UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FÁBIO YU NAGAHAMA

IPSFlow: Um *framework* para Sistema de Prevenção de Intrusão baseado em Redes Definidas por *Software*

DM 26 / 2013

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2013

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FÁBIO YU NAGAHAMA

IPSFlow: Um *framework* para Sistema de Prevenção de Intrusão baseado em Redes Definidas por *Software*

Dissertação submetida à Banca Examinadora do Programa de Pós-Graduação em Engenharia Elétrica da UFPA para a obtenção do Grau de Mestre em Engenharia Elétrica.

Prof. Dr. Eduardo Coelho Cerqueira
(ORIENTADOR – PPGEE/UFPA)

Prof. Dr. Antônio Jorge Gomes Abelém
(CO-ORIENTADOR – PPGCC/UFPA)

UFPA / ITEC / PPGEE

Campus Universitário do Guamá

Belém-Pará-Brasil

2013

Dados Internacionais de Catalogação-na-Publicação (CIP)

Nagahama, Fábio Yu, 1979-
Ipsflow: um framework para sistema de prevenção de intruso baseado em redes definidas por software / Fábio Yu Nagahama. - 2013.

Orientador: Eduardo Coelho Cerqueira;
Coorientador: Antônio Jorge Gomes Abelém.
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2013.

1. Redes de computadores - medidas de segurança. 2. Recurso de redes de computadores. 3. Tecnologia de rede de computador. I. Título.

CDD 22. ed. 005.8

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

IPSFLOW: UM *FRAMEWORK* PARA SISTEMA DE PREVENÇÃO DE INTRUSÃO BASEADO EM REDES DEFINIDAS POR *SOFTWARE*

AUTOR: FÁBIO YU NAGAHAMA

DISSERTAÇÃO DE MESTRADO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARÁ E JULGADA ADEQUADA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE COMPUTAÇÃO APLICADA.

APROVADA EM 09 / 10 / 2013

BANCA EXAMINADORA:

Prof. Dr. Eduardo Coelho Cerqueira
(Orientador – PPGEE/UFPA)

Prof. Dr. Antônio Jorge Gomes Abelém
(Membro – PPGCC/UFPA)

Prof. Dr. Roberto Samarone dos Santos Araújo
(Membro – PPGCC/UFPA)

Prof. Dr. Raimundo Viégas Junior
(Membro – PPGCC/UFPA)

VISTO:

Prof. Dr. Evaldo Gonçalves Pelaes
(Coordenador do PPGEE/ITEC/UFPA)

DEDICATÓRIA

Dedico este trabalho de Dissertação de Mestrado a meu pai Masahiro Nagahama (*in memoriam*), que dentre os vários valores que nos ensinou, sempre nos mostrou que a educação deve ser uma das prioridades de nossas vidas e fez de tudo para proporcioná-la aos seus filhos. Partiu para o mundo espiritual antes desta conquista, que lamento muito por não ter proporcionado em tempo.

Também dedico à minha esposa Milena, e principalmente aos meus filhos Fernando e Suzana, que acabaram sofrendo com meus momentos de preocupação e *stress*, mesmo sem entender os motivos de tal comportamento por serem pequenos demais.

AGRADECIMENTOS

Primeiramente agradeço a Deus e meus pais, Shizuko Nagahama e Masahiro Nagahama (*in memoriam*), por ter conseguido chegar até aqui. Agradeço muito minha esposa, Milena, que sempre esteve ao meu lado durante toda essa jornada, me dando apoio total, para qualquer decisão que pudesse tomar no final. Muito obrigado aos meus filhos, Fernando e Suzana, que sempre foram e serão meus impulsionadores para alcançar novas conquistas.

Um obrigado muito especial para o Prof. Dr. Antônio Abelém, que vem me orientando de longas datas e nunca desistiu. Agradeço ao Prof. Dr. Eduardo Cerqueira pela sua acolhida no PPGEE, por suas cobranças e incentivo transmitidos através de suas palestras e reuniões.

À amiga Priscilla Lanne, que sempre incentivou a concluir este trabalho.

Um agradecimento muito especial à amiga Elisângela Aguiar, pelo seu apoio, suas cobranças, sugestões, dicas e as valiosas revisões expressas, que sem as quais possivelmente não conseguiria conquistar metade do que alcancei nesta empreitada.

Muito obrigado a Hugo Toda, Airton Ishimori, Fernando Farias e João Salvatti pelas suas ricas contribuições e consultorias, sem pelas quais não conseguiria finalizar este trabalho. Agradeço também a Patrício Cordeiro e Robson Gonçalves por suas contribuições.

Aos membros da banca que aceitaram avaliar este trabalho e compartilhar seus conhecimentos através de críticas que serão utilizadas para o meu aprendizado e melhoria deste trabalho.

Obrigado Socorro Palheta por toda a sua ajuda e assessoria nos assuntos relacionados à secretaria do Programa.

A todos outros colegas que não lembro no momento, mas que incentivaram e torceram por esta conquista.

E por fim, não posso deixar de agradecer aos que duvidaram de mim, pois estes foram um dos maiores fortalecedores para a conclusão deste mestrado.

RESUMO

Os Sistemas de Detecção e Prevenção de Intrusão (*Intrusion Detection Systems* – IDS e *Intrusion Prevention Systems* - IPS) são ferramentas bastante conhecidas e bem consagradas no mundo da segurança da informação. Porém, a falta de integração com os equipamentos de rede como *switches* e roteadores acaba limitando a atuação destas ferramentas e exige um bom dimensionamento de recursos de *hardware* como processamento, memória e *interfaces* de rede de alta velocidade, utilizados para implementá-las. Diante de diversas limitações deparadas por pesquisadores e administradores de redes, surgiu o conceito de Rede Definida por *Software* (*Software Defined Network* – SDN), que ao separar os planos de controle e de dados, permite adaptar o funcionamento da rede de acordo com as necessidades de cada um. Desta forma, devido à padronização e flexibilidade propostas pelas SDNs, e das limitações apresentadas dos IPSs, esta dissertação de mestrado propõe o IPSFlow, um *framework* que utiliza uma rede baseada na arquitetura SDN e o protocolo OpenFlow para a criação de um IPS com ampla cobertura e que permite bloquear um tráfego caracterizado pelos IDS(s) como malicioso no equipamento mais próximo da origem. Para validar o *framework*, experimentos no ambiente virtual Mininet foram realizados utilizando-se o Snort como IDS para analisar tráfego de varredura (*scan*) gerado pelo Nmap de um *host* ao outro. Os resultados coletados apresentam que o IPSFlow funcionou conforme planejado ao efetuar o bloqueio de 85% do tráfego de varredura.

PALAVRAS-CHAVES: Segurança, Sistema de Detecção de Intrusão, Sistema de Prevenção de Intrusão, Redes Definidas por Software, *Software Defined Networks*, SDNs, IDS, IPS, OpenFlow.

ABSTRACT

Intrusion Detection and Prevention Systems (IDSs/IPSs) are well known tools and well enshrined in the world of information security. However, the lack of integration with network equipment, such as switches and routers, tends to limit the performance of these tools leads to require a proper dimensioning of hardware resources such as processor, memory and high-speed network interfaces used to implement them. Faced with several limitations encountered by researchers and network administrators, the concept of Software Defined Network (SDN), that separates the data and control planes, emerged allowing to adapt the operation of the network according to their needs. Thus, due to standardization and flexibility offered by SDNs, and the limitations presented by IDSs, this dissertation proposes IPSFlow, a framework that uses a network based on the SDN architecture, and the OpenFlow protocol, to create an IPS with wide coverage that blocks a malicious traffic in the equipment closer to the origin. To validate the framework, experiments in the virtual Mininet environment were conducted using Snort as IDS to analyze scanning traffic generated by Nmap from a host to another. The results show that the IPSFlow worked as planned by blocking almost 85% of scanning traffic.

KEYWORDS: Security, Intrusion Detection System, Intrusion Prevention System, Software Defined Network, SDN, IDS, IPS, OpenFlow.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

ACL	Access Control List
ALARMS	A Language for Arbitrary Redirection for Measuring and Security
API	Application Programming Interface
ARP	Address Resolution Protocol
BSD	Berkeley Software Distribution
CERT	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança
DDoS	Distributed Denial of Service
DNS	Domain Name System
GERCOM	Grupo de Estudo em Redes de Computadores e Comunicação Multimídia
HIDS	Host Based Intrusion Detection Systems
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
ISP	Internet Service Provider
JSON	JavaScript Object Notation
MAC	Media Access Control
NAT	Network Address Translation
NIDS	Network Based Intrusion Detection Systems
PCA	Principal Component Analysis
REST	Representational State Transfer
SBSeg	Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais
SDN	Software Defined Network
SNMP	Simple Network Management Protocol
SOHO	Small Office/Home Office
SOM	Self Organizing Maps
SSL	Secure Sockets Layer
TCAM	Ternary Content Addressable Memory

TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UFPA	Universidade Federal do Pará
URI	Uniform Resource Identifier
VLAN	Virtual Local Area Network
WPEIF	Workshop de Pesquisa Experimental da Internet do Futuro
XML	Extensible Markup Language

SUMÁRIO

1 Introdução

Este trabalho apresenta o *framework* IPSFlow que permite melhor integrar os sistemas de detecção de intrusão (*Intrusion Detection System* - IDS) e sistemas de prevenção de intrusão (*Intrusion Prevention System* – IPS) aos dispositivos de rede.

Neste Capítulo, apresentam-se as motivações deste trabalho, evidenciando de modo geral, como o ramo da segurança da informação pode beneficiar-se de uma rede flexível e programável. Em seguida, os trabalhos relacionados que incentivaram a criação do IPSFlow e os objetivos do trabalho serão apresentados. Por fim, a organização dos demais Capítulos será abordada.

1.1 Motivação

A cada dia, novos ataques ou mesmo variações dos ataques previamente identificados surgem e são lançados na rede em busca de vulnerabilidades. Segundo o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br), em 2011, foram reportados quase 400.000 incidentes, em 2012 este número ultrapassou 460.000 e até junho de 2013 este número passou dos 173.000 incidentes (CERT, 2013). Desta forma, ferramentas como IDSs e IPSs, que permitam a identificação e o tratamento automatizado de incidentes de segurança na rede, tornam-se cada vez mais importantes na atividade realizada por um administrador de segurança da informação.

Porém, estas ferramentas possuem requisitos específicos para que tenham uma atuação eficiente. Elas devem possuir uma ampla cobertura e bom desempenho para não passar a falsa sensação de proteção ou degradar o desempenho da rede. Aspectos como flexibilidade e administração simplificada também devem ser considerados na escolha e instalação destes sensores para não resultar que a empresa/instituição fique na dependência de um único fabricante ou fornecedor da solução.

Os equipamentos de redes como *switches* e roteadores mais modernos são dotados normalmente de um plano de dados e um plano de controle. O plano de dados é o elemento responsável por todo o encaminhamento do tráfego de rede entre as portas do equipamento e o plano de controle é o responsável por definir como o encaminhamento deve ser feito no plano de dados. Sendo assim, no modelo atual, qualquer definição e modificação do plano de controle só pode ser realizada pelo fabricante do equipamento, obrigando os administradores e usuários a ficarem dependentes de seus fornecedores no que se refere a novas funcionalidades implementadas e disponibilizadas nos equipamentos de rede.

Com o objetivo de prover maior autonomia a pesquisadores e administradores de rede, permitindo-lhes definir como a mesma deve funcionar, o conceito de Rede Definida por *Software* (*Software Defined Network* – SDN) foi elaborado e apresentado para a comunidade de redes de computadores. Em uma SDN, o plano de controle é separado do plano de dados e é delegado a um elemento externo chamado de Controlador, permitindo que pesquisadores, administradores e até usuários programem o comportamento da rede. Nesta arquitetura, a definição do funcionamento interno (plano de dados) dos equipamentos de rede como *switches* e roteadores continua sob a responsabilidade de seus fabricantes. Desta forma, reduz-se a dependência do proprietário com o fornecedor do equipamento.

Neste contexto, o OpenFlow, proposto por McKeown *et al*. (2008), é um *framework* aberto nos moldes das SDNs, que através do protocolo OpenFlow disponibiliza *interfaces* de programação para construção de Controladores de rede. Em uma rede OpenFlow, o Controlador tem uma visão completa da rede e, assim, tem a capacidade de manipular os fluxos de dados logo que estes são recebidos em qualquer *switch* que implemente as funcionalidades contidas na especificação em Openflow (2011).

Desta forma, esta dissertação apresenta o IPSFlow, uma arquitetura de IPS que utiliza uma SDN e o protocolo OpenFlow para a construção de um IPS com ampla cobertura na rede e que permita captura seletiva e distribuída de tráfego nos *switches* para análise em um ou mais IDSs. No IPSFlow, de acordo com o resultado da análise feita pelos IDSs, o Controlador OpenFlow poderá bloquear o fluxo de forma automática no *switch* mais próximo da origem do tráfego, impedindo assim, que o tráfego malicioso circule livre pela rede.

1.2 Trabalhos Relacionados

O uso do OpenFlow, na área de segurança e monitoramento de redes, já foi abordado e apresentou bons resultados em pesquisas anteriores, como, por exemplo, nas pesquisas de Ballard *et al*. (2010), Braga *et al*. (2010), Mehdi *et al*. (2011) e Xing (2013).

No trabalho de Ballard *et al*. (2010), diante do aumento das taxas de transmissão das *interfaces* e da reduzida quantidade de portas espelho que os equipamentos de rede possuem, o OpenSAFE foi proposto como uma solução para redirecionamento de tráfego à altas taxas de transmissão (*line rate*) para propósitos de monitoramento. Este trabalho também propôs uma linguagem de alto nível chamada de ALARMS (*A Language for Arbitrary Redirection for Measuring and Security*) que tem como objetivo especificar fluxos e simplificar o gerenciamento e o monitoramento de rede.

Já em Braga *et al.* (2010), utilizando-se da característica de manipular fluxos de forma centralizada no Controlador, o OpenFlow foi usado em conjunto com uma rede neural artificial do tipo *Self Organizing Maps* (SOM) e foi proposto um método simplificado (*lightweight*) para detectar ataques do tipo Negação de Serviço Distribuído (*Distributed Denial of Service* - DDoS). A rede neural proposta neste trabalho toma como base as estatísticas coletadas dos *switches* OpenFlow em intervalos específicos e faz análises das médias como da quantidade de pacotes por fluxo, quantidade de *bytes* por fluxo, duração do fluxo, porcentagem de correspondência entre pares de fluxos, crescimento no número de fluxos simples e variação de portas utilizadas. Além de apresentar resultados de detecção semelhantes de outras propostas convencionais, este trabalho apresenta duas vantagens: a capacidade de manipular e detectar ataques DDoS em tráfego capturado de forma *online*; e a facilidade em manipular informações de fluxos dos dados, no qual em outras abordagens só seria possível através de modificações no *software* do *switch*.

