devido às características apresentadas anteriormente, a popularidade e flexibilidade de aplicação do IDS Snort, este sistema foi eleito como ferramenta de detecção para o desenvolvimento deste trabalho de conclusão.

3.1.1 Regras da Ferramenta SNORT

As regras do Snort constituem a parte mais importante que condiz com o sucesso ou não da detecção das intrusões dentro de uma rede. Com a criação de novas regras dentro de uma rede, a ferramenta se torna uma poderosa arma de segurança de redes. O Snort tem uma sintaxe bem simples para a construção de tais regras que facilita seu uso (REHMAN, 2003)

Abaixo é apresentada uma regra do Snort:

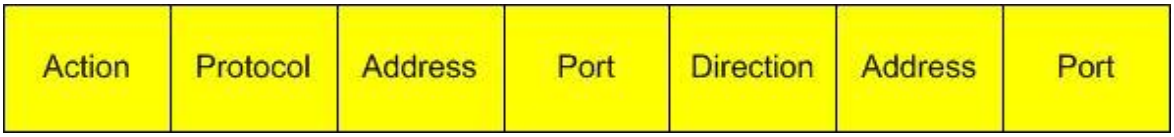
alert tcp any any -> 192.168.1.0/24 111 *(content:"|00 01 86 a5|"; msg:"mountd access";flags:A+;)*

A entrada que está em negrito é o cabeçalho da regra e o conteúdo que está entre os parênteses são as opções da regra. As opções de regras não são exigências, pois elas são usadas apenas por uma questão de fazer mais rígidas as definições de pacotes para coleta ou alertas nas regras na ferramenta Snort (SNORT, 2016).

3.1.1.1 Cabeçalho da Regra

O cabeçalho é a primeira porção de cada regra e define quem está envolvido. Contêm as ações das regras, protocolos, endereços IP de origem e destino, máscaras de rede e informações de portas de origem e destino (REHMAN, 2003). A Figura 5 mostra a estrutura do Cabeçalho da Regra (SNORT, 2016):

Figura 5: Estrutura do Cabeçalho da Regra no Snort



Fonte: (REHMAN, Rafeeq. 2003)

O primeiro campo do cabeçalho da regra é o *Action*, que instrui o Snort sobre quais são as medidas a serem tomadas quando forem satisfeitos os critérios estabelecidos. Pode tomar as seguintes ações:

- **Alert:** Cria uma entrada no diretório de alertas e faz registro do pacote, onde este diretório é único e armazena o registro de todas as detecções realizadas.
- **Log:** apenas um registro, onde não realiza qualquer registro do trafego no diretório de alertas.
- **Pass:** a regra sendo acionada, mas existindo 'pass' especificada na ação, o Snort irá fazer o *drop* do pacote, e não fará qualquer tipo de processamento do pacote.
- **Activation**: não se limita a alertar, mas também é utilizada para ativar outras regras que ficam em modo suspenso até serem ativadas.
- **Dynamic:** permanecem suspensas até serem ativadas por uma regra de ativação. Uma vez ativadas o seu comportamento é idêntico às das regras *log*.

No campo *Protocol*, o mesmo indica qual o tipo de tráfego na rede que a regra se destina ou aplica. A seguir, o campo *Address* define o endereço de origem e destino que podem ser de um único host, múltiplos host ou de endereços de rede.

No campo *Port*, determina a porta de origem de onde o tráfego é originado. Pode ser especificado como um número, um conjunto, ou ainda a palavra chave *any* que representa todas as portas possíveis.

Em *Direction*, este campo permite especificar o sentido da direção do pacote. Existe duas opções disponíveis, permitir especificar a direção do fluxo ou que a direção é indiferente para a regra. As opções válidas são:

- ->: define a origem e o destino
- <>: direção do pacote é indiferente (bidirecional)

3.1.1.2 Opções da Regra

Opções da regra é a segunda parte após o cabeçalho. Define que atributos do pacote devem ser inspecionados e quais os valores que devem conter para ser considerado hostil. Ele contém mensagens de alertas e informações sobre em que partes dos pacotes de rede que as informações devem ser inspecionadas para determinar que ação tomar. Por exemplo, no conteúdo do pacote, a carga, os *flags* ou cabeçalho. É somente usada se o pacote cumpre com todos os critérios dos requisitos do cabeçalho da regra (REHMAN, 2003) (SNORT, 2016).

Dentro das opções de regras, usa-se algumas palavras chaves especificas como (SNORT, 2016):

- **Id:** Verifica a identidade do cabeçalho TCP/IP para um valor específico;
- **msg:** Imprime uma mensagem de alerta e logs de pacotes;
- **flags**: Efetua um teste nas *flags* TCP/IP para certos valores;
- **seq**: Testa o campo número de sequência TCP/IP para um valor específico;
- **ack**: Função onde testa o campo de confirmação TCP/IP para um valor específico ;
- **iType**-: Testar o tipo de campo ICMP contra um valor específico;

- **minfrag:** Estabelece um valor limite para um tamanho menor aceitável do fragmento de TCP/IP;
- **ttl:** Estabelece um determinado valor do cabeçalho IP do campo *Time To Live* - TTL;
- **session**: Copiar uma informação da camada de aplicação para uma determinada sessão.

3.2 Mininet

O Mininet é um emulador que foi desenvolvido para a criação rápida de uma topologia de rede com hosts, switches, controladores e enlaces virtuais em apenas um único computador utilizando o protocolo *OpenFlow*, assim permitindo emular uma SDN com suas especificações em pouco tempo. O ambiente é disponibilizado em forma de uma máquina virtual com as ferramentas necessárias pré-instaladas, para que outras pessoas possam baixá-lo, executá-lo, examiná-lo e modificá-lo conforme a necessidade. Também existe a possibilidade de instalar a ferramenta em computadores com Linux nativo, sem a necessidade de executar em máquinas virtuais (MININET, 2016).

É importante elencar que existem outros emuladores, como por exemplo, o Estinet. Ele é um emulador proprietário, onde pode ser utilizado para a realização de simulações com diversos controladores. Mas ele apresenta muitas desvantagens, como o fato de pouco material disponível para estudo e pesquisa, ser pouco conhecido e utilizado para simulações em SDN, mas principalmente por ser um software proprietário e com custo, o que inviabiliza um trabalho de nível acadêmico (WANG, 2014).

Ao criar uma rede no Mininet, são emulados (KETI, 2015):

• **Links:** age como um fio conectando a duas Ethernets de rede virtuais, conectando hosts, switches e controladores.

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

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

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

Para interagir com a rede e controlar os dispositivos da rede emulada, o Mininet oferece uma interface de linha de comando ou *Command Line Interface* – CLI, assim permitindo que os nomes dos nodos (hosts, switches e controladores) sejam usados ao invés do endereço IP deles (KETI, 2015).

O Mininet é uma ferramenta muito confiável com desempenho extremamente rápido, onde resulta em uma plataforma eficaz para que a comunidade do SDN possa realizar diversos testes com alta fidelidade. A ferramenta permite emular até 4096 hosts em um único terminal, mas perde desempenho quando os recursos solicitados numa nova topologia ultrapassar os disponíveis na CPU - *Central Processing Unit*, ou exigir largura de banda superior ao da maquina física (YAN, 2015).

Assim, a ferramenta Mininet foi o emulador escolhido, pois fornece um modo simplificado e sem custos, para que pesquisadores consigam realizar testes, mesmo tendo topologias complexas, sem a necessidade de uma estrutura física para cada segmento da rede. Oferece condições para que os pesquisadores possam desenvolver no interpretador Python, que é fornecido pela ferramenta, à criação de muitos experimentos envolvendo suas arquiteturas de redes.

3.3 Controlador *Ryu*

Ryu é um controlador criado para SDN baseado em Python e suporta até a versão 1.5 do protocolo *Openflow*, onde fornece um conjunto de bibliotecas definidas para facilitar a criação de aplicativos para o gerenciamento de rede e controle, assim permitindo customizar o cenário para atender a necessidade mais especificas dentro de uma organização, pois o desenvolvedor pode facilmente modificar os componentes existentes ou implementar seu próprio código para atender suas demandas (RYU, 2016).