Para Mehdi *et al*. (2011), a disseminação do acesso à Internet através da banda larga em residências e pequenos escritórios (*Small Office/Home Office* - SOHO) faz aumentar os riscos com a segurança do provedor de serviços de Internet (*Internet Service Provider* – ISP), uma vez que nestes tipos de clientes, a preocupação com a segurança da rede é mínima ou inexistente. Neste cenário, devido à variedade de dispositivos utilizados por seus clientes, os cuidados do ISP em segurança se concentram no núcleo de sua rede ao invés de atuar no ponto mais próximo da causa do problema. Desta forma, os autores propuseram e avaliaram o uso do OpenFlow na detecção de tráfegos anômalos nas redes de clientes de ISPs. O trabalho, apresentou que as SDNs, usando o OpenFlow e o Controlador NOX, permitem uma detecção de anomalias de forma flexível, altamente precisas e à altas taxas dentro da rede dos clientes do ISP.

Xing *et al.* (2013) apresentou o SnortFlow, uma proposta de IPS em ambiente de nuvem baseado na solução com Xen, utilizando *switches* OpenFlow para auxiliar na captura do tráfego. Embora atividades de contramedida tenham sido apresentadas para a reconfiguração da rede, estas não foram avaliadas. Os autores do trabalho elaboraram um protótipo no qual o agente SnortFlow foi instalado nos domínios Dom 0 e Dom U. Nesse ambiente, os melhores resultados foram obtidos no Dom 0, onde apresentou menor consumo de processador e menor descarte de pacotes ao atingir a taxa de recepção de 2500 pacotes por segundo. A avaliação do SnortFlow foi focada no desempenho, visto que os autores não detalharam as características do tráfego que estava sendo detectado pelo Snort e nem a análise que estava sendo feita sobre ele.

O uso de OpenFlow para propósitos de segurança mostrou-se bastante promissor nos trabalhos anteriores de Ballard (2010), Braga (2010), Mehdi (2011) e Xing (2013), os quais são sumarizados no Quadro 1. Porém, estes trabalhos se limitaram a monitorar ou apenas detectar e alertar a ocorrência de tráfegos maliciosos na rede e não se tem notícias do uso do resultado da detecção de intrusão para bloqueio automático do tráfego malicioso, tal como é proposto neste trabalho.

Trabalho	Monitoramento	Análise	Especifica linguagem	Tipo de ataque	Provê Bloqueio
Ballard et al. (2010)	S	N/A	S	N/A	N
Braga et al. (2010)	N	Controlador	N	DDoS	N
Mehdi et al (2011)	S	Controlador	N	N/A	N
Xing (2013)	N	Externo	N	N/A	N

Quadro 1– Quadro resumo dos trabalhos relacionados.

Desta forma, a proposta do IPSFlow é estender os trabalhos listados no Quadro 1 integrando os dispositivos de segurança como IDSs e IPSs com os equipamentos de rede, fazendo com que a visão centralizada e completa da rede permita realizar: a capturar do tráfego em praticamente qualquer posição da rede, encaminhar o tráfego para análise em um ou mais IDSs e reutilizar o resultado das análises para a reconfiguração automática das Tabelas de Fluxos dos *switches*.

1.3 Objetivos

Diante do apresentado, devido a flexibilidade permitida pela programabilidade da rede via OpenFlow, este trabalho tem como objetivo geral, apresentar o IPSFlow, um *framework* que permite a elaboração de IPSs com ampla cobertura na rede e que através de SDN possibilite a seleção e o bloqueio seletivo dos tráfegos de rede. Como desdobramento de tal objetivo, os seguintes objetivos secundários foram definidos:

- Descrever o *framework* IPSFlow;
- Descrever o ambiente experimental;
- Realizar experimentos e coletar dados para validar o funcionamento do *framework*;

- Analisar os resultados.

1.4 Organização do Trabalho

Além deste capítulo introdutório, o restante do trabalho está estruturado da seguinte forma:

No Capítulo 2, os conceitos de IDSs e IPSs convencionais são apresentados, bem como os desafios relacionados às suas atuações na rede.

Em seguida, no Capítulo 3, a arquitetura de SDNs e o protocolo OpenFlow são detalhados.

A proposta do IPSFlow é apresentada no Capítulo 4, bem como os detalhes de seu funcionamento e implementação.

O Capítulo 5 detalhará a metodologia utilizada, bem como o ambiente experimental criado para a coleta dos dados em cenários sem e com o uso do IPSFlow.

No Capítulo 6, os dados obtidos no Capítulo 5 são apresentados e analisados.

Por fim, as considerações finais e trabalhos futuros são discutidos no Capítulo 7.

2 Sistemas de Detecção e Prevenção de Intrusão

Os IDSs e IPSs são ferramentas consagradas e com importância reconhecida pela comunidade da segurança da informação. Assim, este Capítulo tem por finalidade apresentar os conceitos fundamentais relacionados com os IDSs e IPSs para um melhor entendimento de como estes sistemas funcionam e podem ser integrados aos demais dispositivos de rede. Após uma breve descrição e classificação dos IDSs/IPSs, este Capítulo apresenta os desafios inerentes a estas ferramentas ao serem utilizadas em uma rede com *switches* e roteadores convencionais.

2.1 Classificação dos IDSs

Os IDSs são ferramentas utilizadas para monitorar eventos e analisar sinais ou anomalias de intrusão em sistemas computacionais. Sua função primordial é detectar atividades maliciosas, ou anômalas, e alertar o administrador do sistema que estes eventos estão ocorrendo. Os IPSs são IDSs que além de alertar, possuem a finalidade de minimizar os impactos causados por sistemas comprometidos (MUKHOPADHYAY, 2011).

Os IDSs podem ser classificados basicamente de acordo com o local de atuação e com o tipo da técnica utilizada para análise conforme é apresentado na Figura 1. Quanto ao local de atuação, podem ser classificados como *Host Based Intrusion Detection Systems* (HIDS) e *Network Based Intrusion Detection Systems* (NIDS).

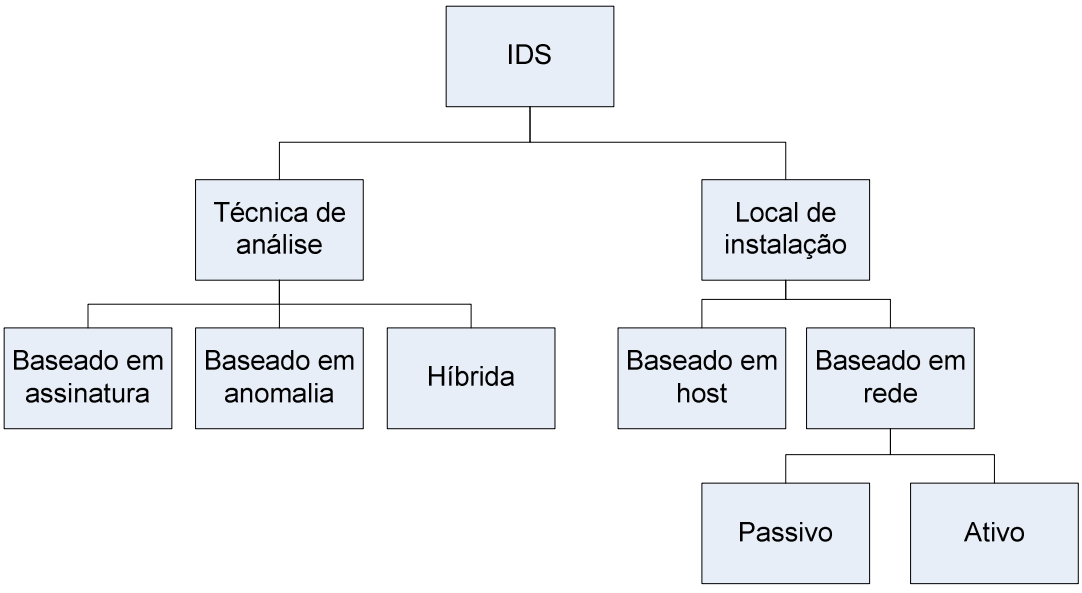


Figura 1 – Classificação de IDSs

Em um HIDS, o sensor é instalado em um *host* e sua atuação tem como objetivo analisar as informações contidas na própria máquina. Este tipo de IDS tem como objetivo analisar aspectos internos ao *host* como processos em execução, alterações não autorizadas em arquivos, instalação de executáveis, alterações de registros, verificação da integridade de executáveis, envio de tráfego excessivo na rede, entre outros.

Os HIDS possuem algumas vantagens como detectar ataques em eventos locais que não poderiam ser detectados pela rede; analisar dados criptografados e não serem limitados pelo uso de *switches* na rede sem o recurso de espelhamento de portas. Por outro lado, um HIDS possui algumas desvantagens como, por exemplo, a difícil instalação e manutenção; ser detectável pelo invasor e estar propenso a ataques; limitada detecção de ataques de rede e sua execução pode interferir no desempenho e funcionamento do próprio *host* onde está instalado (SCARFONE, 2007).

Já em um NIDS, o sensor é instalado na rede e sua(s) *interface*(s) de rede atua(m) em um modo especial denominado de "modo promíscuo", onde passa a ter a capacidade de capturar o tráfego de rede em tempo real e permite analisar os pacotes que estão trafegando na rede, mesmo estes não sendo destinados para o próprio sensor.

De acordo com o local de instalação, um NIDS ainda pode ser classificado como ativo ou passivo. No modo ativo, o NIDS é instalado em um ponto para interceptar o tráfego de rede, de forma semelhante a uma ponte (*bridge*), e analisar o tráfego passante. Desta forma, os NIDSs ativos possuem a capacidade de bloquear um tráfego caracterizado como malicioso ou indevido, sendo esta uma das principais vantagens deste tipo de IDS. Porém, um subdimensionamento do *hardware* utilizado pode adicionar retardos excessivos aos pacotes, podendo degradar o desempenho da rede.

No modo passivo, o NIDS analisa apenas cópias dos pacotes da rede que atravessam o *switch* ou *hub* onde este está conectado. Devido à heterogeneidade de fabricantes e modelos de equipamentos utilizados em uma rede, sem uma integração com outro equipamento de rede, que normalmente devem ser da mesma solução e/ou fabricante, a ação de um NIDS passivo fica limitada a apenas notificar a detecção de um tráfego malicioso.

Os NIDS possuem algumas vantagens como serem transparentes para o atacante; serem de fácil implantação; não interferirem no desempenho dos *hosts* e serem independentes de plataforma. Por outro lado, possuem desvantagens como: dificuldades em tratar dados de redes de alta velocidade; adição de retardos nos pacotes, quando instalados no modo ativo; possuir cobertura limitada e a dificuldade em manipular dados criptografados (SCARFONE, 2007).

Quanto à técnica de detecção utilizada, um IDS pode ser classificado como baseado em assinaturas ou em anomalias. Os IDSs baseados em assinaturas fazem a análise utilizando-se de uma base de assinaturas de ataques com padrões previamente conhecidos e catalogados. Tem como vantagem o pouco consumo de recursos e o rápido processamento. Porém, tem como desvantagem: exigir constante atualização da base de assinaturas; ter um alto índice de falsos positivos e exigir um bom nível de conhecimento na geração da base.

Os IDSs baseados em anomalias comparam o comportamento atual da rede com o comportamento previamente registrado como normal na fase de aprendizagem, e alertam quando ocorre um desvio nos padrões de tráfego de rede. Tem como vantagem poder detectar novos ataques através do desvio de comportamento sem a necessidade de conhecer detalhadamente a intrusão. Porém, tem como desvantagem: a possibilidade de gerar grande número de falsos alertas em decorrência de modificação no comportamento do *host* ou da rede, mesmo não se tratando de um tráfego malicioso; e o extensivo processo para a geração do perfil de comportamento considerado normal.

Uma terceira opção de IDSs é a chamada híbrida, que combina diferentes classificadores e vem sendo estudada, apresentando bons índices de detecção e baixa taxa de falsos positivos, como feito por Panda *et al.* (2012), que avaliou a combinação das técnicas *Decision Trees (DT)*, *Principal Component Analysis (PCA)*, *Stochastic variant of Piramol estimated subgradient solver in SVM* (SPegasos), *Ensembles of Balanced Nested Dichotomies for Multi-class Problems* (END), *Random Forest* e *Grading*.

Em uma rede, devido à grande variedade de dispositivos na rede (*notebooks*, *desktop*, celulares, *tablets* etc.) e sistemas operacionais (Linux, Windows, MAC OS etc.), o NIDS se torna uma opção mais utilizada devido a administração simplificada do ambiente quando comparada com os HIDS.

2.2 Desafios em Sistemas de Detecção e Prevenção de Intrusão na Rede

De acordo com o local de instalação de um sensor, este pode atuar como IDS ou IPS e possuir coberturas distintas. A Figura 2 apresenta uma típica rede composta por um *switch* de núcleo que interliga roteadores, servidores e as demais redes através de *switches* de distribuição ou acesso. As posições identificadas com as letras A, B e C representam alguns locais onde um IPS ou IDS pode ser instalado e as linhas tracejadas identificadas com os números 1, 2 e 3 são exemplos de alguns tipos de tráfego que uma estação interna à rede pode iniciar.

Na posição A, o IPS atua no modo ativo e só consegue capturar e analisar o tráfego passante (*i.e.*, fluxo 1), não tendo ações sobre os demais tráfegos confinados dentro da rede (i.e., fluxos 2 e 3). Para capturar os fluxos dos tipos 2 e 3, o IPS pode ser conectado ao *switch* central (posição B) ou nos *switches* de distribuição (posição C) e receber uma cópia do tráfego através do espelhamento de portas (modo passivo). Porém, uma vez que apenas uma cópia do tráfego é recebida, o bloqueio de tráfego malicioso torna-se inviável e sua atuação fica restrita a apenas gerar alertas de forma semelhante a um IDS.

Para que um tráfego seja bloqueado nas posições B e C, o IPS necessitaria atuar em conjunto com um equipamento capaz de bloquear o tráfego malicioso e esta é uma situação rara de acontecer devido à variedade de fabricantes e modelos de equipamentos utilizados nas redes.

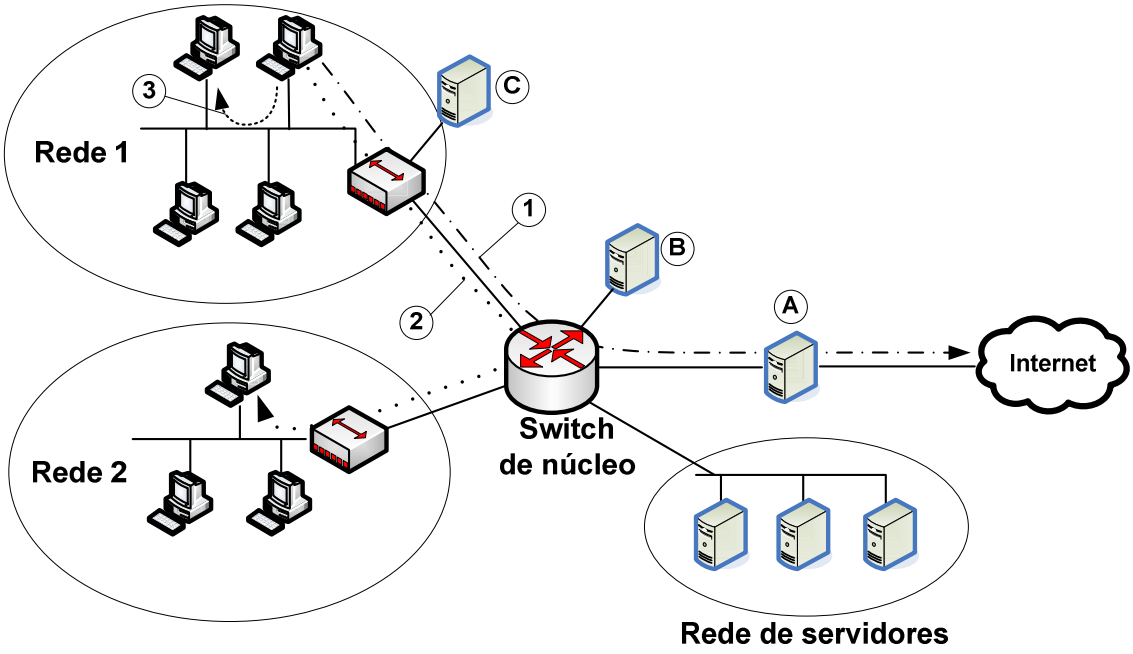


Figura 2 - Visão geral de uma rede com IPS/IDS.

Em quaisquer das posições (A, B ou C), não é comum que o tipo de tráfego a ser encaminhado aos IDSs e IPSs seja filtrado previamente. Assim, o *hardware* destas ferramentas deve ser devidamente dimensionado para que as etapas de captura, análise e eventual reencaminhamento do tráfego não adicionem retardos excessivos aos pacotes (no caso do IPS) ou para que todos os pacotes copiados sejam recebidos e analisados adequadamente para uma avaliação mais apurada (no caso do IDS).

Balancear a carga adicionando mais IPSs e IDSs pode evitar uma eventual necessidade de superdimensionamento do *hardware*. Porém, a adição de IPSs ativos pode acarretar na

adição de mais retardo nos pacotes. Para o caso de IDSs passivos, a limitação passa a ser dos *switches*, visto que nem todos são capazes de realizar o espelhamento de um número elevado de portas (BALLARD, 2010). Além disso, pelos mesmos motivos apresentados, o uso de diferentes sensores com diferentes técnicas de análise torna-se inviável.

Distribuir IPSs ativos em cada conexão ao *switch* central pode restringir a atuação de um ataque, mas a sincronização e administração de vários IPSs torna esta opção bastante onerosa e consequentemente inviável. As soluções de IPSs ativos e distribuídos de grandes fabricantes comerciais facilitam a administração, mas implicam na dependência de seus clientes com seus fornecedores.

Nesses casos, a adição de módulos ou mesmo a atualização do ambiente tendem a ser restritos apenas ao fornecedor, inviabilizando qualquer tipo de personalização por parte do administrador da rede. Mesmo em soluções com *softwares* livres, questões como desempenho, administração onerosa e sincronização de informação tornam-se um problema.

3 Redes Definidas por *Software*

O conceito de SDN surgiu da percepção que pesquisadores e administradores de rede tiveram ao se depararem com limitações que a arquitetura de construção de comutadores convencionais impunha à inovação e flexibilidade no funcionamento da rede.

Neste Capítulo, serão apresentados os conceitos acerca das SDNs, iniciando-se com um comparativo deste novo paradigma com o modelo tradicional de construção de comutadores. Em seguida, o OpenFlow é detalhado, onde serão apresentados os seus componentes e o seu funcionamento.

3.1 Desacoplamento dos Planos de Controle e de Dados

Nos equipamentos de redes convencionais é comum que os planos de controle e de dados sejam implementados de forma acoplada dentro do mesmo *hardware*. Desta forma, qualquer funcionalidade nova só pode ser definida e implementada pelo fabricante do equipamento, tornando-se uma arquitetura rígida do ponto de vista de novas funcionalidades (SOFTWARE, 2012). A Figura 3 apresenta uma visão lógica dos planos de controle e de dados no mesmo equipamento.

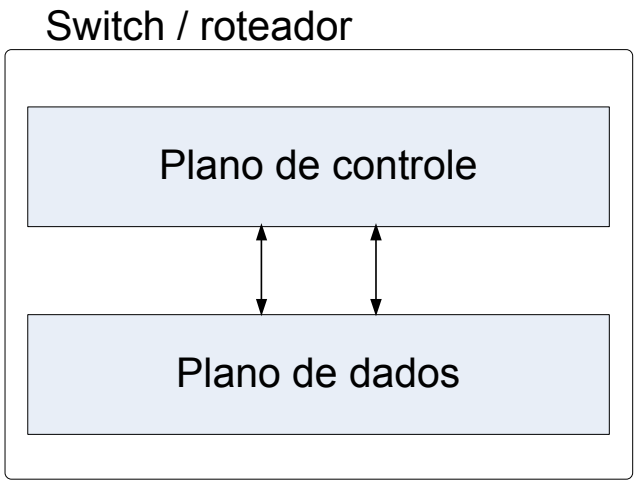


Figura 3 – Planos de dados e controle no mesmo *hardware*.

Na arquitetura das SDNs, o plano de controle é separado fisicamente do plano de dados e é acondicionado em um elemento externo chamado de Controlador. Desta forma, a definição do funcionamento da rede pode ficar a critério do administrador, enquanto que a implementação dos mecanismos de comutação dos dados continua sob o sigilo e

especialidade dos fabricantes. A Figura 4 apresenta uma visão da separação dos planos de controle e de dados em componentes distintos segundo o que preconiza a arquitetura SDN.

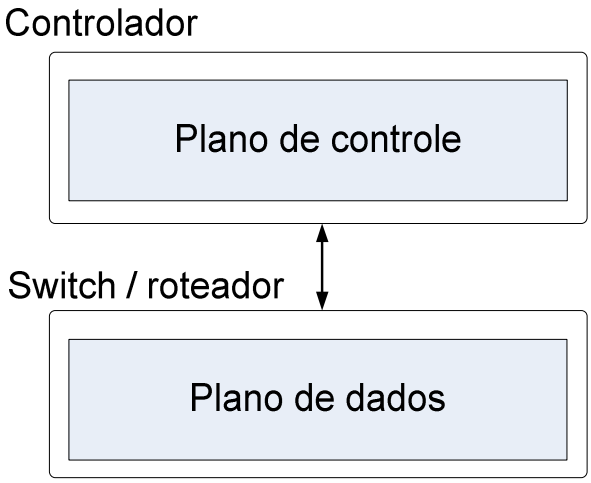


Figura 4 – Planos de dados e controle desacoplados.

Com as SDNs, o administrador passa a ter o controle da rede através de um ponto lógico central independentemente do fabricante do equipamento, o que simplifica bastante sua operação. Através da centralização do plano de controle, as SDNs permitem que a rede seja dinamicamente configurada por aplicações desenvolvidas pelos próprios administradores ou usuários da rede, que passam a enxergar a rede como sendo constituída de um único comutador. Portanto, a inteligência e o estado da rede são logicamente centralizados e, como resultado, as redes têm potencial para se tornarem mais flexíveis e facilmente adaptáveis (OPENFLOW, 2011; SOFTWARE, 2012).

3.2 OpenFlow

Atendendo as premissas das SDNs, o OpenFlow é considerado o primeiro padrão de *interface* de comunicação que especifica um protocolo com o propósito de interconectar os planos de controle e de dados definidos pela arquitetura de SDN, e vem recebendo o reconhecimento e apoio de grandes fabricantes de equipamentos e empresas de Tecnologia da Informação como NEC[1], HP[2], Extreme Networks[3], Google[4] e outros. Ele trabalha com o conceito de fluxos para definir tráfegos de rede baseado em regras predefinidas, que podem

[1] http://www.necam.com/SDN/
[2] http://h17007.www1.hp.com
[3] http://www.extremenetworks.com/solutions/datacenter_sdn.aspx
[4] http://research.google.com/pubs/Networking.html

ser configuradas de forma estática ou dinâmica por um Controlador SDN (SOFTWARE, 2012; MCKEOWN, 2008).

O OpenFlow foi concebido a partir da percepção que a maioria dos *switches Ethernet* e roteadores possuem tabelas de fluxos que, embora sejam diferentes para cada fabricante, possuem um conjunto de funções comuns que podem ser executadas em diversos dispositivos de rede (MCKEOWN, 2008). Estas tabelas de fluxos são comumente construídas em memórias do tipo *Ternary Content Addressable Memory* (TCAM), e operam em taxa de linha (*line-rate*) para a implementação de *firewalls*, lista de controle de acesso (*Access Control List* – ACL), tradução de endereço de redes (*Network Address Translation* – NAT) etc.

A especificação do OpenFlow define basicamente dois macro componentes representados na Figura 5, o Comutador (*switch*) e o Controlador.

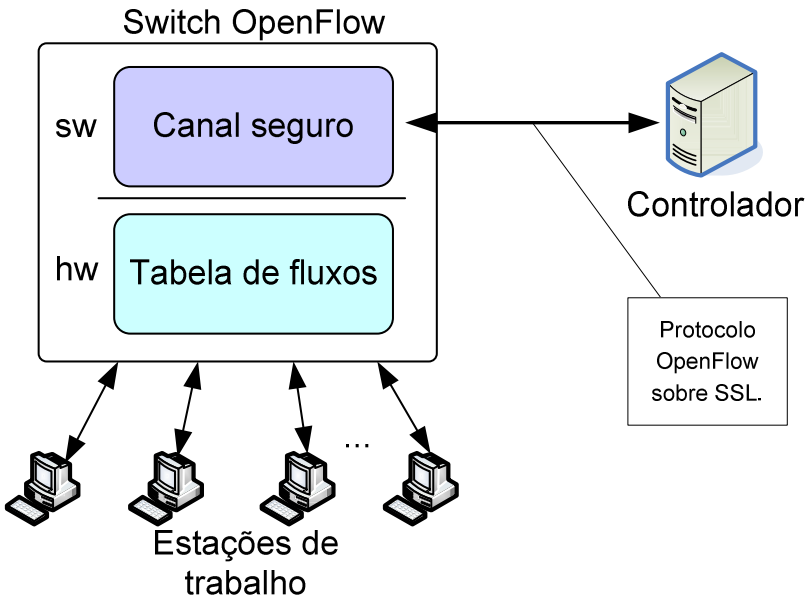


Figura 5 – Principais componentes do OpenFlow. Adaptado de (MCKEOWN, 2008).

3.2.1 Comutador OpenFlow

O Comutador OpenFlow é o equipamento de rede como *switch* ou roteador que, de forma idêntica a uma rede convencional, continua sendo o responsável pelo encaminhamento dos pacotes no plano de dados, de acordo com o que for definido na Tabela de Fluxos. Conforme a Figura 5, um Comutador OpenFlow é basicamente constituído de três elementos:

- **Tabela de Fluxos:** Utilizada para identificar fluxos e as suas respectivas ações de como o *switch* deve processar os pacotes do mesmo fluxo. Conforme a Figura 6, a Tabela de Fluxos é composta de três partes: campos de cabeçalhos, contadores e ações. A Figura 7 apresenta os campos de cabeçalho da

especificação 1.0 do OpenFlow (OPENFLOW, 2009). Os campos de cabeçalho são utilizados para comparar com as informações do cabeçalho retiradas do pacote recebido no *switch* (*Match*). Os contadores servem para contabilizar a quantidade de pacotes que coincidiram com aquele registro na Tabela de Fluxos. E as ações definem o tratamento que deve ser dado aos pacotes recebidos.

- **Canal Seguro:** Meio pelo qual o *switch* se comunica com o Controlador e vice-versa. É através deste canal que o Controlador configura e gerencia os *switches*, recebe eventos e notificações ou envia um pacote para um *switch*. A comunicação entre o Controlador e o *switch* deve ocorrer de forma segura utilizando-se o protocolo *Secure Sockets Layer* (SSL) entre eles;
- **Protocolo OpenFlow:** Define uma forma aberta e padronizada para a comunicação entre o Controlador e o Comutador. O protocolo OpenFlow especifica uma *interface* padrão pela qual as Tabelas de Fluxos podem ser definidas externamente, evitando a programação direta dos *switches*.

Em relação ao OpenFlow, um *switch* pode ser classificado em dois tipos: *Switch* OpenFlow dedicado e *Switch* habilitado com OpenFlow. O primeiro tipo de equipamento é específico para funcionar com o OpenFlow e somente encaminha pacotes entre as portas de acordo com as definições do Controlador. Já o *Switch* habilitado com OpenFlow é um *switch* tradicional dos fabricantes no qual os recursos para o OpenFlow foram adicionados. Desta forma pode encaminhar pacotes através do modo convencional, sem o uso de um Controlador.

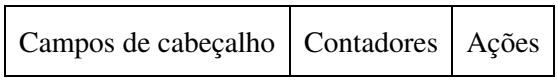


Figura 6 – Composição da Tabela de Fluxos em um Comutador OpenFlow. Adaptado de (MCKEOWN, 2008).