Durante a escolha deste controlador, foi pesquisado também o controlador NOX. Ele foi o primeiro controlador SDN desenvolvido. Inicialmente suportava as linguagens C++ e Python, mas as versões foram separadas, onde a versão Python tem o nome de POX e a versão C++ ficou como NOX. Desde que foi criado, o NOX era bastante adotado e foi usado como base

para outros controladores, mas seu uso esta em grande queda frente a novos controladores. Uma das razões é o fato deste controlador não suportar as versões mais recentes do *OpenFlow*, pois suporta apenas até a versão 1.0 (GUDE et al, 2008).

Pelas razões descritas no controlador NOX e pelos fatores que foram elencados anteriormente no controlador *Ryu*, elas nos levaram a escolha do controlador *Ryu* em conjunto com Mininet para a realização da simulação de SDN neste trabalho de conclusão.

4. TRABALHO DESENVOLVIDO

Com a criação de uma nova tecnologia, novas formas de explorar as vulnerabilidades são criadas, e é preciso se prevenir utilizando sistemas que permitam de alguma forma identificar estes ataques e com isso, se proteger do mesmo.

O objetivo principal deste trabalho de conclusão é implementar novas regras de identificação de ataques de intrusão que podem ocorrer em uma rede SDN com base em modelos propostos na literatura. Para a criação destas novas regras, são utilizadas algumas técnicas que foram apresentadas em duas propostas encontradas na literatura.

No artigo de Le *et al*. (2015), o mesmo cria a SDN com espelhamento de portas entre o Controlador e *OpenFlow Switch* com direção ao Snort para a captura e análise de todo o tráfego. O autor utiliza algumas características de como acontecem os ataques de intrusão, sendo uma delas o intervalo de tempo que acontece o ataque, onde algumas investidas podem acontecer dentro de um curto período de tempo, enquanto outros têm uma duração mais extensa, como uma varredura no sistema por minutos ou até horas. Ele também utiliza uma base onde armazena informações dos pacotes de rede que são informados na Tabela 2, no intuito de ter algumas características dos ataques que pretende identificar.

Tabela 2: Informações dos pacotes de rede

Nome do recurso	*Descrição*	*Tipo*
duration	Tempo da conexão (em segundos)	Contínuo
numTCPFin	Número de flags FIN	Contínuo
numTCPSyn	Número de flags SYN	Contínuo
numTCPReset	Número de flags RST	Contínuo
numTCPPush	Número de flags PUSH	Contínuo
numTCPAck	Número de ACK	Contínuo
numTCPUrg	Número de flags URG	Contínuo
numPktSrc	Número de pacotes da origem para o destino	Contínuo
numPktDst	Número de pacotes do destino para a origem	Contínuo
protocol	Tipo protocolo	Contínuo

Fonte: (LE, A. et al. 2015)

No trabalho feito por Chen *et al*. (2015), os autores sugerem um método onde pressupõe que o atacante irá usar ferramentas de varredura de rede para identificar os serviços que são executados em um host remoto, e até mesmo em outros hosts que estão na mesma sub-rede. Os

autores implementam alguns algoritmos, e o objetivo de uma delas é gerar informações falsas, como quais portas estão abertas e que irão responder ao atacante. Uma vez que o invasor envia pacotes através das ferramentas de detecção para mais de um host dentro de um tempo limitado, a fonte que o atacante enviou o pacote para o falso anfitrião será gravada na "lista suspeita". No entanto, se o atacante não enviar pacotes para o host alvo, mas realizar a varredura de portas para detectar os serviços trabalhando em um dos outros anfitriões, e procurar os pontos fracos, ele estará na "Lista suspeita" onde será possível criar a regra no Snort através destas informações.

Um dos métodos utilizados na criação de regras no Snort a de um host como sendo alvo de ataque, onde inicialmente não terá nenhuma assinatura para detecção de intrusão, onde utilizaremos uma ferramenta para realizar ataques direcionados a este host que terá como objetivo fornecer informações sobre os ataques alvo deste trabalho de conclusão. Um atacante envia pacote em uma determinada faixa de tempo, então, utilizaremos esta informação juntamente com as descritas na Tabela 2 para determinar as características dos ataques. Utilizando este método, retiramos muitas informações que são úteis para a criação de novas assinaturas de detecção, pois tendo um host como alvo e os logs dos ataques que fazem as descobertas de vulnerabilidades, temos uma base de informações de como é realizado cada ataque no qual desejamos detectar e assim criar novas assinaturas na ferramenta Snort.

Para verificar o real funcionamento das regras, é utilizado o *Dataset* Darpa 99, onde seus dados foram gerados a partir de tráfego real em um ambiente dentro do Centro de Pesquisas Lincoln Labs do *Massachussetts Institute of Technology* - MIT. Este *dataset* foi produzido de uma forma simples, foi conectada a Internet uma rede de uma base da Força Aérea, produzindo atividades através de um *script* e foi injetado ataques em pontos definidos por um período de sete semanas (LIPPMANN et al, 2000). O mesmo é utilizado para analisar as funcionalidades das regras de intrusões entre o Snort e o SDN para a detecção de dois tipos de ataques:

- **FTP:** significa File Transfer Protocol (Protocolo de Transferência de Arquivos) e é um protocolo utilizado para transferir arquivos de uma máquina para outra em uma rede de computadores.
- **Telnet:** este protocolo propicia uma comunicação bidirecional entre dois nodos da rede. Sua maior utilização reside na emulação de terminais remotos, onde

> permite a um usuário estabelecer uma conexão TCP com um servidor remoto e assim o usuário executa comandos escritos no teclado em uma máquina que esta distante.

O desenvolvimento deste trabalho foi divido em três etapas e esta organizada da seguinte forma: na seção 4.1 é apresentada a forma de como foi realizado a integração entre o IDS Snort e SDN, na Seção 4.2 é apresentada a coleta e análise dos dados das intrusões destinadas a cada tipo de serviço e por fim, na Seção 4.3 é apresentado a criação detalhada de cada regra conforme o método descrito neste capítulo.

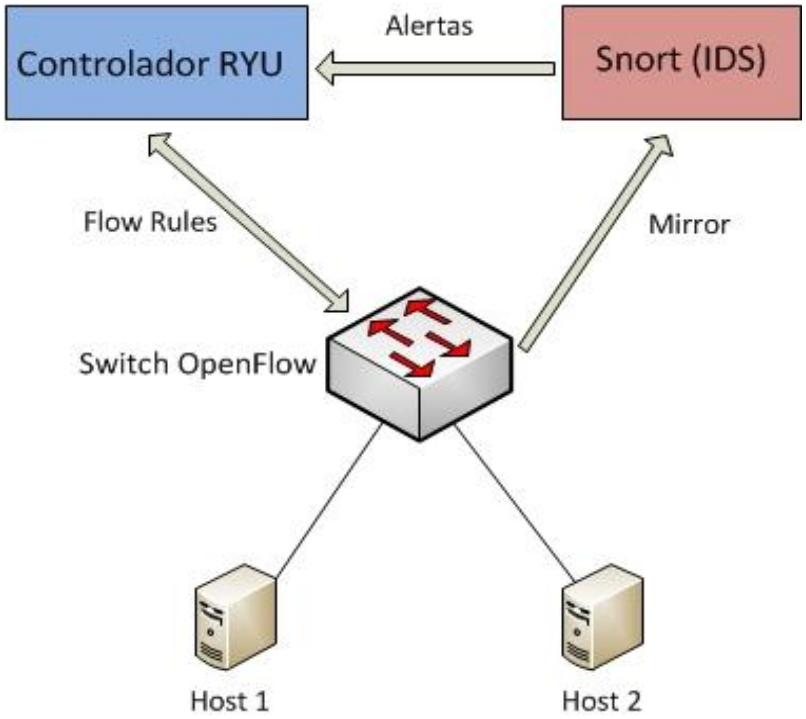
4.1 Integração IDS Snort e SDN

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. Na figura 6, a seguir, podemos ver a arquitetura que foi desenvolvida para este trabalho:

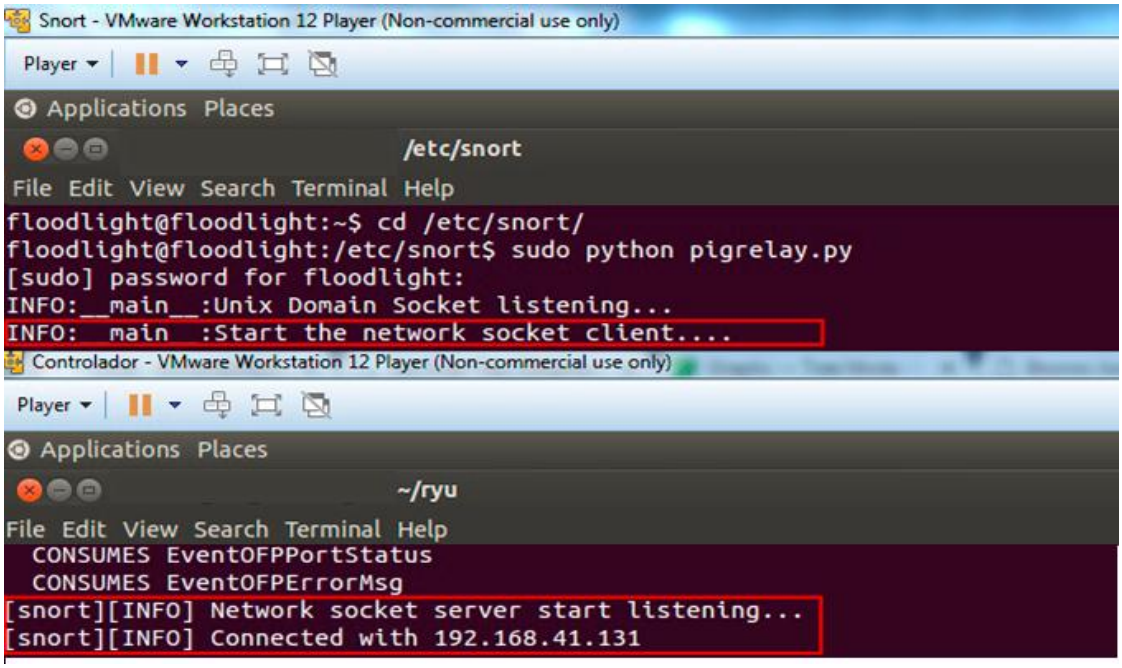
Figura 6: Arquitetura proposta para a Integração entre *Ryu* e Snort



Como ilustrado na Figura 6, 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.

A Figura 7 apresenta a requisição e conexão bem sucedida realizada entre Snort ao controlador *Ryu* através de uma API disponibilizada pelo próprio controlador, onde foi necessário modificar o modo de conexão para que funcionasse via *Network Socket* e o número da porta padrão de conexão ao Snort, para a correta conexão entre as ferramentas.

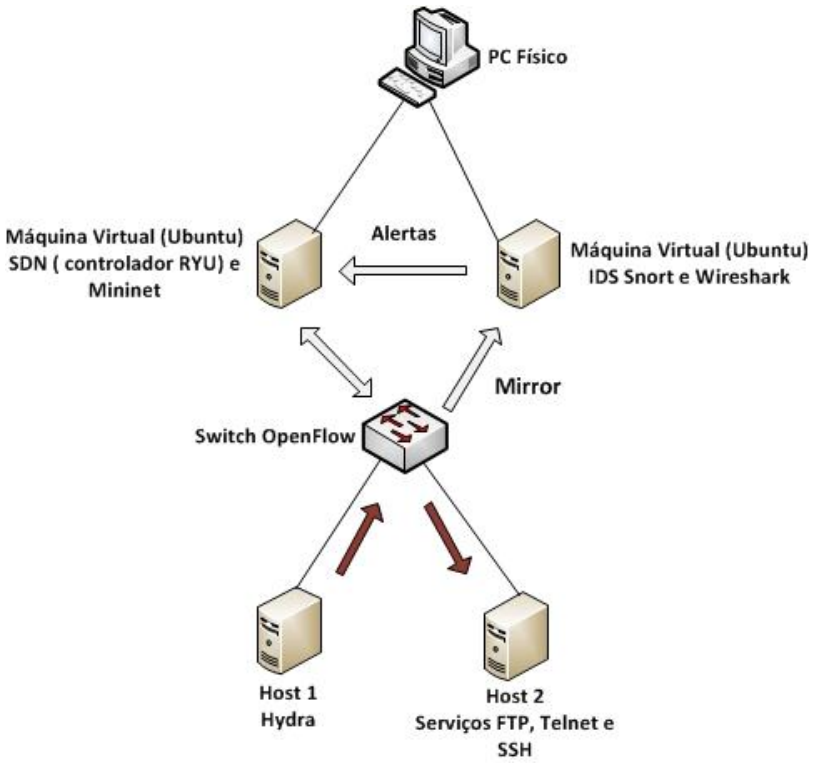
Figura 7: Conexão entre Snort e controlador



4.2 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 (WIRESHARK, 2016). A Figura 8 apresenta o ambiente de teste com as ferramentas mencionadas.

Figura 8: Ambiente de teste

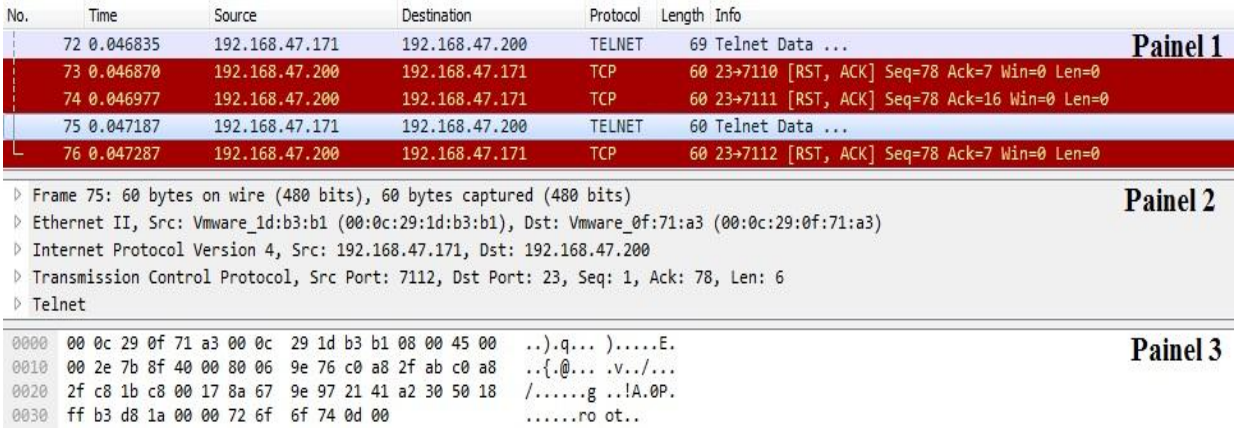


Como mostra na Figura 8, em um primeiro momento, para verificar como um ataque de intrusão se comporta no exato instante em que o mesmo acontece e assim verificar seu funcionamento e seus padrões, utilizou-se a ferramenta *Hydra*, onde se efetuou ataques entre os hosts dentro do emulador Mininet com direção para os protocolos FTP e Telnet. O host no qual é alvo, esta por padrão vulnerável a qualquer requisição enviada para ele, não tendo nenhum dispositivo que impeça os ataques de acontecerem.

O Hydra, uma ferramenta desenvolvida por Van Hauser e David Maciejak, utiliza como padrão de ataque *bruteforce*, onde ele consiste em consecutivas tentativas de verificação de senhas sobre um serviço em um determinado alvo, a qual se utiliza um banco de palavra para verificação dos dados (THC, 2016). Exemplificando, a ferramenta usa palavras dessa base e faz uma série de verificações com cada palavra até ser equivalente a real senha do host alvo.

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 como podemos ver na Figura 9.