Os Comutadores OpenFlow devem suportar as seguintes ações básicas:

- **Encaminhar os pacotes por uma ou várias portas**: Utilizada para comutar os pacotes ao longo da rede. O encaminhamento dos pacotes depende da existência de entradas na Tabela de Fluxos com regras definindo o encaminhamento dos pacotes do fluxo;
- **Encapsular e encaminhar os pacotes do fluxo para o Controlador**: Na ausência de entradas na Tabela de Fluxos que coincidam com as características do fluxo, as informações do fluxo são encaminhadas para o Controlador para a

tomada de decisão. Normalmente ocorre com o primeiro pacote de um fluxo, para que seja decidido se um registro na Tabela de Fluxos deverá ou não ser adicionado;

- **Descartar os pacotes do fluxo**: Em vez de encaminhar os pacotes através das portas, um *switch* pode ser instruído para descartar os pacotes, seja por questões de segurança, controle da rede ou outro motivo qualquer.

Para os Comutadores habilitados com OpenFlow, uma quarta ação é prevista:

- **Encaminhar os pacotes pelo canal de processamento normal do *switch***: Esta ação pode ser utilizada para pacotes que não devem ser processados pelo domínio OpenFlow. Neste caso, o encaminhamento dos pacotes é feito nos mesmos moldes das redes convencionais.

Vale destacar que um fluxo é um conjunto de pacotes que possuem características em comum, como por exemplo: pacotes com o mesmo endereço de origem; pacotes destinados para um endereço IP específico e para uma mesma porta do *Transmission Control Protocol* (TCP); pacotes com os mesmos endereços *Media Access Control* (MAC) de origem para outro endereço MAC de destino; etc. A Figura 7 apresenta os principais campos de cabeçalho em uma Tabela de Fluxos que são definidos na especificação 1.0 do OpenFlow (OPENFLOW, 2009). Além destes campos, para cada entrada na Tabela de Fluxos são definidos alguns contadores e uma ação a ser tomada.

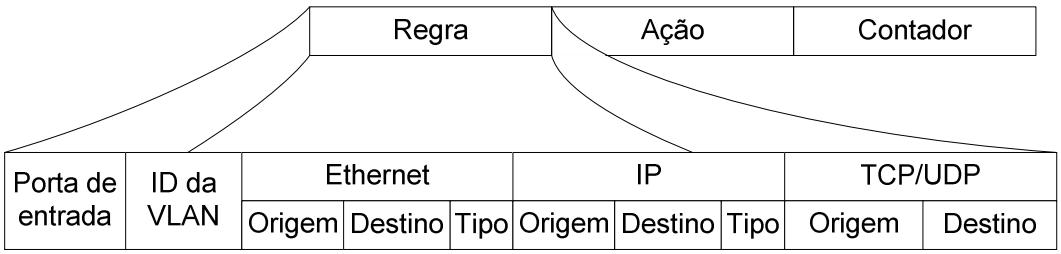


Figura 7 – Detalhe dos campos de cabeçalho de uma Tabela de Fluxos. Adaptado de Mckeown (2008).

3.2.2 Mensagens OpenFlow

A especificação tomada como base para este trabalho foi a versão 1.0 do protocolo OpenFlow, que define os seguintes três tipos de mensagens:

- **Controlador para Comutador**: este tipo de mensagem é enviado do Controlador para o Comutador e pode ou não requerer resposta dos Comutadores. O Quadro 2 lista as mensagens deste tipo.

Tipo da mensagem	**Descrição**
Features	Enviada para solicitar as características do Comutador. O Comutador deve enviar uma resposta com as características suportadas pelo *switch.*
Configuration	Enviada para configurar ou consultar parâmetros do Comutador. O Comutador envia uma resposta apenas para consultas.
Modify-State	Enviada para gerenciar o estado do Comutador.
Read-State	Enviada para coletar estatísticas da Tabela de Fluxos e portas do Comutador.
Send-Packet	Enviada para instruir o Comutador a enviar um determinado pacote por uma porta de saída específica.
Barrier	Utilizada para solicitar e receber confirmações da execução de operações.

Quadro 2– Mensagens do tipo Controlador para Comutador.

- **Assíncrona**: estas mensagens são enviadas pelo Comutador, sem a solicitação prévia do Controlador, para informar a chegada de pacote, mudança de estado do Comutador ou erros. O Quadro 3 lista as quatro mensagens deste tipo.

Tipo da mensagem	Descrição
Packet-in	Utilizada para informar o Controlador quando um pacote é recebido pelo Comutador e este não possui nenhuma entrada na Tabela de Fluxos que corresponda com as informações do pacote.
Flow-Removed	Utilizada para informar que uma entrada na Tabela de Fluxos foi removida.
Port-Status	Utilizada para informar a mudança de estado de uma porta. Ex.: quando a porta é ativada ou desativada.
Error	Utilizada para notificar problemas.

Quadro 3 – Mensagens do tipo assíncrona.

- **Simétrica**: São mensagens enviadas em ambas as direções, sem solicitação prévia. As três mensagens deste tipo são listadas no Quadro 4.

Tipo da mensagem	Descrição
Hello	Mensagens trocadas entre Controlador e Comutadores no estabelecimento de conexões.
Echo	Mensagens trocadas entre Controlador e Comutadores para manter conectividade.

	Uma mensagem *echo request* deve ser respondida com uma *echo reply*. Estas mensagens podem ser utilizadas para calcular latência e taxa de transferência.
Vendor	Alternativa utilizada para o Comutador informar funcionalidades adicionais dentro do escopo da mensagem OpenFlow.

Quadro 4 – Mensagens do tipo simétrica.

3.2.3 Controlador OpenFlow

O Controlador OpenFlow é onde o plano de controle é implementado. É o componente que abstrai as particularidades do equipamento de rede para facilitar a programação remota através do protocolo OpenFlow e do canal de comunicação seguro. Ele é o responsável por adicionar ou remover as entradas na Tabela de Fluxos, ou seja, é o elemento que define a comutação do tráfego no plano de dados.

Atualmente há uma série de Controladores sendo desenvolvidos nas mais variadas linguagens de programação. Dentre eles pode-se citar o Floodlight (FLOODLIGHT, 2012), Beacon (BEACON, 2013) e Maestro (MAESTRO, 2013), todos em Java; Nox (NOX, 2013), em C++; e Pox (POX, 2013), em Python. Para permitir a programabilidade da rede, o Controlador OpenFlow fornece uma *Application Programming Interface* (API) para que sejam desenvolvidas aplicações de diversas natureza como de encaminhamento, *firewall*, gerenciamento e monitoramento, que interagem com os mecanismos de encaminhamento dos Comutadores OpenFlow.

3.2.4 Funcionamento do OpenFlow

O funcionamento básico do OpenFlow consiste em, ao receber o primeiro pacote do fluxo no primeiro *switch* da rede, verificar se há algum tratamento definido para este fluxo na Tabela de Fluxos. Havendo uma entrada na Tabela de Fluxos, a ação definida é executada. Caso contrário, os dados do fluxo são extraídos, encapsulados e encaminhados ao Controlador para tomada de decisão (gerando uma mensagem do tipo *packet_in*).

Ao receber um *packet_in*, o Controlador consulta a(s) aplicação(ões) utilizada(s) na rede e decide como os pacotes deste fluxo devem ser tratados. Em seguida, uma resposta é enviada ao *switch*, que atualiza a Tabela de Fluxos com as instruções recebidas, através de mensagens do tipo *modify-state* seguido de outra do tipo *send-packet*. Assim, pacotes subsequentes do mesmo fluxo receberão o mesmo tratamento, até que a entrada na Tabela de Fluxos seja removida (OPENFLOW, 2011). A Figura 8 representa de forma resumida esses passos.

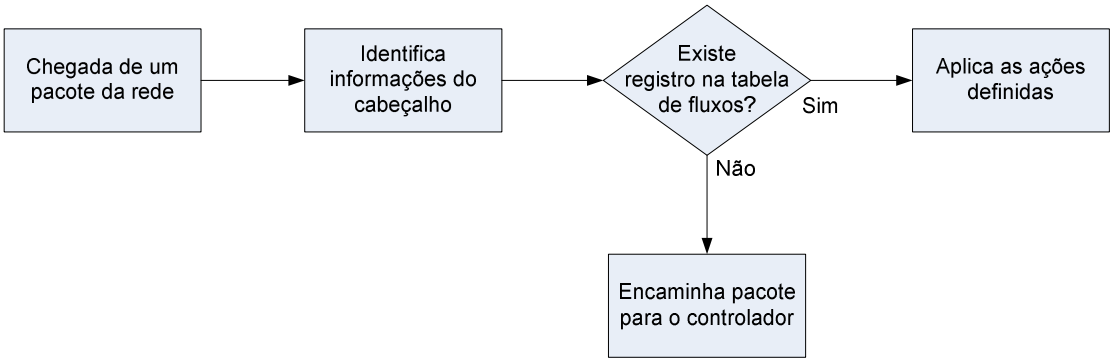


Figura 8 – Etapas do processamento dos fluxos em Comutador OpenFlow. Adaptado de (MCKEOWN, 2008).

4 IPSFlow

Diante das limitações apresentadas como a falta de integração entre os equipamentos de redes com os dispositivos de segurança e a deficiência na seleção do tráfego a ser encaminhado para análise, a flexibilidade oferecida pelas SDNs e a padronização proposta pelo OpenFlow tornam-se grandes aliadas na administração de IPSs, pois combinam a rápida comutação de pacotes no plano de dados, com a flexibilidade e a visão única do plano de controle (LANTZ, 2010). Desta forma, o uso da arquitetura SDN favorece o aumento da área de abrangência do IPS, permitindo este atuar de uma maneira mais proativa na origem do problema. Assim, esta dissertação apresenta o IPSFlow como alternativa para integrar os IDSs/IPSs com os dispositivos de rede. Até a conclusão deste trabalho, não se tem notícias de trabalhos com propósitos semelhantes ao IPSFlow.

Este Capítulo tem como objetivo apresentar o IPSFlow através da descrição de sua arquitetura e funcionamento geral, seguido do detalhamento da construção do Controlador IPSFlow.

4.1 Arquitetura e Funcionamento do IPSFlow

O IPSFlow é uma arquitetura de IPS que utiliza uma rede baseada no modelo de SDN e o protocolo OpenFlow para permitir a construção de um IPS distribuído com ampla cobertura e com a capacidade de efetuar a captura seletiva[5] de um fluxo para análise em um ou mais IDSs. Por atuar sobre uma SDN, o IPSFlow possui a capacidade de efetuar o bloqueio de um fluxo caracterizado como malicioso ou indevido, em um *switch* mais próximo de sua origem, impedindo assim, que o tráfego seja disseminado para dentro da rede. O IPSFlow também prevê mecanismos para que os resultados de diferentes IDSs sejam combinados para o bloqueio automático de um tráfego malicioso.

Na arquitetura do IPSFlow, o Controlador gerencia e armazena as regras que definem o encaminhamento dos fluxos na rede com base nas definições de segurança, e um ou mais IDSs que realizarão a análise do tráfego e geração de alertas. A Figura 9 apresenta uma visão geral da arquitetura do IPSFlow onde tem-se uma rede com *switches* OpenFlow conectados ao Controlador por onde o administrador fará toda a configuração das regras do ambiente. Os *switches* possuem conexões lógicas com o conjunto de IDSs, para onde o tráfego a ser

[5] Neste trabalho, define-se como captura seletiva a capacidade de selecionar quais tipos de fluxos devem ser capturados.

analisado deve ser enviado, que podem estar concentrados em um determinado local ou distribuídos pela rede.

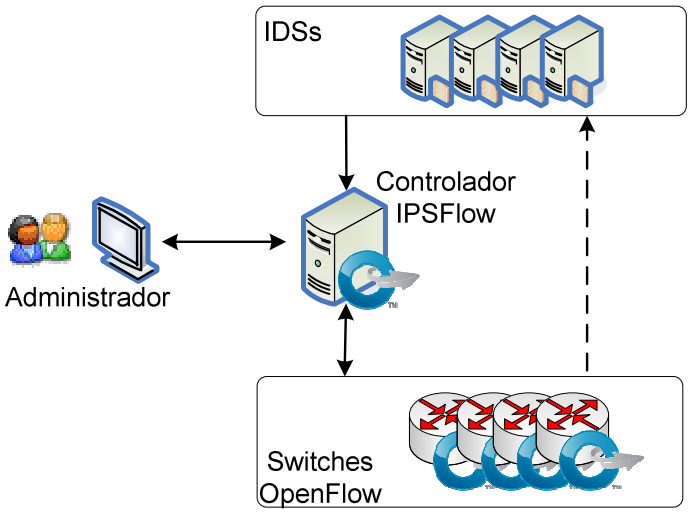


Figura 9 – Arquitetura do IPSFlow.

A Figura 10 apresenta o funcionamento do IPSFlow em uma SDN que utiliza *switches* OpenFlow. Quando o primeiro pacote de um fluxo é recebido no *switch* diretamente conectado ao *host* (passo 1), os processos de consulta à Tabela de Fluxos e o Controlador seguem conforme especificado no OpenFlow e apresentados no Capítulo 3 (passo 2) (OPENFLOW, 2011).

Ao receber uma requisição de um *switch*, o Controlador verifica a existência de regras para captura e análise do tráfego recebido. Caso o fluxo esteja definido para ser analisado, os pacotes do fluxo são encaminhados para o destinatário (passo 3) e uma cópia enviada para análise no(s) IDS(s) (passo 4). Ao concluir-se que o fluxo se trata de um tráfego malicioso, o(s) IDS(s) envia(m) o resultado ao Controlador (passo 5) para que o tráfego passe a ser bloqueado no *switch* OpenFlow mais próximo da origem. Caso contrário, nenhuma intervenção é feita no fluxo passante.

4.1.1 O Controlador IPSFlow

O Controlador IPSFlow foi construído baseado no Floodlight para estar alinhado à adoção deste controlador pelo Grupo de Estudos em Redes de Computadores e Comunicação Multimídia (GERCOM[6]) da Universidade Federal do Pará (UFPA) em seus projetos e para uma melhor integração com outras soluções desenvolvidas pelos demais membros do grupo.

[6] http://gercom.ufpa.br

Nesta versão inicial do Controlador IPSFlow, foram implementadas três *interfaces* de comunicação: o primeiro para fazer a comunicação com ferramenta cliente utilizada pelo administrador da rede, o segundo para fazer a comunicação com os *switches* OpenFlow e o terceiro para receber os alertas gerados pelo IDS.

A *interface* de comunicação voltada para o administrador é a responsável por tratar toda a interação com o administrador. Nesta operação, ele recebe as mensagens no formato *JavaScript Object Notation* (JSON) através de *interfaces web* do tipo Transferência de Estado Representativo (*Representational State Transfer* – REST). Através desta *interface*, o administrador tem a capacidade de determinar quais fluxos podem trafegar na rede e definir quais destes fluxos devem ser encaminhados para análise através da configuração de um campo "booleano" denominado *analyse*. Caso este campo esteja configurado como verdadeiro, os fluxos que coincidirem com este registro na Tabela de Fluxos serão encaminhados para serem analisados pelo IDS.

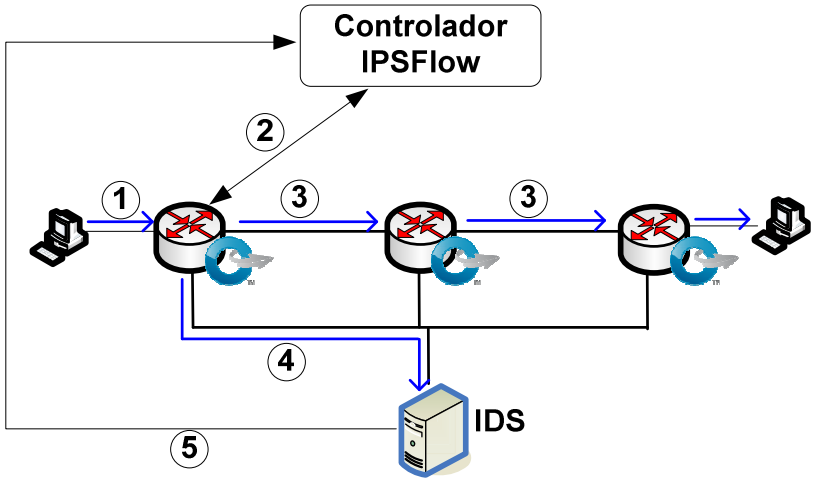


Figura 10 – Funcionamento do IPSFlow.

Já a *interface* de comunicação voltada para o *switch* é por onde o IPSFlow recebe as mensagens do tipo *packet_in* dos *switches* OpenFlow e envia as do tipo *flow_mod* e *packet_out* para a configuração das Tabelas de Fluxo. E por fim, a *interface* de comunicação do IDS é por onde o Controlador recebe as notificações de alertas dos IDSs através de mensagens recebidas através de conexões *socket* TCP.

A Figura 11 apresenta as *interfaces* de comunicação explanadas anteriormente, bem como os módulos que compõem o Controlador IPSFlow, que serão detalhados à seguir.

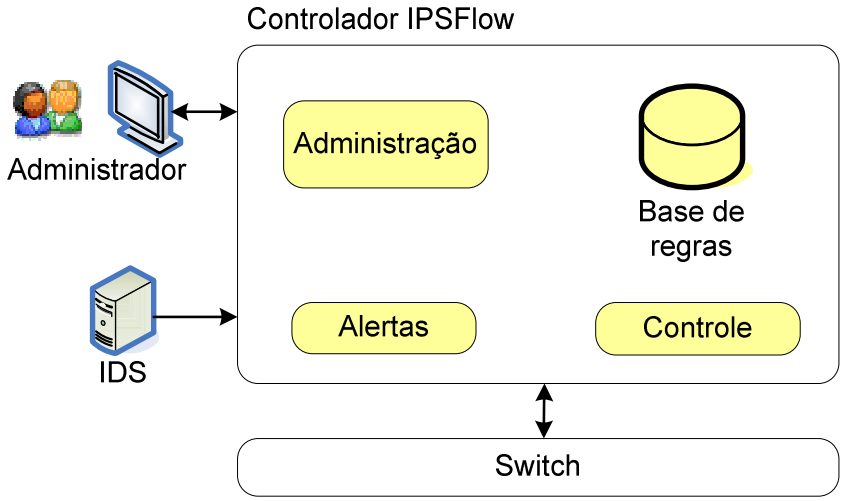


Figura 11 – Módulos do Controlador IPSFlow.

4.1.2 Módulo Administração

Este módulo é responsável por receber as mensagens da ferramenta cliente que o administrador de rede utilizará para definir instruções de quais tipos de fluxos devem ser liberados para trafegar na rede e se este tipo de tráfego deve ser analisado pelo(s) IDS(s). Também é através deste módulo que o administrador pode consultar as regras definidas na Base de Regras e ativar ou desativar o funcionamento do IPSFlow na rede.

A implementação atual deste módulo prevê a troca de mensagens no formato JSON através da *interface web* do tipo REST, mas pode ser adaptado para outros formatos como o *eXtensible Markup Language* (XML) através do uso de APIs. O Quadro 5 apresenta os principais métodos REST que são disponibilizados pelo IPSFlow.

***Uniform Resource Ientifier* (URI)**	**Método**	**Argumentos do URI (op)**	**Campos de dados**	**Descrição**
/wm/ipsflow/module/<op>/json	GET	*status*	--	Verifica o estado do IPSFlow.
		enable		Habilita o IPSFlow no Controlador.
		disable		Desabilita o IPSFlow no Controlador.
/wm/ipsflow/rules/json	GET	--	--	Lista todas as regras existentes no formato JSON.
	POST	--	Par de parâmetros no formato "**campo**":"**valor**"; pode ser qualquer	Cria uma nova regra IPSFlow

			combinação das opções: “**switchid**”:“<xx:xx:xx:xx:xx:xx:xx:xx>” “**src-inport**”:“<short>” “**src-mac**”: “<xx:xx:xx:xx:xx:xx>” “**dst-mac**”: “<xx:xx:xx:xx:xx:xx>” “**dl-type**”: “<ARP ou IPv4>” “**src-ip**”: “<A.B.C.D/M>” “**dst-ip**”: “<A.B.C.D/M>” “**nw-proto**”: “<TCP ou UDP ou ICMP>” “**tp-src**” :“<short>” “**tp-dst**”:“<short>” “**priority**”:“<int>” “**action**”:“<ALLOW ou DENY>” “**analyse**”:“<TRUE ou FALSE>”	com base nas informações passadas no corpo da mensagem JSON.
	DELETE	--	"**ruleid**": "<int>"	Apaga a regra “ruleid”.

Quadro 5 – Métodos REST expostos pelo IPSFlow.

Embora o desenvolvimento da ferramenta cliente de administração do IPSFlow não tenha sido foco desta dissertação, utilitários como o comando *curl*, existente nos ambientes Unix/Linux, podem ser utilizados para enviar comandos e receber respostas do IPSFlow. A Figura 12 apresenta alguns exemplos de comunicação com o IPSFlow utilizando o comando *curl*.

```
curl http://localhost:8080/wm/ipsflow/module/enable/json

curl -X POST -d '{"src-ip": "10.0.0.3/32", "nw-proto":"ICMP",
"analyse":"TRUE" }' http://localhost:8080/wm/ipsflow/rules/json

curl -X POST -d '{"src-ip": "10.0.0.2/32", "dl-type":"ARP" }'
http://localhost:8080/wm/ipsflow/rules/json

curl -X POST -d '{"src-ip": "10.0.0.1/32", "nw-proto":"UDP", "tp-
src":"5060", "action":"DENY" }'
http://localhost:8080/wm/ipsflow/rules/json

curl -X POST -d '{"src-ip": "10.0.0.7/32", "dst-ip": "10.0.0.10/32", "nw-
proto":"TCP", "tp-dst":"80", "action":"ALLOW", "analyse":"TRUE"  }'
http://localhost:8080/wm/ipsflow/rules/json
```

Figura 12 – Exemplos de execução do comando *curl* para o IPSFlow.

4.1.3 Módulo Controle

O módulo de controle é o que recebe de fato as mensagens do tipo *packet_in* dos *switches*. Nele, as informações do fluxo são extraídas e utilizadas para consultar a base de

regras para verificar quais ações devem ser tomadas para o fluxo recebido. A decisão pode ser para descartar os pacotes, permitir o encaminhamento sem análise ou permitir o encaminhamento com cópia para análise no IDS. De forma semelhante ao funcionamento da aplicação Firewall, no IPSFlow, todos os fluxos devem ser descritos para que os pacotes de rede sejam encaminhados ao longo da rede.

Após a mensagem do tipo *packet_in* ser processada, um caminho lógico entre os *hosts* de origem e destino do fluxo é definido. Uma mensagem do tipo *flow_mod* é criada e envidada para os *switches* participantes para a configuração do caminho fim-a-fim. São nestas mensagens *flow_mod* que o espelhamento de portas é definido para que o IDS possa capturar o tráfego.

No pacote padrão do Floodlight, as mensagens do tipo *flow_mod* e *packet_out* eram armazenados temporariamente em um *buffer* antes de serem enviados para o *switch*. Este retardo provocava o funcionamento inadequado do IPSFlow, pois era necessário que estas mensagens fossem enviadas logo após suas criações. Desta forma, foi criado o módulo de envio de mensagens OpenFlow, com o objetivo de evitar este armazenamento prévio das mensagens *flow_mod* e *packet_out.* Neste módulo, as mensagens são enviadas tão logo estas sejam criadas.

4.1.4 Módulo Base de Regras

Este módulo é o encarregado pela manutenção da base de regras, que é o local onde as ações a serem tomadas para um determinado fluxo são armazenadas, tais como o bloqueio de fluxo e o encaminhamento de fluxo por uma porta com ou sem cópia para análise. É nele que o módulo Controle irá se basear para responder a uma requisição de um *switch*.

4.1.5 Módulo Alertas

Na situação em que uma cópia do tráfego está sendo encaminhado para ser analisado em um IDS, durante a passagem dos pacotes, caso o IDS detecte algum tráfego indevido e gere um alerta, este é o módulo responsável por receber, tratar o resultado da análise do IDS. Para bloquear um fluxo, este módulo constrói mensagens do tipo *flow_mod* semelhante ao utilizado no módulo Controle e envia para o *switch* mais próximo do *host* originador do tráfego que gerou o alerta. A diferença destas mensagens está no campo *action*, que agora instrui o *switch* a descartar os pacotes daquele fluxo.

5 Coleta de Resultados

Uma tentativa de ataque pela rede normalmente é precedido pela etapa de reconhecimento do alvo. Nesta etapa, tenta-se fazer um levantamento de quais computadores estão conectados, quais serviços de rede são disponibilizados por eles, nome e versão dos respectivos *daemons* destes serviços, versões do sistema operacional etc. Este levantamento é amplamente utilizado pelos atacantes para identificar potenciais alvos, uma vez que permite associar vulnerabilidades com as informações coletadas.

Por ser uma das primeiras etapas de um possível ataque, este tipo de atividade é o tipo de incidente responsável por quase 50% dos incidentes reportados ao CERT.br (CERT, 2013). Uma vez que uma varredura mal sucedida tende a confundir o atacante, os experimentos com o IPSFlow focaram no bloqueio deste tipo de tráfego. Para isto, foram elaborados seis cenários de rede no Mininet (MININET, 2013) e com o auxílio da ferramenta Nmap (NMAP,2013), simulou-se varreduras na rede entre dois *hosts*. Para a detecção, utilizou-se o Snort (SNORT, 2013) com uma regra específica para gerar alertas na detecção de pacotes característicos de uma varredura.

Optou-se por utilizar o Mininet, Snort e Nmap para os experimentos, pois estas ferramentas são de código aberto, livres de uso e, consequentemente, bastante utilizados. No caso específico do Mininet, a indisponibilidade de *switches* reais com suporte a OpenFlow também contribuiu para a escolha desta ferramenta.

Neste Capítulo serão apresentados com mais detalhes as ferramentas utilizadas nos experimentos, bem como o ambiente e a metodologia de experimentos elaborados para avaliar o funcionamento do IPSFlow.

5.1 Ferramentas auxiliares

Para a realização dos experimentos com o IPSFlow, foram utilizadas diversas ferramentas, das quais as mais importantes serão apresentadas com mais detalhes nas seções a seguir.

5.1.1 Mininet

O Mininet (MININET, 2013) é emulador de rede que permite a criação de redes virtuais como *hosts*, *switches*, roteadores e enlaces sobre um único núcleo (*kernel*), e tem como base o protocolo OpenFlow.

Uma rede virtual no Mininet pode ser construída através do comando *mn* com alguns parâmetros adicionais. O Quadro 6 lista algumas das principais opções utilizadas para construção de redes no Mininet. Outras opções podem ser consultadas em sua documentação (MININET, 2013).

Opção	Descrição
--test	Executa o comando passado como parâmetro após a criação da rede. Exemplo: "pingall", "ping", "iperf" etc.
--topo	Define a topologia da rede a ser criada.
--switch	Define o tipo de *switch* a ser utilizado na rede. Exemplo: "user", para executar o *switch* no *namespace* do usuário; "ovsk", para utilizar o *switch* do tipo Open vSwitch.
--controller	Especifica qual Controlador será utilizado na rede. Exemplo: "nox", para utilizar o Controlador NOX *Classic* que pode ser instalado junto com o Mininet; "remote", para indicar o uso de um Controlador externo. Caso utilize a opção remote, informações adicionais como endereço IP e porta de conexão ao Controlador devem ser especificadas.
--custom	Algumas topologias básicas já estão disponibilizadas no pacote de instalação do Mininet. Porém, topologias personalizadas podem ser criadas utilizando-se sua API Pyhton. Esta opção especifica qual arquivo contém as informações de topologias a serem utilizadas no Mininet.

Quadro 6 – Algumas das principais opções utilizadas na criação de uma rede virtual no Mininet.

Ao construir uma rede, o Mininet emula os enlaces, *hosts*, *switches* e Controladores conforme apresenta a Figura 13. Para os enlaces, é criado um par de "*Ethernet* virtual" (*veth-pair*), atuando como uma conexão ponto-a-ponto entre os dispositivos. Os *hosts* são processos de *Shell*, cada um movido e protegido em um *namespace* e conectados com o processo pai do Mininet através de um tubo (*pipe*).

Os *switches* virtuais OpenFlow proveem o mesmo mecanismo de configuração e funcionamento de um *switch* OpenFlow real. O Controlador pode estar localizado tanto na rede simulada quanto em uma rede real, bastando que haja conectividade IP entre o Controlador e o equipamento onde o *switch* está sendo executado.

O Mininet está disponível para ser utilizado de duas formas. Na primeira, máquinas virtuais com todo o ambiente do Mininet configurado pode ser baixado da Internet e

carregados em *players* de virtualização como VirtualBox[7] ou VMware[8]. Na segunda, o pacote de instalação pode ser baixado e instalado diretamente no computador onde o Mininet será executado.