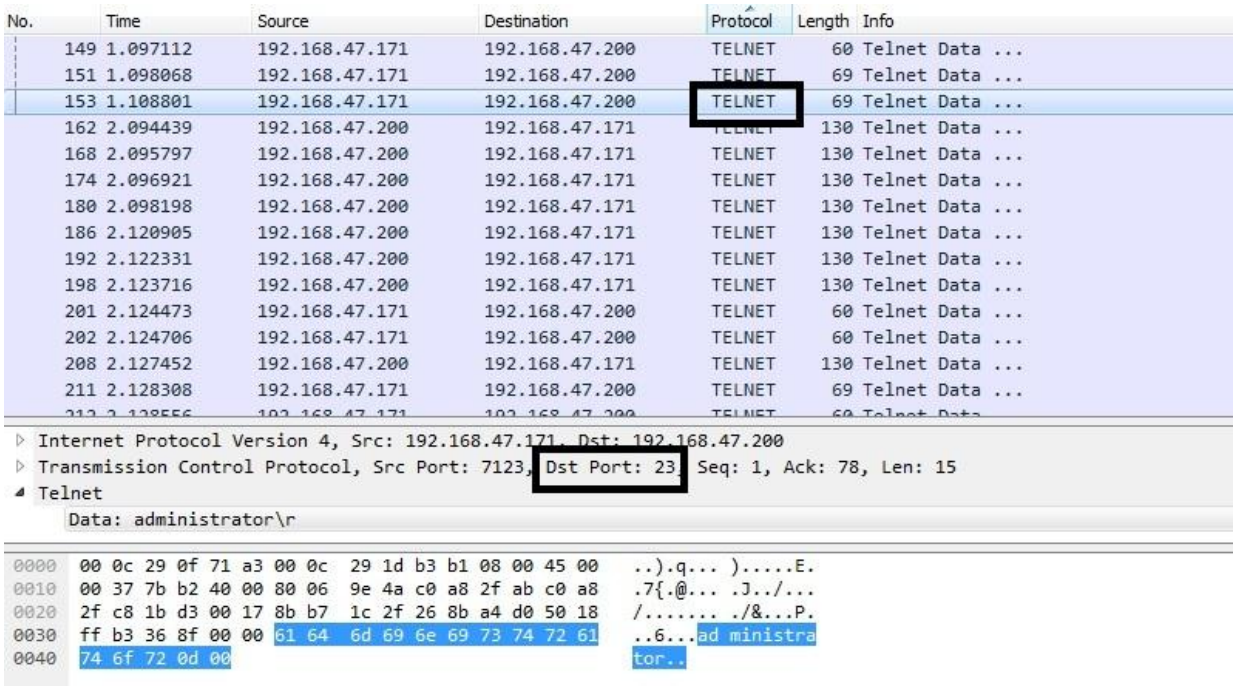
Figura 9: Tráfego coletado através da ferramenta *Wireshark*



Neste ambiente, a visualização da coleta é dividida em três painéis: no primeiro painel, cada linha se refere a um único pacote capturado e é possível identificar o endereço que realizou o ataque e qual o host de destino do ataque, os protocolos no qual esta direcionando os ataques entre outras informações. No segundo painel, mostra informações detalhadas do pacote selecionado e seus diferentes parâmetros. Esta organizada em uma estrutura de árvore no modo que pode ser expandido, possibilitando uma análise detalhada das informações do pacote. E no ultimo painel, mostra o pacote inteiro capturado em valores hexadecimal e em código *American Standard Code for Information Interchange* – ASCII.

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. A Figura 10 mostra o tráfego do ataque direcionado para a porta 23 do serviço de Telnet, onde os arquivos estão organizados para que apareça a tentativa de intrusão em sequencia, assim permitindo a análise de cada ataque em direção ao serviço específico.

Figura 10: Tráfego gerado pelo ataque no *Wireshark*



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.

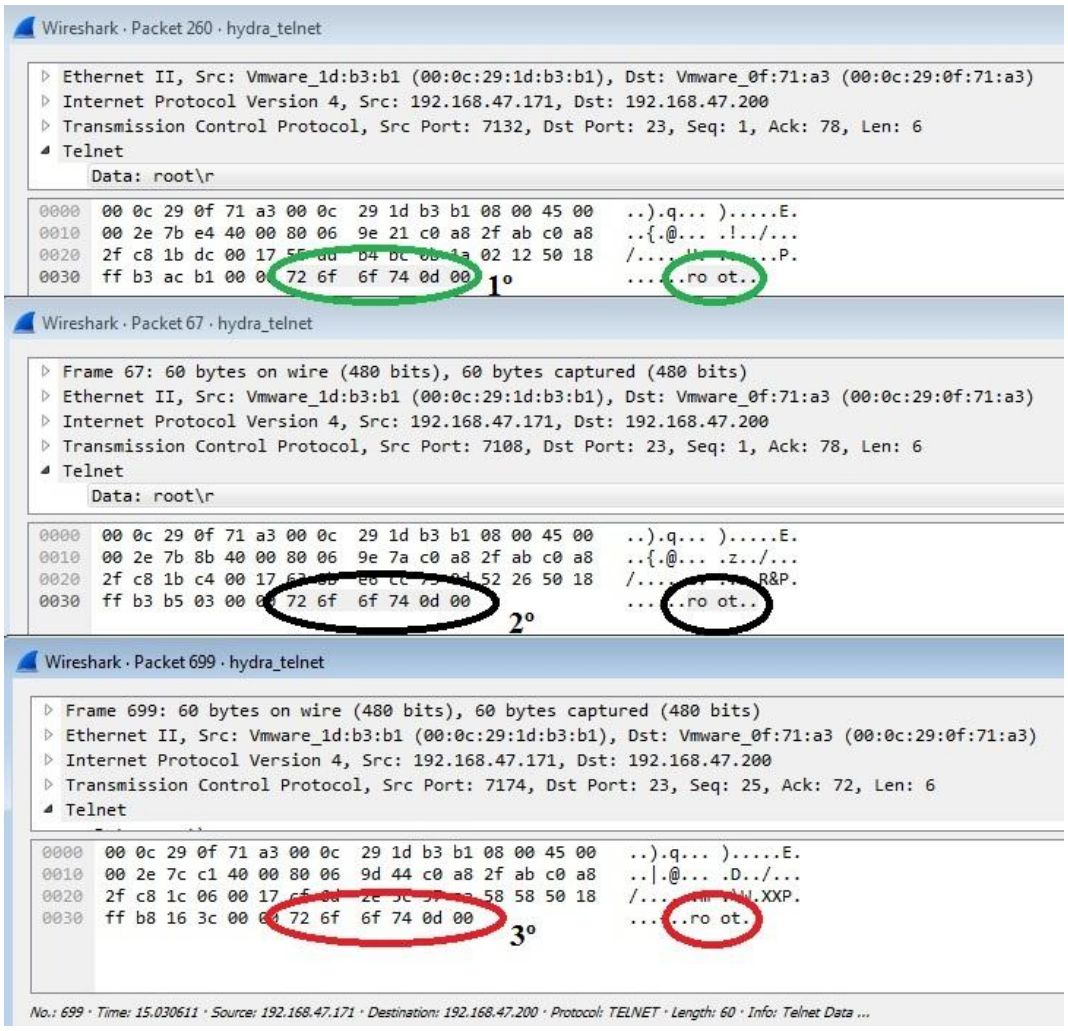
4.3 Desenvolvimento de Regras no Snort

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.

4.3.1 Desenvolvimento 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. Na Figura 11, é feito um filtro pelo serviço Telnet e verificado inicialmente, como seria o ataque caso o atacante acesse como usuário root.