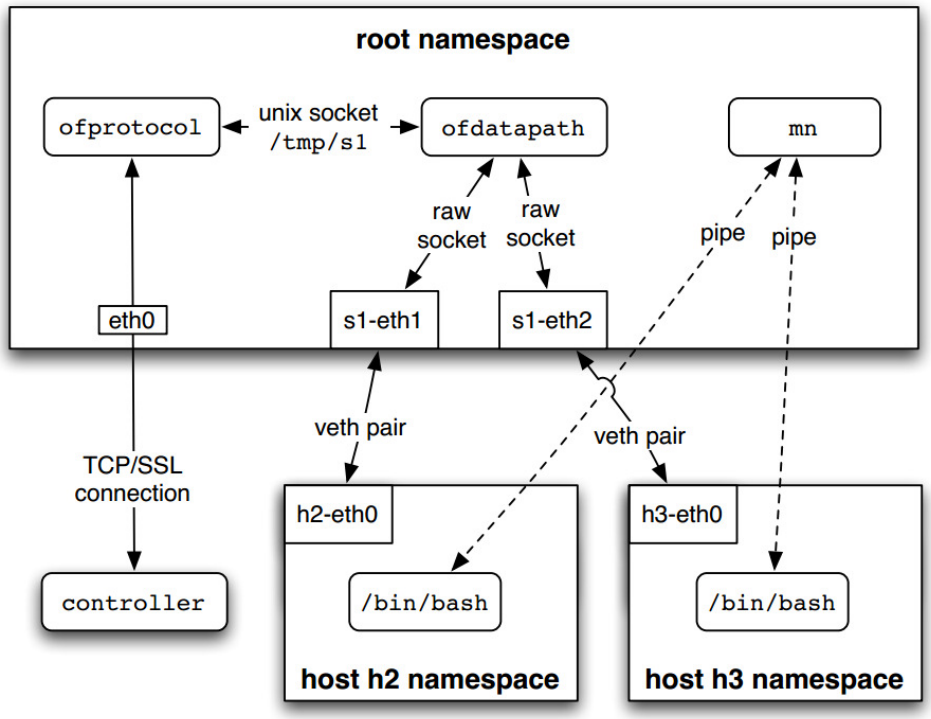


Figura 13 – Exemplo com visão geral do Mininet. Fonte: (LANTZ, 2010)

Dentre as principais características do Mininet, pode-se destacar:

- **Imediato**: uma pequena rede pode ser criada em alguns segundos e com isso as etapas de execução, edição e *debug* podem ser realizadas rapidamente.
- **Flexível para criação de topologias**: permite a criação desde redes simples com um *switch* a redes mais complexas como a de um centro de dados ou de um *backbone*.
- **Permite a execução de programas reais**: Qualquer programa disponível no *host* onde o Mininet esteja sendo executado estará disponível para o *host* virtual do Mininet.
- **Facilmente portável**: Os *switches* no Mininet podem ser programados utilizando-se o protocolo OpenFlow. Desta forma, a rede virtual pode ser facilmente portada para uma rede real com *switches* OpenFlow.

[7] http://www.virtualbox.org
[8] http://www.vmware.com

- **Replicável**: As redes virtuais podem ser facilmente replicadas e os experimentos executados em outros computadores.
- **Fácil uso**: Experimentos no Mininet podem ser executados a partir de *scripts* na linguagem de programação Python.
- **Aberto com desenvolvimento e evolução constante**: Como o código fonte está disponível, possui uma ampla comunidade de desenvolvimento no qual qualquer um pode estudá-lo, modificá-lo, corrigir erros, solicitar ou implementar novos recursos e disponibilizar correções.

Porém, por ser um ambiente emulado, o Mininet possui também algumas limitações:

- **Compartilhamento de recursos**: os recursos do *host* como processador, memória, sistema de arquivos e escopo de processos (*Process Identifier* - PID) são compartilhados entre os elementos virtuais (*hosts*, *switches*, roteadores) e o *host* real. Desta forma, modificações em arquivos ou encerramento de processos acidentais impactam diretamente no *host*.
- **Único tipo de *kernel*:** Por utilizar um único *kernel*, não é possível executar programas baseados em *Berkeley Software Distribution* (BSD), Windows ou outro sistema operacional nos *hosts* virtuais.
- **Rede isolada**: Não há comunicação entre a rede do Mininet com a rede onde o *host* está conectado, ou seja, não há comunicação do ambiente Mininet com uma rede real.

5.1.2 Snort

O Snort é um sistema de detecção e prevenção de invasão (IDS/IPS) livre, bastante conhecido pela comunidade de segurança, e que combina a análise baseada em regras e anomalias (SNORT, 2013), podendo ser configurado para operar em três modos: *sniffer*, *packet logger* e de sistema de detecção de intrusão (NIDS).

No modo *sniffer*, o Snort simplesmente captura o tráfego de rede e os apresenta de forma contínua na tela. No modo *packet logger*, além de capturar o tráfego, o Snort grava as informações dos pacotes em disco. E no modo NIDS, atua como um IDS de rede, capturando e realizando análise sobre os pacotes de rede. O modo NIDS é o que mais tem relação com este trabalho e será mais detalhado.

O Snort é composto por quarto componentes: Decodificador de Pacote, Pré-processador, Motor de Detecção e Módulos de Saída (KOHLENBERG, 2007), que operam

em conjunto para detectar ataques e gerar saídas em um dos formatos disponíveis, que serão apresentados na seção 5.1.1.4. A Figura 14 apresenta como estes componentes são arranjados e o fluxo seguido para a captura e detecção de um possível ataque.

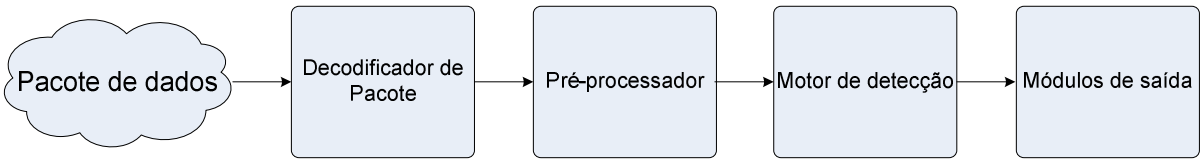


Figura 14 – Componentes básicos da arquitetura do Snort. Adaptado de (KOHLENBERG, 2007).

5.1.2.1 Decodificador de Pacote

O Decodificador de Pacotes é o responsável por capturar pacotes e prepará-los para serem encaminhados para o Pré-processador ou diretamente ao Motor de Detecção. A captura de pacotes pode ser feita de diversas maneiras através das Bibliotecas de Aquisição de Dados (*Data Acquisition Library* - DAQ).

A DAQ substitui as chamadas diretas para as funções da biblioteca *libpcap* e atua como uma camada de abstração entre os diversos *hardwares* e as *interfaces* de *software*, sem que modificações sejam feitas no Snort. A DAQ comumente utilizada é a Pcap e é a utilizada por padrão, caso nenhuma outra seja especificada. Além de escutar uma *interface* de rede para capturar os pacotes, o uso desta DAQ ainda permite especificar um arquivo com as informações dos pacotes previamente capturados.

5.1.2.2 Pré-processador

São como *plugins* que podem ser carregados dinamicamente e utilizados com o Snort para organizar ou modificar os pacotes, antes do Motor de Detecção executar qualquer atividade sobre estes. Os pré-processadores são bastante úteis na detecção de intrusão, pois podem executar tarefas como desfragmentação de pacotes, decodificar URIs *Hypertext Transfer Protocol* (HTTP), remontar fluxos TCP etc.

5.1.2.3 Motor de Detecção

Este é o componente mais importante do Snort, pois é o responsável por tentar detectar a existência de sinais de ataque nos pacotes de dados. O Motor de Detecção utiliza regras como base para a análise dos pacotes. Cada regra possui uma série de condições que ao serem satisfeitas, uma ação é tomada sobre o pacote. O tempo da análise realizada pelo Motor de

Detecção depende basicamente da quantidade de regras configuradas, da configuração do equipamento utilizado para o Snort e a quantidade de tráfego capturado na rede.

A detecção de atividades maliciosas no Snort é baseada em regras, que são especificações de assinaturas de atividades maliciosas previamente identificadas e catalogadas. As assinaturas podem estar presentes no cabeçalho ou mesmo na carga (*payload*) do pacote.

Uma regra no Snort pode ser utilizada para gerar alertas, registrar em *log* alguma mensagem, ou bloquear ou liberar um pacote (caso o Snort atue no modo ativo) de acordo com a sua construção. A regra no Snort é composta por duas partes: uma parte para os cabeçalhos da regra e outra para opções da regra (REHMAN, 2003), conforme apresenta a Figura 15.

Figura 15 – Estrutura de uma regra do Snort. Adaptado de (REHMAN, 2003).

a) Cabeçalho da Regra

A parte dos cabeçalhos da regra contém informações de qual ação deve ser tomada pela regra e define também os critérios que devem ser utilizados para comparar com o cabeçalho dos pacotes recebidos (*matching*). A parte das opções da regra contém critérios adicionais utilizados também para comparar com os pacotes de dados. A estrutura geral do cabeçalho de uma regra do Snort pode ser observada na Figura 16.

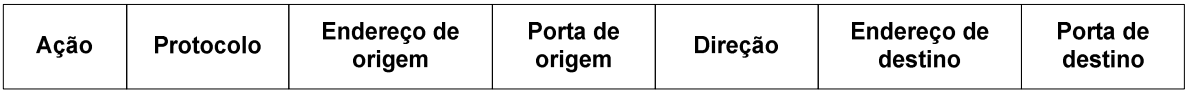


Figura 16 – Estrutura do cabeçalho de uma regra do Snort. Adaptado de (REHMAN, 2003).

a.1) Campo Ação

O campo de ação determina qual ação deve ser tomada ao encontrar um pacote de dados que coincida com os critérios de comparação. Uma ação só é tomada quando todas as condições definidas na regra coincidem com as informações do pacote. O Snort possui cindo ações predefinidas:

- ***Pass***: Instrui o Snort a ignorar a análise sobre o pacote.
- ***Log***: Utilizada para registrar as informações do pacote de acordo com as opções de *log* que foram definidas no arquivo de configuração do Snort.

- ***Alert***: Gera alertas nas situações onde todas as condições da regra coincidam com as informações do pacote. Assim como no *Log*, o tipo de alerta gerado é definido nas configurações do Snort. A diferença entre as ações *Log* e *Alert* é que a primeira somente registra as informações do pacote enquanto que a *Alert*, além de registrar as mesmas informações, gera alertas para o administrador.
- ***Activate***: Esta ação é utilizada para gerar um alerta e ativar uma outra regra logo em seguida. Desta forma, condições adicionais podem ser verificadas nas novas regras ativadas.
- ***Dynamic***: São as regras invocadas através das regras com a ação *Activate* definida previamente. Estas regras só são utilizadas para comparar com um pacote se forem ativadas por outra.

Além destas regras predefinidas, o Snort também permite que novas ações sejam criadas pelo administrador, como por exemplo, o envio de mensagens ao serviço *Syslog*, envio de mensagens *trap* de *Simple Network Management Protocol* (SNMP), armazenar os dados do pacote em arquivos XML ou combinar as diversas outras ações do Snort.

a.2) Campo Protocolo

O segundo campo de uma regra do Snort define para qual tipo de pacote a regra deve ser aplicada. O Snort consegue processar os protocolos: IP, *Internet Control Message Protocol* (ICMP), TCP e *User Datagram Protocol* (UDP).

a.3) Campo Endereço de Origem ou Endereço de Destino

Existem dois campos de endereços IPs, um para verificar a origem e outro para verificar o destino dos pacotes. Este campo pode ser utilizado para definir um único *host* ou um endereço de rede. O termo *any* pode ser utilizado para generalizar e aplicar a regra para qualquer endereço. O endereço IP é definido no formato *Classless InterDomain Routing* (CIDR) (FULLER, 2006), onde o endereço IP é sucedido de uma "/" e um número inteiro que representa a quantidade de *bits* da máscara de rede. Por exemplo, “192.168.0.1/24” representa todos os *hosts* da rede com endereço de 192.168.0.1 a 192.168.0.254; e “10.0.0.1/32” representa apenas o *host* com o endereço IP 10.0.0.1.

Como mencionado anteriormente, há um campo para definir o endereço de IP de origem e outro para definir o endereço de destino do pacote. O endereço de origem é especificado no terceiro campo e o endereço de origem é definido no sexto campo.

Ainda há a possibilidade de se definir faixas de endereços ou mesmo a exclusão de determinados endereços. A lista de endereços é definida separando-se os endereços por vírgula e envolvendo-os com colchetes. A exclusão é feita utilizando-se o sinal de exclamação antes do(s) endereço(s). Por exemplo, "[10.0.0.1/32,192.168.0.1/24]" representa os *hosts* com endereços 10.0.0.1 e de 192.168.0.1 a 192.168.0.254; e a "!10.0.0.2/32" faz com que pacotes com este endereço sejam ignorados pela regra.

a.4) Campo Porta de Origem ou Porta de Destino

Os campos de número da porta definem para quais portas de origem e destino da camada de transporte uma regra deve ser aplicada. De forma semelhante ao endereço IP, o termo *any* também pode ser utilizado para aplicar a regra para qualquer porta. O campo de número de porta é relevante apenas nos casos em que o protocolo especificado seja TCP ou UDP. Caso o protocolo especificado seja IP ou ICMP, este campo não possui influência na regra. O campo de número de porta também pode ser definido por faixas, bastando para isso separar os limites com um sinal de dois pontos (:). Pode-se especificar apenas um dos limites da faixa. Desta forma, especifica-se apenas o início ou o fim da faixa. Exemplo: ":1024" especifica todas as portas até a 1024, inclusive; e "1:100" especifica as portas de 1 a 100.

De forma semelhante ao endereço IP, uma porta pode ser ignorada utilizando o sinal de exclamação (!). Porém, não é permitido o uso de vírgulas para definir várias portas como no caso dos Endereços de Origem ou de Destino.

a.5) Campo Direção

O campo direção indica a orientação ou direção na qual a regra deve ser aplicada. A direção pode ser -> ou <>. No uso do operador ->, o endereço IP e o número da porta do lado esquerdo do operador são considerados como sendo da origem do tráfego; e as informações do lado direito são consideradas como sendo do destino. O operador bidirecional <> instrui o Snort a considerar o par endereço/porta em ambas as direções.