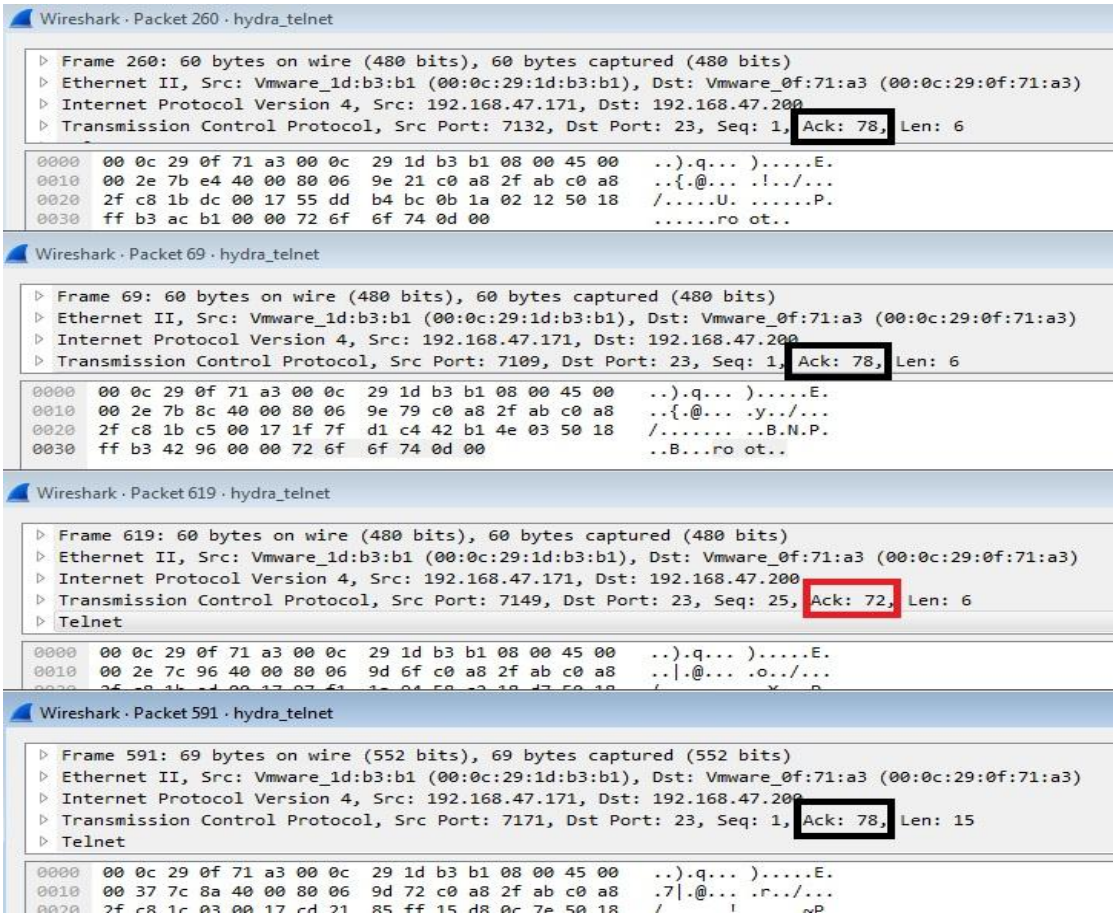
Figura 11: Análise do tráfego para ataque contra Telnet



Como mostra a Figura 11, logo 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 que estão circulados na Figura 11, temos no primeiro circulo os hexadecimais: 72 6f 6f 74 0d 00. No segundo e no terceiro circulo temos os mesmos hexadecimais apresentado anteriormente. 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. Na Figura 12, analisamos qual o valor que mais tem repetição dentro do tráfego gerado pelo ataque, e definimos o valor do ACK em 78.

Figura 12: Definição do valor de ACK



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;)

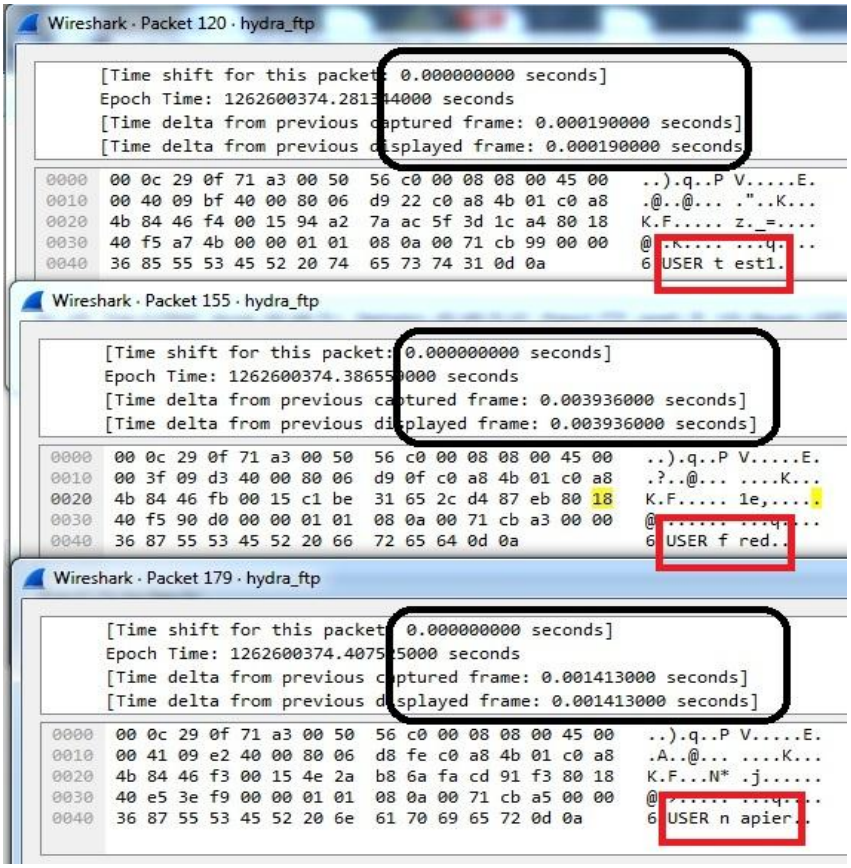
Onde **alert tcp any any** é de qualquer origem, **-> any 23** para qualquer destino em direção a porta 23. A definição **(msg:"Possivel ataque Telnet"** é a mensagem no qual o Snort irá mostrar na detecção de alguma intrusão. Para definir o valor de ACK é usado **flow:to_server; flags: A; ack: 78** 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 |.

4.3.2 Desenvolvimento 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. Na Figura 13, foi filtrado o trafego para o serviço FTP para melhor analisar este tipo de intrusão.

Figura 13: Análise do tempo de cada tentativa de intrusão

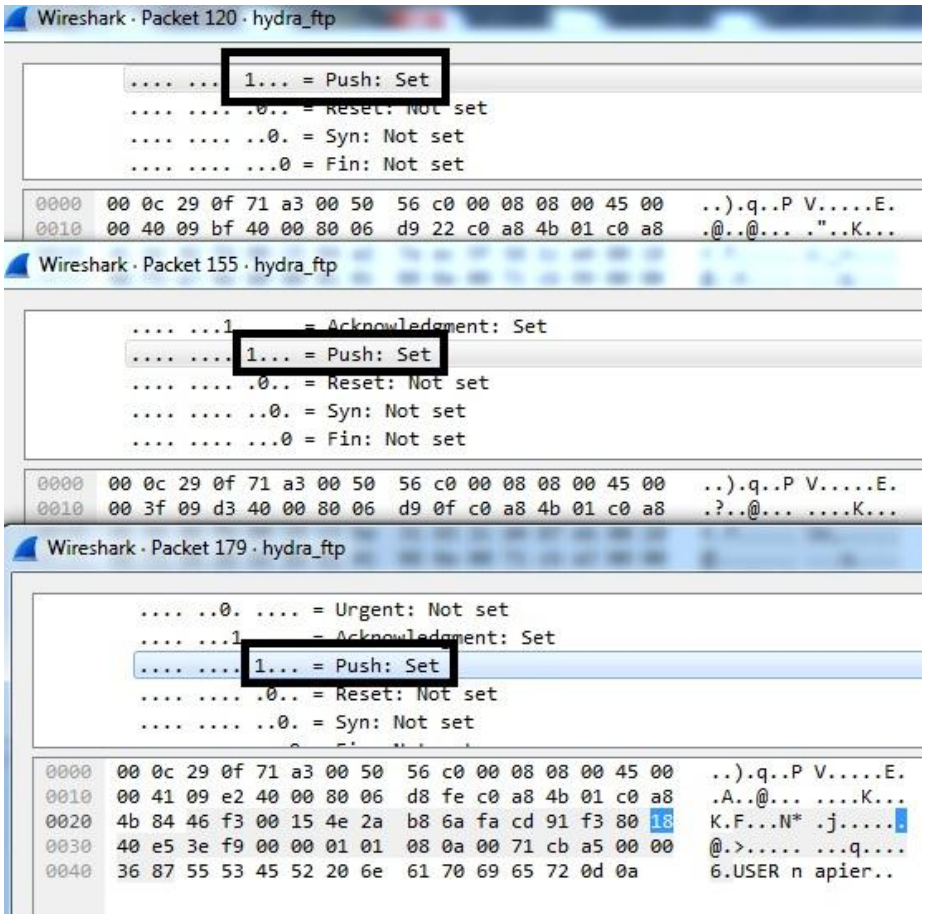


Como mostra a Figura 13, o tempo em que levou cada intrusão é inferior a 1 segundo, independente do usuário no qual esta tentando acesso no *host* alvo como mostra a seleção em quadrado. Como a ferramenta utiliza uma base de dados com possíveis chaves de acesso, terá muitas tentativas de conexão em um curto espaço de tempo, menor que 1 segundo como mostrado na Figura 13. Tendo como essas características, foi definida a primeira parte da regra,

onde caso aconteça 18 tentativas de conexão no intervalo de 1 segundo, o Snort deverá detectar.

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. Na Figura 14, se o bit na flag PSH estava em 1, e pode-se concluir que o mesmo esta com a função ativa.

Figura 14: Análise do flag Push



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.

Este capítulo apresentou as funcionalidades de cada etapa do processo, desde a integração entre o SDN a partir do controlador *Ryu* e o IDS Snort até a etapa de coleta e análise do tráfego. Detalhou ainda, todas as operações fundamentais para a criação das regras no Snort. Os testes realizados e a metodologia aplicada para validar este trabalho são apresentados no Capítulo 5.

5. 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.

5.1 Metodologia

Nesta seção é apresentado o ambiente desenvolvido para realização dos testes, os dispositivos utilizados e tecnologias aplicadas neste trabalho. Com o objetivo de validar este projeto foi elaborado o seguinte cenário: como dispositivo principal do projeto, onde foram criadas a máquinas virtuais, foi utilizado um computador tipo PC, com processador Intel Core i3-2120 de 3.30GHz e 6Gb de memória RAM, utilizando um disco com capacidade de até 1Tb. Como sistema operacional, foi utilizado o Windows 7 Professional, onde para a virtualização das máquinas virtuais foi empregado a versão 12.1.1 do VMWare.

Na primeira máquina virtual denominada “Controlador SDN”, responsável pelo ambiente onde estão o controlador *Ryu*, o emulador Mininet e o Hydra, foi criado uma máquina virtual com as seguintes especificações: 1 processador core virtual, 2Gb de memória RAM e utilizando um disco virtual com capacidade de 20Gb. Utiliza como sistema operacional Ubuntu na versão 14.04, por questão de estabilidade do sistema. Dentro desta máquina virtual, para emular o SDN, foi utilizado a versão 2.2.0 do Mininet, Também foi utilizado a ferramenta THC Hydra na versão 7.5, onde efetuou os ataques dentro do ambiente para análise do tráfego.

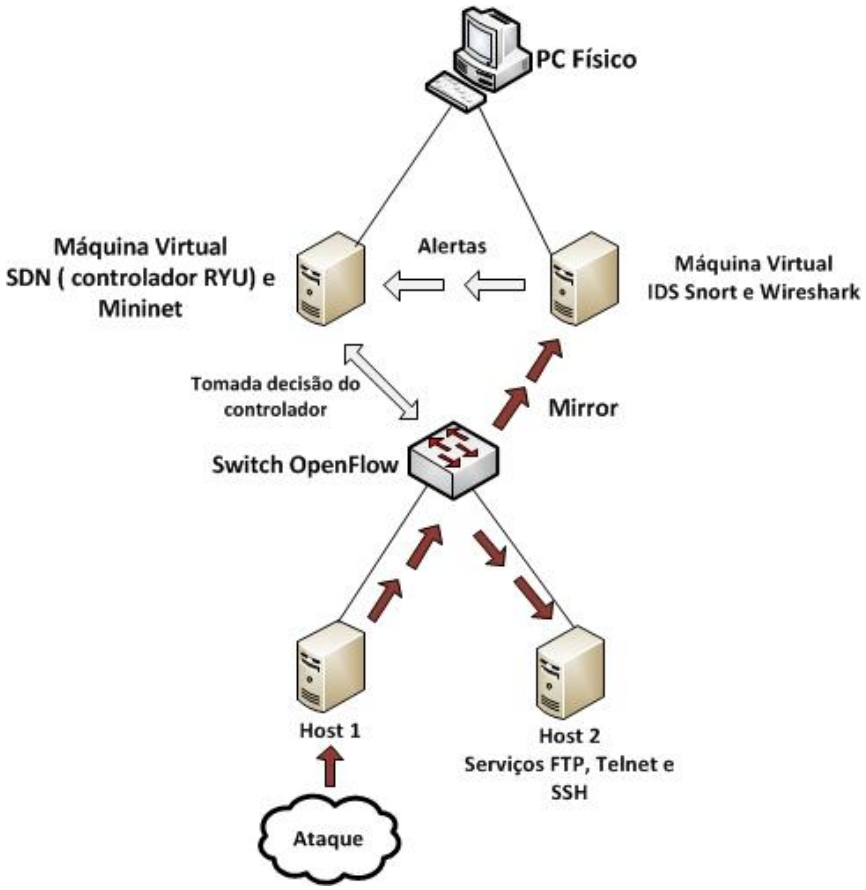
Na segunda máquina virtual denominada “Analisador Snort”, responsável pelo ambiente onde estão o IDS Snort e o analisador de tráfego *Wireshark* foi criado uma máquina virtual com as seguintes especificações: 2 processadores core virtual, 2Gb de memória RAM e utilizando um disco virtual com capacidade de 40Gb. Utiliza como sistema operacional Ubuntu na versão 14.04, por questão de estabilidade do sistema.

Dentro desta máquina virtual, para a criação das regras e identificação de intrusão, foi utilizada a versão 2.9.6.0 do Snort. Para a validação do trabalho, não foram utilizadas as regras disponibilizadas pela comunidade do projeto para que não interferisse nos resultados finais deste trabalho. Ainda nesta máquina virtual, para a análise do tráfego e auxilio na criação das regras, foi utilizada a ferramenta *Wireshark* na versão 1.12.3.

5.2 Testes Realizados

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 15.

Figura 15: Ambiente desenvolvido para validação e resultados



O ambiente ilustrado pela Figura 15 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.

Para validar as 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. Na ferramenta, é necessário também colocar o usuário, o tipo de serviço e o IP de destino no qual pretende atacar. 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.

Os resultados obtidos através dos testes realizados são abordados e ilustrados na seção 5.3, onde são detalhadas estas informações.

5.3 Resultados Obtidos

Os resultados obtidos através da análise da utilização da rede foram os esperados. O sistema Snort conseguiu, através das regras elaboradas, identificar os testes realizados e enviar os alertas ao controlador *Ryu* através da integração entre essas duas ferramentas Esta seção foi subdividida em duas partes: os resultados obtidos através de testes realizados utilizando ataques de intrusão com a ferramenta Hydra esta na subseção 5.3.1. Já na subseção 5.3.2 esta os resultados obtidos através de testes utilizando o *dataset* Darpa 99.