O campo de opções é a porção que segue a parte de cabeçalho de uma regra e é envolvida por parênteses. Em uma regra podem ser definidas apenas uma ou várias opções separadas por ponto-e-vírgula. No uso de várias opões, é realizada uma comparação lógica E (AND) entre as opções e a regra só se torna válida quando todos os critérios foram satisfeitos. As opções são definidas por palavras-chave e algumas permitem a definição de argumentos adicionais. Na existência de mais de um

parâmetro, estes são separados por vírgula. A Figura 17 apresenta alguns exemplos de opções definidas para uma regra. As opções que aceitam argumentos possuem os

```
(msg: “Alerta de invasão”)
(msg: “content:"TEST"; gid:1000001; rev:1)
(flags: SF; msg: “SYN-FIN Scan”; gid:1000003)
```

Figura 17 – Exemplo de duas opções definidas em uma regra no Snort.

argumentos definidos após o sinal de dois pontos (:).

b) Opções da Regra

O Snort possui uma grande variedade de opções que podem ser utilizadas para auxiliar em uma melhor definição de uma regra. Este trabalho listará algumas das principais opções disponíveis:

- ***Flags***: Utilizado para verificar quais *flags* estão configuradas no cabeçalho TCP de um pacote. Estas *flags* são utilizadas por diversas ferramentas de segurança, inclusive por ferramentas varredura de portas como o Nmap (NMAP, 2013). O Snort tem a capacidade de verificar as *flags* apresentadas no Quadro 7, as quais podem ser manipuladas com operadores de negação (!), AND (+) ou OR (*).
- ***Ack***: Determina o número de reconhecimento (*Acknowledgement Number*) que deve ser verificado no cabeçalho TCP de um pacote, quando a *flag* ACK é configurada. Ferramentas como o Nmap utilizam este recurso do cabeçalho TCP para verificar se um *host* pode ser alcançado e realizar a varredura de serviços abertos, fechados ou possivelmente filtrados.
- ***Classtype***: Utilizado para classificar uma determinada regra tomando como base o arquivo de classificação “classification.config” do Snort.
- ***Content***: O Snort tem a capacidade de verificar um determinado padrão dentro do conteúdo do pacote. Este padrão pode ser sequência de caracteres ASCII ou dados binários no formato hexadecimal.
- ***Offset:*** Esta opção é utilizada em conjunto com a opção *Content* e determina a quantidade de *bytes* que deve ser deslocado para iniciar a busca pelo padrão definido em *Content*.

Flag	Argumento utilizado na regra do Snort
FIN - *Flag* de *Finish*	F
SYN - *Flag* de *Sync*	S
RST - *Flag* de *Reset*	R
PSH - *Flag* de *Push*	P
ACK - *Flag* de *Acknowledge*	A
URG - *Flag* de *Urgent*	U
CWR - *Congestion Window Reduced*	C
ECE - ECN-Echo	E
Nenhuma *flag* definida	0

Quadro 7 – Lista de *flags* TCP que o Snort é capaz de verificar nos pacotes TCP. Adaptado de (SNORT, 2013).

- ***Depth***: Utilizado também em conjunto com a opção *Content*, determina o limite máximo na busca pelo padrão definido em *Content.* A combinação das opções *Offset* e *Depth* delimitam o escopo da pesquisa.
- ***Content-list***: É utilizado em conjunto com um nome de arquivo. No arquivo utilizado como argumento para esta opção deve conter as sequencias de caracteres que devem ser pesquisadas dentro do pacote.
- ***Nocase***: Utilizado em conjunto com a opção *Content* e sua função é determinar que a pesquisa seja feita por padrões de forma insensível à caixa. Esta opção não possui argumentos.
- ***Itype***: Esta opção é utilizada para identificar o tipo de um pacote ICMP. O Quadro 8 apresenta os tipos de pacote ICMP que podem ser detectados.
- ***Icode***: Esta opção é utilizada para identificar o código de um pacote ICMP. A opção *Itype* define o tipo e *Icode* define o subtipo do pacote ICMP.
- ***Logto***: Esta opção instrui o Snort a gravar os registros de *log* no arquivo especificado como argumento.
- ***Msg***: Conforme apresentado em exemplo anterior, esta opção adiciona a mensagem definida nos *logs* e nos alertas.
- ***Priority***: Opção utilizada para definir a prioridade de uma regra dentre as diversas da mesma classificação.
- ***Rev:*** Serve para identificar a versão de uma regra e distingui-las para propósitos de manutenção.

Valor	Tipo do pacote ICMP
0	*Echo reply*
3	*Destination unreachable*
4	*Source quench*
5	*Redirect*
8	*Echo request*
11	*Time exceed*
12	*Parameter problem*
13	*Timestamp request*
14	*Timestamp reply*
15	*Information request*
16	*Information reply*

Quadro 8 – Tipos de pacotes ICMP que podem ser detectados pelo Snort. Adaptado de (REHMAN, 2003).

- ***Flow***: A opção *Flow* é utilizada para aplicar regras em sessões TCP para pacotes seguindo por uma determinada direção.
- ***Sid**:* Opção utilizada para identificar uma regra no Snort. Os identificadores de 0 a 99 são reservados para uso futuro, de 100 a 1.000.000 são reservados para as regras predefinidas nas distribuições do Snort, e os acima de 1.000.000 podem ser utilizados para regras locais.
- ***Threshold***: Define limites para a aplicação da regra. Esta opção possui configurações adicionais como *type*, *track*, *count* e *seconds*. A configuração *type* pode ser *limit*, *threshold* ou *both*. No *limit*, um alerta é gerado apenas para o primeiro evento em um determinado intervalo de tempo e ignora os demais eventos que ocorrerem no intervalo de tempo. No *threshold*, um alerta é gerado a cada ocorrência do evento durante o intervalo de tempo. E em *both*, um alerta é gerado a cada ocorrência do evento em um determinado intervalo de tempo após ocorrerem um determinado número do mesmo evento. A configuração *track* pode ser *by_src* ou *by_dst* para incrementar o contador *count* por endereço IP de origem ou de destino. E *count* e *seconds* determinam a quantidade de ocorrências e o intervalo de tempo a considerar para os eventos.

5.1.2.4 Módulos de Saída

São *plugins* que controlam o tipo de saída gerado pelo Snort. De acordo com o resultado da análise realizada pelo Motor de Detecção sobre um pacote, as informações do pacote podem ser utilizadas para serem registradas em *log*, gerar alertas, enviar *traps* SNMP, enviar mensagens ao sistema *syslog*, gerar saídas em XML, gerar registros em bancos de dados etc.

O Snort possui uma variedade de opções para módulos de saída. Dentre as principais opções pode-se destacar:

- ***Alert_syslog***: Este módulo envia os alertas para o sistema de *logs* do sistema operacional e permite uma grande flexibilidade para o usuário, pois permite especificar as facilidades e as prioridades em que os alertas devem ser registrados.
- ***Alert_fast***: Este módulo gera os alertas de forma compacta em apenas uma linha no arquivo de saída especificado, contendo apenas o momento, a mensagem de alerta e os endereços de IP de origem e destino, bem como as respectivas portas. É mais rápido que o método completo, pois não registra todas as informações do cabeçalho do pacote.
- ***Alert_full***: Este módulo gera os alertas com todas as informações do cabeçalho do pacote e é o modo padrão no Snort caso não seja especificado outro. Os alertas são gerados no diretório padrão `/var/log/snort` ou em outro diretório especificado na hora da execução do Snort. Nestes diretórios, subdiretórios são criados por IP com informações decodificadas dos pacotes que geraram o alerta. Uma vez que a criação dessa estrutura demanda bastante do Snort, somente é recomendado em situações de pouco tráfego.
- ***Alert_unixsock***: Este módulo cria um *socket* UNIX (KERRISK, 2010) e envia os alertas através deste canal. Outros programas ou processos podem escutar este canal e receber os alertas do Snort e os pacotes de dados em tempo real.
- ***Log_tcpdump***: Este módulo registra os pacotes em um arquivo no formato da ferramenta *tcpdump*. Este método é interessante para realizar a análise posterior dos pacotes capturados.
- ***Csv***: Este módulo permite gravar os alertas em um formato que permite a importação fácil por um banco de dados.

- ***Unified2***: Este método permite armazenar os pacotes e alertas em arquivos distintos ou em um único arquivo, e permite uma série de opções para flexibilizar os registros.
- ***Log null***: Método utilizado para gerar alertas sem o registro dos pacotes.

5.1.3 Nmap

O Nmap (NMAP, 2013) é uma ferramenta livre de código aberto bastante utilizada para atividade de auditoria e descoberta de rede. Também é bastante utilizada para atividade de inventário de rede, monitoramento de *host*, verificação de quais serviços estão sendo disponibilizados e o tempo em que estes serviços estão no ar.

Através de parâmetros passados para Nmap, pode-se realizar varreduras das mais variadas maneiras. Diversos parâmetros podem ser utilizados com o Nmap para definir o tipo de varredura desejado. A lista completa de opções pode ser verificada em seu sítio ou na documentação que acompanha a ferramenta em NMAP (2013).

Na execução do Nmap, o que não for opção ou argumento da opção é considerado como sendo especificação do *host* alvo. O alvo pode ser um ou vários *hosts* e pode ser especificado na notação CIDR (FULLER, 2006), uma lista utilizando-se o hífen para especificar uma faixa de IPs ou uma lista separada por vírgula. O(s) *host*(s) alvo(s) também podem ser definidos através de arquivos texto utilizando-se a opção "-iL". Outras opções podem ser utilizadas para excluir um ou outro *host* da varredura (NMAP, 2013).

O resultado do Nmap é uma lista das possíveis portas abertas no *host* avaliado. A Figura 18 apresenta um exemplo de um resultado de uma varredura realizada no *host* de IP 10.0.0.1. Dentre as informações apresentadas, as principais são: latência de comunicação entre o *host* varredor e *host* varrido, a quantidade total de portas consideradas fechadas, a lista de portas abertas com o nome do serviço comumente associado àquela porta e o tempo utilizado para a varredura. O estado das portas avaliadas pelo Nmap pode ser:

- ***Open***: indica que há uma aplicação aguardando conexão/pacote naquela porta.
- ***Filtered***: indica que um *firewall*, filtro ou outro obstáculo está impedindo alcançar a porta e desta forma o Nmap não tem como determinar se a porta está aberta (*open*) ou fechada (*closed*).
- ***Closed***: indica que não há aplicação alguma aguardando conexão/pacote naquela porta.

- ***Unfiltered***: indica que a porta responde às sondagens do Nmap, mas o Nmap não tem como determinar se a porta está aberta (*open*) ou fechada (*closed*).

O Nmap oferece uma variedade de opções para a descoberta e varredura de um *host*. Antes de tentar descobrir quais portas estão sendo abertas, o Nmap verifica se o *host* alvo pode ser alcançado. Caso nenhuma opção seja especificada no comando, por padrão, o Nmap envia pacotes ICMP do tipo *echo request*, pacote TCP com a *flag* SYN para a porta 443, um pacote TCP com *flag* ACK para a porta 80 e um pacote ICMP do tipo *timestamp request*, nesta sequencia, para determinar se um *host* está no ar.

```
Starting Nmap 4.62 ( http://nmap.org ) at 2013-08-04 23:48 BRT
Interesting ports on teste.example.com.br (10.0.0.1):
Not shown: 1709 closed ports
PORT     STATE SERVICE
25/tcp   open  smtp
80/tcp   open  http
111/tcp  open  rpcbind
1040/tcp open  netsaint
8000/tcp open  http-alt
9102/tcp open  jetdirect

Nmap done: 1 IP address (1 host up) scanned in 0.178 seconds
```

Figura 18 – Exemplo de resultado da execução do comando *nmap* para um *host.*

Após descobrir se o(s) alvo(s) está(rão) no ar, o Nmap passa para a etapa de coleta de informações realizando uma varredura nas portas. Caso nenhuma opção seja especificada para realizar a varredura, é feita uma varredura aleatória das mil portas mais comuns utilizando o protocolo TCP com a *flag* SYN. Ao término da varredura, ele tenta descobrir o nome do alvo fazendo uma pesquisa reversa no servidor de *Domain Name System* (DNS). Uma porta é marcada como aberta caso um pacote TCP com a *flag* SYN/ACK seja recebida e fechada caso um pacote TCP com a *flag* RST seja recebida. Caso nenhuma resposta seja recebida ou pacotes ICMP do tipo 3 (*unreachable error*) cheguem, ela é marcada como sendo *filtered.*

Diversas opções de varredura estão disponíveis e podem ser consultadas na documentação do Nmap. O Quadro 9 apresenta algumas das opções bastante utilizadas com o comando Nmap.

Opção	Descrição
-p	Especifica quais portas devem ser varridas no *host* alvo. As portas podem ser definidas individualmente separadas por vírgulas; através de listas utilizando

	hífen para definir os limites ou nomes de acordo com o especificado no arquivo *nmap-services*. O uso apenas do hífen (-) é o mesmo que definir a faixa de 1 a 65535.
-r	Por padrão, o Nmap faz a varredura das portas de forma aleatória. Esta opção instrui o Nmap a executar a varredura de forma sequencial.
-n	Instrui o Nmap a não realizar uma pesquisa reversa no servidor de DNS à procura do nome do *host*.
-Pn	Pula toda a etapa de descoberta de um *host*.
-sn	Pula a etapa de varredura das portas.
--max-retries n	Determina o número (n) de tentativas que o Nmap varrerá uma porta, na ausência de uma resposta. Por padrão, o Nmap tentará quatro vezes.

Quadro 9 – Algumas opções bastante utilizadas com o Nmap

5.2 Ambiente e metodologia do experimento

A avaliação do IPSFlow foi realizada no ambiente virtual do Mininet em um *notebook* com processador Intel[9] Core™ 2 Duo T7500 de dois núcleos de 2.2 *gigahertz*, 4 *gigabytes* de memória principal, sistema operacional Linux Ubuntu[10] na versão 12.04 LTS de 64 *bits*.

O Controlador foi executado diretamente no *notebook* e por esta razão, optou-se por instalar o Mininet diretamente neste computador em vez de utilizar a máquina virtual disponibilizado em MININET (2013). Para a geração de tráfego foi utilizado o Nmap para Linux e para o IDS utilizou-se o Snort. O Quadro 10 apresenta as versões dos softwares utilizados nos experimentos.

Uma vez que, do ponto de vista lógico, os pacotes de dados seguem um caminho linear na rede, os experimentos foram realizados em três topologias de rede (com um, dois e três *switches*) em duas configurações (com e sem o encaminhamento do tráfego para ser analisado no IDS), totalizando seis cenários. Para cada cenário foram utilizados dois *hosts*, no qual o *Host* 1 foi utilizado para realizar a varredura e o *Host* 2 foi o alvo. As Figuras 19, 20 e 21 apresentam as topologias de rede com um, três e cinco *switches* respectivamente. Em suma, os seis cenários são:

- **Cenário 1**: um *switch* conectando os *Hosts* 1 e 2, com o tráfego **não** sendo analisado pelo IDS.