5.3.1 Utilização Ferramenta Hydra

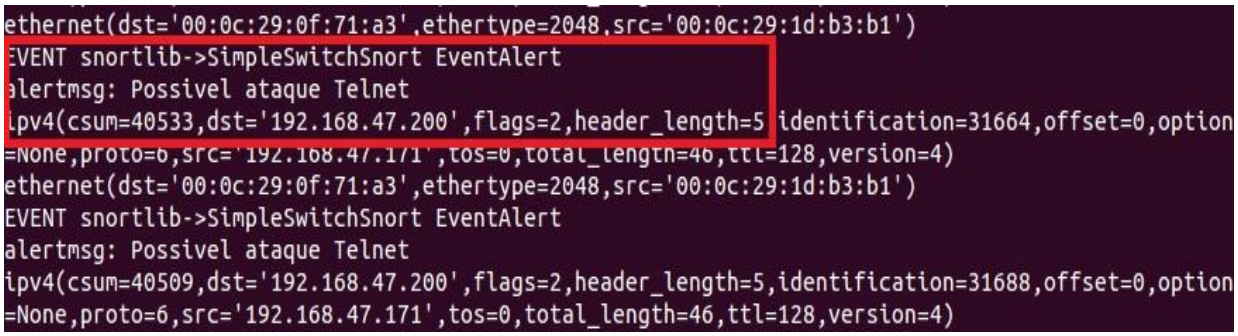
No que diz respeito à simulação de ataque ao serviço Telnet, conforme a Tabela 3, a ferramenta gerou 10995 mil pacotes, nos quais 3510 mil pacotes eram de requisições ao servidor onde se encontra o serviço. O Snort foi capaz de identificar dentro destes ataques, 34% destas requisições.

Tabela 3: Experimento com ataques gerados pelo Hydra contra Telnet

	Número de pacotes gerados	Número ataques para Telnet	Identificação de ataque
Snort	10995 mil pacotes	3510 mil pacotes	34%

Mas apesar de acertos, o Snort não detectou 66% dos ataques que foram lançados pela ferramenta, assim gerando um alto número de falsos negativos. Contudo, a regra foi desenvolvida para que registrasse ataques contendo somente o usuário root, assim ocasionando este índice de falsos negativos. A Figura 16 mostra o controlador *Ryu* recebendo os alertas gerados pelo Snort, assim comprovando a integração entre as ferramentas.

Figura 16: Alertas recebidos no Controlador para ataque contra Telnet



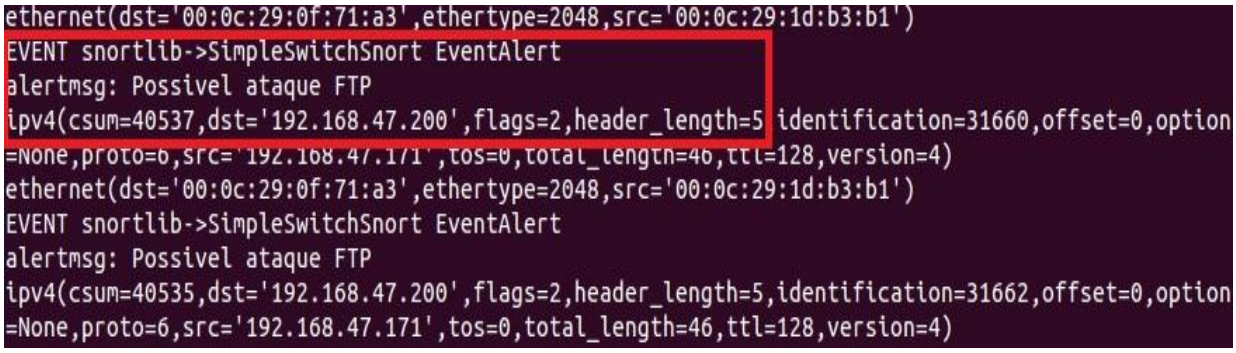
No caso de ataque contra o serviço FTP utilizando a ferramenta Hydra, conforme a Tabela 4, a mesma gerou 6140 mil pacotes, nos quais 3680 mil pacotes eram de requisições ao servidor onde se encontra o serviço. O Snort foi capaz de identificar dentro destes ataques, 91% destas requisições.

Tabela 4: Experimento com ataques gerados pelo Hydra contra FTP

	Número de pacotes gerados	Número de ataques para FTP	Identificação de ataques
Snort	6140 mil pacotes	3680 mil pacotes	91%

Mas apesar de um alto número de acertos, o Snort não detectou 8% dos ataques que foram lançados pela ferramenta, assim gerando um baixo número de falsos negativos. A Figura 17 mostra o controlador *Ryu* recebendo os alertas gerados pelo Snort, assim comprovando a integração entre as ferramentas.

Figura 17: Alertas recebidos no Controlador para ataque contra FTP



5.3.2 Utilização *Dataset* Darpa 99

Primeiramente, analisamos as regras dentro do primeiro cenário, onde está o tráfego da rede sem qualquer ataque, mas com requisição para os serviços no qual foram criadas as regras neste trabalho. Neste cenário, contamos com 9642 mil pacotes retirados do *dataset* para efetuar os testes.

No caso da regra desenvolvida contra ataque ao serviço Telnet, conforme a Tabela 5, dos pacotes extraídos do *dataset*, 3942 mil pacotes é de requisição ao serviço onde não são ataques maliciosos. O Snort identificou dentro deste cenário, 13% destas requisições, assim sendo falsos positivos gerados pela regra no qual foi proposta.

Tabela 5: Experimento utilizando Darpa 99 com tráfego normal contra Telnet

	Número de pacotes extraido Dataset Darpa99	**Número pacotes não maliciosos para Telnet**	**Alerta da regra (Falso Positivo)**
Snort	9642 mil pacotes	3942 mil pacotes	13%

No que diz respeito à regra contra ataque ao serviço FTP, como podemos ver na Tabela 6, dos pacotes extraídos do *dataset*, 1697 mil pacotes são tentativas de conexão ao serviço onde não são considerados ataques. O Snort identificou dentro deste cenário, 9% destas requisições, assim sendo falsos positivos gerados pela regra no qual foi proposta.

Tabela 6: Experimento utilizando Darpa 99 com tráfego normal contra FTP

	Número de pacotes extraido Dataset Darpa99	**Número pacotes não maliciosos para Telnet**	**Alerta da regra (Falso Positivo)**
Snort	9642 mil pacotes	1697 mil pacotes	9%

Neste cenário, podemos concluir que apesar das regras desenvolvidas estarem identificando tráfego normal como sendo um ataque, o percentual é baixo, pois pelo número de tráfego que foi colhido dentro do *dataset* simulando um trafego real sem qualquer trafego malicioso. Com o aperfeiçoamento destas regras pode-se reduzir ainda mais este percentual.

No segundo ambiente, está o tráfego da rede com tráfego malicioso contra os serviços propostos. Neste arquivo, contamos com 11719 mil pacotes retirados do *dataset* para efetuar os testes.

Analisamos primeiramente a regra desenvolvida contra ataque ao serviço Telnet, conforme a Tabela 7, dos pacotes extraídos do *dataset*, 7934 mil pacotes é de requisição ao serviço onde são ataques maliciosos. O Snort identificou dentro deste cenário, 57% destas requisições, como sendo ataques contra este serviço.

Tabela 7: Experimento utilizando Darpa 99 com tráfego malicioso para Telnet

	Número de pacotes extraido Dataset Darpa99	**Número pacotes maliciosos para Telnet**	**Identificação de ataque**
Snort	11719 mil pacotes	7934 mil pacotes	57%

Após, analisamos a regra desenvolvida contra ataque ao serviço FTP, conforme a Tabela 8, dos pacotes extraídos do *dataset*, 2291 mil pacotes é de requisição ao serviço onde aconteceram ataques maliciosos. O Snort identificou dentro deste cenário, 83% destas requisições, como sendo ataques contra este serviço.

Tabela 8: Experimento utilizando Darpa 99 com tráfego malicioso para FTP

	Número de pacotes extraido Dataset Darpa99	**Número pacotes maliciosos para FTP**	**Identificação de ataque**
Snort	11719 mil pacotes	2291 mil pacotes	83%

Neste cenário, podemos concluir que o índice de acerto da regra para FTP é maior que o para Telnet, visto que existe uma taxa alta de falso negativo na primeira regra, contudo esta mesma regra foi direcionada somente para tentativa de acesso com o usuário root.

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.

6. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

As vulnerabilidades que são representadas pelo grande avanço da Internet e por novas tecnologias, são exploradas por usuários mal intencionados, onde fazem uso de diversas técnicas para explorar essas vulnerabilidades mediante ataques de intrusão. Os métodos convencionais de segurança que são baseados em barreiras de proteção e mecanismos de controle de acesso acabam ficando obsoletos com o rápido avanço destas ameaças.

Neste trabalho de conclusão foi apresentado o desenvolvimento de regras no Snort contra ataques de intrusão em direção aos protocolos FTP e Telnet dentro de um ambiente SDN emulado na ferramenta Mininet. Para a implementação destas regras, foram estudados alguns métodos propostos na literatura para a detecção de intrusão. Após a definição desses métodos, foram efetuados ataques através de uma ferramenta de intrusão para analisar alguns padrões destes mesmos ataques. Através dessas analises, foram criadas a regras de detecção de intrusão no Snort. Além disso, foi realizada a integração entre o controlador *Ryu* e o IDS Snort para que a cada detecção gerado no IDS, este por sua vez envie alertas ao controlador no SDN.

Após esta etapa de implementação das regras e integração entre as ferramentas, a metodologia de testes foi aplicada utilizando a mesma ferramenta de intrusão e também com o uso de um *dataset* com dados de uma rede normal e com tráfego malicioso. 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 a este projeto, pode-se mencionar a possibilidade de melhoria nas regras criadas, onde pode-se focar na tentativa de intrusão através de múltiplos usuários, sem definição de um usuário padrão dentro desta regra. Isso é importante, pois o atacante pode conter uma extensa base que contenham diversos usuários para tentativa de acesso ao sistema.

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.

Por fim, 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.

REFERÊNCIAS

AL-JARRAH, O.; ARAFAT, A. *Network Intrusion Detection System using attack behavior classification. In:* information and communication systems (ICICS), 2014 5th international conference, 2014. p.1–6.

BHARDWAJ, Pawan K. *A+, Network+, Security+ Exams*. 1³ ed. Sebastopol: O´Reilly, 2007.

BHUYAN, M.; BHATTACHARYYA, D.; KALITA, J. *Network Anomaly Detection: methods, systems and tools*. Communications Surveys Tutorials, IEEE, 2014. v.16, n.1, p.303–336.

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

BUL'AJOUL, W.; JAMES, A.; PANNU, M. *Network Intrusion Detection Systems in*

High-Speed Traffic in Computer Networks. In: IEEE 10TH, 2015.

CHEN, P; CHEN, Y. *Implementation of SDN Based Network Intrusion Detection and Prevention System.* ICCST, 2015.

COUTO, Rodrigo de Souza. *Openflow por Rodrigo de Souza Couto.* 2010. Disponível em < http://repositorio.unisc.br/jspui/bitstream/11624/532/1/tc-gustavo2.pdf >. Acesso em: Março de 2016.

FAGUNDES, B. et al. *Snortik – Uma Integração do IDS Snort e o Sistema de Firewall do Mikrotik Routeros.* XIV ERRC, 2016.

FERREIRA, Fernando Nicolau Freitas. *Segurança da informação*. Rio de Janeiro: Ciência Moderna, 2003.

FOUNDATION, Open Networking. *Open Networking Foundation white papers***.** Disponível em: < https://www.opennetworking.org/sdn-resources/sdn-definition >. Acesso em: Agosto de 2016.

FU, Zhang. *Mitigating Distributed Denial-of-Service Attacks*: *Application-defense and Network-defense Methods*. Gothenburg, 2012. p.59-59

GUDE, N. et al. *Nox:* Towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 2008. 38(3):105–110.

KETI, F; ASKAR, S. *Emulation of Software Defined Networks Using Mininet in Different Simulation Environments*. Proceedings of the 2015 6th International Conference on Intelligent Systems, Modelling and Simulation. IEEE, 2015.

KIM, H; FEAMSTER, N. *Improving Network Management with Software Defined Networking.* Communications Magazine, IEEE, 2013.

KREUTZ, D. et al. *Software-Defined Networking:* a comprehensive survey. Computing Research Repository, v.abs, 2014.

LARA, A; RAMAMURTHY, B. *OpenSec:* A Framework for Implementing Security Policies using OpenFlow. CISSS, 2014.

LE, A. et al. *Flexible Network-based Intrusion Detection and Prevention System on Software-defined Networks*. ICACA, 2015.

LIPPMANN, R. et al. *The 1999 DARPA Off-Line Intrusion Detection Evaluation.* Draft of paper submitted to Computer Networks, In Press, 2000.

MATTOS, D. M. F.; FERRAZ, L. H. G.; DUARTE, O. B. *Um mecanismo para isolamento seguro de redes virtuais usando a abordagem hıbrida Xen e OpenFlow*. Em XIII SBSeg, 2013. p. 128–141.

MCKEOWN, N; ANDERSON, T. *OpenFlow:* Enabling Innovation in Campus Networks. ACM SIGCOMM, 2008.

MEHRA, Pritika. *A brief study and comparison of Snort and Bro Open Source Network Intrusion Detection Systems*. Ijarcce, 2012.

MININET.ORG. *Mininet:* An Instant Virtual Network on your Laptop (or other PC). 2016. Disponível em: < http://mininet.org >. Acesso em: Março de 2016.

NADEAU, T. D; GRAY, K. *SDN:* Software Defined Networks. ”O’Reilly Media, Inc.”, 2013.

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

NAKAMURA, E. T. *Segurança de Redes em Ambiente Corporativos*. Novatec Editora, São Paulo, 2007.

NOBRE, J.C.A. *Ameaças e Ataques aos Sistemas de Informação:* Prevenir e Antecipar: Cadernos UniFOA, Volta Redonda, 2007.

ONF. *Open Networking Foundation*, 2016. Disponível em: < www.opennetworking.org/sdn-resources/openflow >. Acesso em: Março de 2016.

REHMAN, Rafeeq. *Intrusion Detection with SNORT:* Advanced IDS Technique Using SNORT, Apache, MySQL, PHP and ACID. 1 ed. New Jersey: Prentice Hall, 2003.

RNP.br. *Rede Nacional de Ensino e Pesquisa*. Disponível em: < http://www.rnp.br >. Acesso em: Fevereiro de 2016.

ROTHENBERG, C. et al. *OpenFlow e redes definidas por software:* um novo paradigma de controle e inovação em redes de pacotes; Cad. CPqD Tecnologia, Campinas, 2011. v 7, n.1, p. 65-76.

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

SABAHI, F; MOVAGHAR, A. *Intrusion Detection:* A Survey. Systems and Networks Communications. The Third International Conference on Systems and Networks Communications, 2008.

SNORT. *The Snort Project*. Disponível em: <http://www.snort.org/>. Acesso em: Março de 2016.

THC.org. *The Hacker's Choice.* Disponível em < http://www.thc.org >. Acesso em: agosto de 2016.

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

XIE, M., Hu, J. and Slay, J. *Evaluating Host-Based Anomaly Detection Systems*: Application of the One-Class SVM Algorithm to ADFA-LD. Proceedings of the 11th IEEE International Conference on Fuzzy Systems and Knowledge Discovery, Xiamen, 2014. p.19-21.

XING, T.; HUANG, D.; XU, L. et al. *SnortFlow*: A OpenFlow-Based Intrusion Prevention System in Cloud Environment. In: *2nd GENI Research and Educational Experiment Workshop*, 2013. p. 89–92

ZAMAN, S.; KARRAY, F. *TCP/IP Model and Intrusion Detection Systems*. In: Advanced Information Networking and Applications Workshops, 2009. WAINA '09. INTERNATIONAL CONFERENCE ON, 2009. p.90–96.

YAN, J.; DONG, J. *VT-Mininet*: Virtual-time-enabled Mininet for Scalable and Accurate Software-Define Network Emulation. Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research. ACM, 2015.

WANG, Shie-Yuan. *Comparison of SDN OpenFlow network simulator and emulators:* EstiNet vs. Mininet. Computers and Communication (ISCC), IEEE Symposium on 2014.

WIRESHARK. *Wireshark.* Disponível em*:* < https://www.wireshark.org>. Acesso em: Outubro de 2016.

Santa Cruz do Sul, 24 de novembro de 2016.

_____

Thiago Oliveira Garcia
Aluno

_____

Professor Me. Charles Varlei Neu
Orientador