[9] http://www.intel.com
[10] http://www.ubuntu.com/

- **Cenário 2**: três *switches* conectando os *Hosts* 1 e 2, com o tráfego **não** sendo analisado pelo IDS.
- **Cenário 3**: cinco *switches* conectando os *Hosts* 1 e 2, com o tráfego **não** sendo analisado pelo IDS.
- **Cenário 4**: um *switch* conectando os *Hosts* 1 e 2, com o tráfego **sendo** analisado pelo IDS.
- **Cenário 5**: três *switches* conectando os *Hosts* 1 e 2 com o tráfego **sendo** analisado pelo IDS.
- **Cenário 6**: cinco *switches* conectando os *Hosts* 1 e 2 com o tráfego **sendo** analisado pelo IDS.

Software	**Versão**
Nmap	5.21
Mininet	2.0
Snort	2.9.2, *build* 78
Ubuntu	12.04 LTS, 64 bits

Quadro 10 – Versões dos *softwares* utilizados nos experimentos.

Em cada cenário, o Snort foi conectado na *interface* 1 dos respectivos *switches* (conforme ilustram as Figuras 19, 20 e 21) e para a detecção de uma varredura na rede, foi construída uma regra que gerava alertas ao detectar 100 pacotes TCP com a *flag* TCP *SYN* tendo como IPs de origem o *Host* 1 e de destino o *Host* 2, em um intervalo de 1 segundo. A Figura 22 apresenta a regra utilizada no Snort para esta detecção, utilizando as opções *threshold* e *msg*. Para simplificar a atuação do Snort, as demais regras que acompanham a instalação foram desabilitadas no arquivo de configuração. O local de instalação do Snort não teve critério específico, pois o único requisito era que este recebesse uma cópia do fluxo passante.

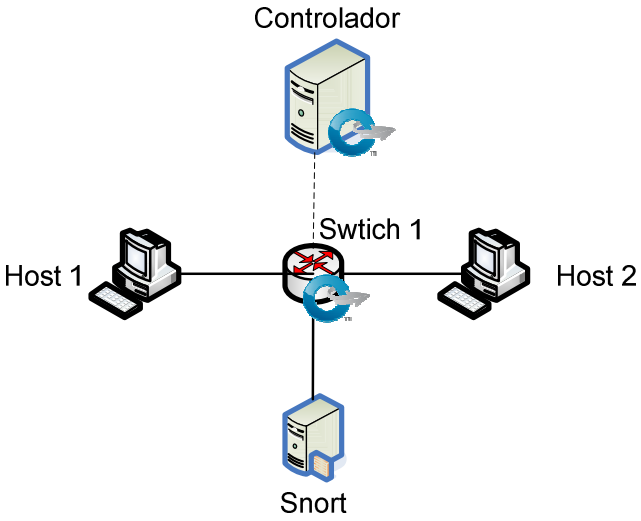


Figura 19 – Cenários 1 e 4, configuração da rede com um *switch*.

Em todos os cenários foram realizados 25 experimentos com 100 varreduras cada, totalizando 2500 amostras. Logo após cada varredura, as estatísticas dos *switches* foram coletadas, tendo como foco a quantidade de pacotes e de *bytes* trafegados na rede. Foram coletadas as quantidades de portas detectadas como fechadas e filtradas, a latência entre os dois *hosts* e o tempo de duração da varredura apresentados nos resultados gerados pelo Nmap. Os dados coletados serão apresentados e analisados no Capítulo 6.

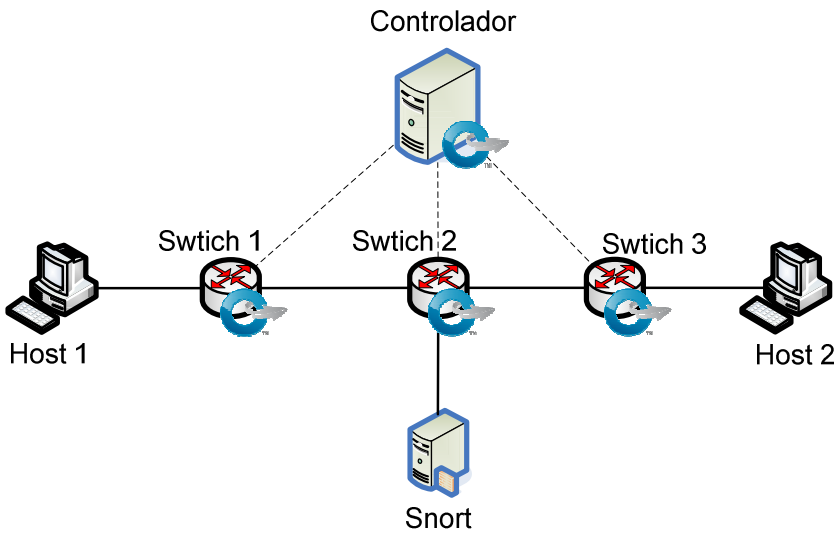


Figura 20 – Cenários 2 e 5, configuração da rede com três *switches*.

Para a varredura nos Cenários 1, 2 e 3, no qual o tráfego de varredura não foi analisado pelo IDS, utilizou-se o Nmap apenas com os parâmetros "-r" e "–n", ou seja, a varredura foi feita sequencialmente nas mil primeiras portas e a resolução do nome do alvo foi desabilitada. Estas opções foram utilizadas para facilitar no rastreamento dos pacotes e por falta de um servidor de nomes para resolver o IP do *host* para o seu respectivo nome.

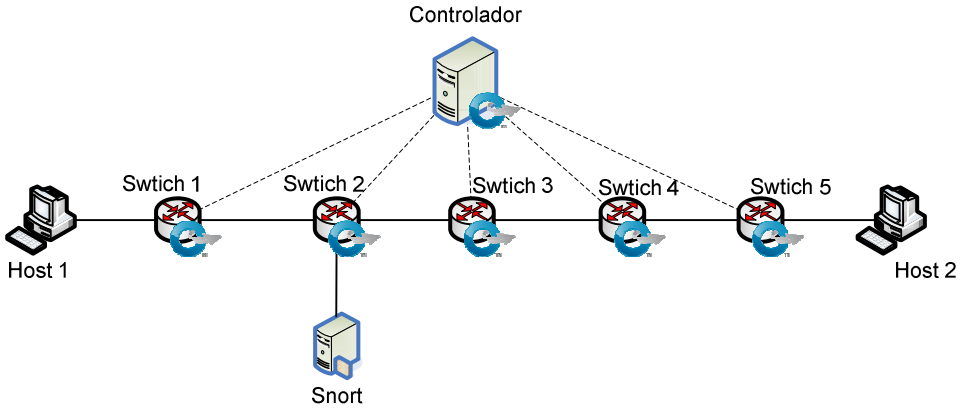


Figura 21 – Cenários 3 e 6, configuração da rede com cinco *switches*.

Já nos Cenários 4, 5 e 6, no qual o tráfego de varredura passou a ser analisado pelo IDS, como no TCP ocorre a retransmissão do pacote caso não haja resposta por parte do destinatário, além dos parâmetros dos Cenários 1,2 e 3, foi utilizada a opção "--max-retries 0", necessária para inibir o reenvio de pacotes TCP após o bloqueio do fluxo, que estava elevando o tempo de varredura para a ordem de minutos.

```
alert tcp any any -> [10.0.0.3] any (flags: S; msg:"Possible TCP
DoS"; threshold: type both, track by_src, count 100, seconds 1;
sid:1000001;rev:1;)
```

Figura 22 – Regra do Snort configurada para detectar varredura na rede.

A cada varredura detectada pelo Snort, uma mensagem de alerta era enviada para o IPSFlow. Diante do alerta recebido, o IPSFlow extraía as informações do alerta e buscava o fluxo que coincidisse com o alerta recebido. Ao encontrar o fluxo, uma mensagem OpenFlow com ação Drop era enviado para o *switch* mais próximo do *host* gerador da varredura para efetuar o bloqueio das demais varreduras.

6 Análise dos Resultados Obtidos

Os experimentos tiveram como foco principal validar o funcionamento adequado do bloqueio realizado pelo *switch* mais próximo do *host* o qual gerou o tráfego malicioso, neste caso o tráfego de varredura de portas. Para isso, foram coletadas informações como a quantidade de portas apresentadas pelo Nmap como fechadas e abertas, a duração das varreduras e a latência medida entre os dois Hosts. Também foram coletadas as quantidades de pacotes e de bytes que trafegaram nos *switches* da rede.

Através da execução dos experimentos descritos no Capítulo 5, para os cenários de 1 a 6, obtiveram-se os resultados que serão descritos neste Capítulo. Para a geração dos gráficos apresentados, calculou-se a média aritmética, o desvio padrão e o erro padrão dos valores coletados e adotou-se o intervalo de confiança de 95% (JAIN, 1991).

6.1 Cenários com o tráfego não sendo analisado pelo IDS

Os Gráficos de 1 a 8 apresentam os resultados obtidos nos Cenários 1, 2 e 3, nos quais o IPSFlow não foi habilitado. Sendo assim, conforme apresenta o Gráfico 1, todas as mil portas TCP varridas no *Host* 2 foram detectadas como fechadas nos Cenários 1, 2 e 3, ou seja, o *Host* 2 respondeu as requisições da varredura com pacotes TCP com *flag* RST habilitado.

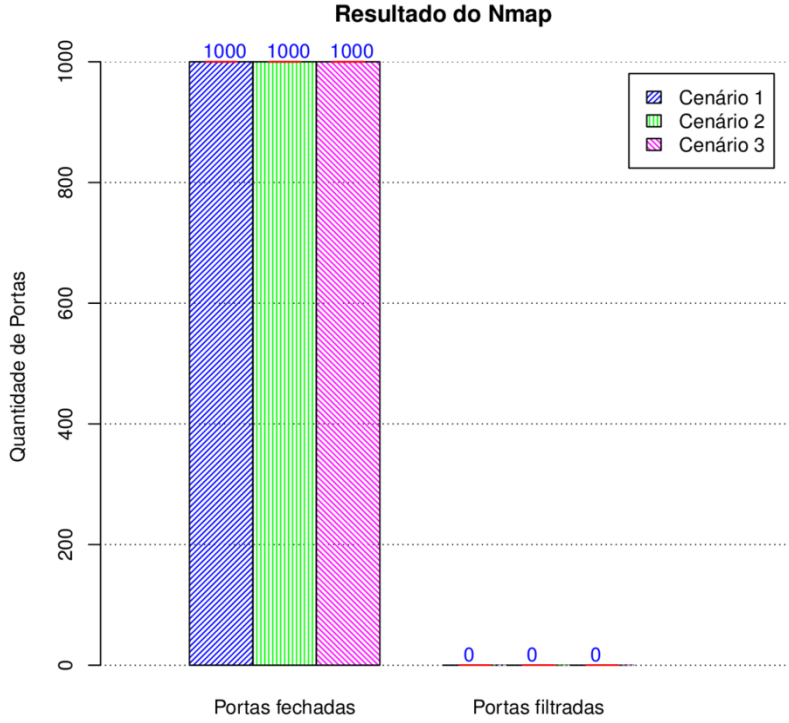


Gráfico 1 - Quantidade de portas detectadas como fechadas ou filtradas pelo Nmap nos Cenários 1, 2 e 3.

No Gráfico 2, são apresentados os tempos demandados para realizar a varredura do Nmap nos Cenários com o IPSFlow desabilitado, bem como a latência média aferida entre os dois *hosts* envolvidos nos experimentos. Uma vez que o tempo de varredura foi apresentado pelo Nmap, supõe-se que a variação apresentada entre os cenários tenha sido provocada pelo ambiente virtual do Mininet, que compartilha recursos com outros processos do *host* em que está sendo executado. Já o aumento na latência se deve a diferença na quantidade de *switches,* que aumenta de acordo com o cenário.

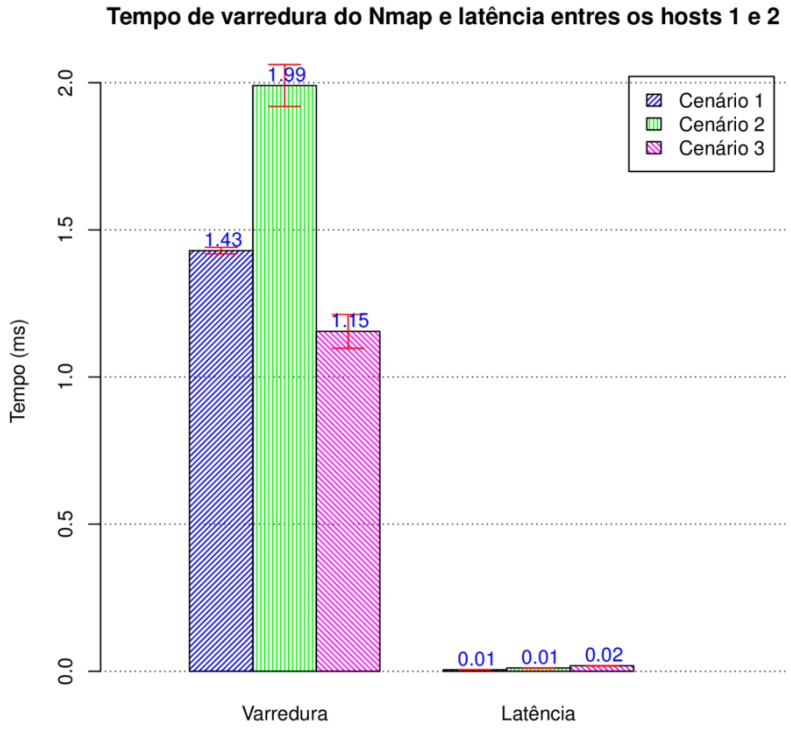


Gráfico 2 - Tempo de varredura do Nmap e latência entre os *Hosts* 1 e 2 nos Cenários 1, 2 e 3.

6.1.1 Cenário 1

Embora o Nmap tenha detectado que as mil portas do *Host* 2 estavam fechadas, verificou-se que nem todos os pacotes enviados do *Host* 1 para o *Host* 2 foram contabilizados pela Tabela de Fluxos. O Gráfico 3 apresenta a quantidade média de pacotes contabilizados nos fluxos do *Host* 1 para o *Host* 2 e vice-versa, bem como o seu somatório. Como pode ser observado, a média de pacotes enviados para o *Host* 2 ficou em torno de 983 pacotes, enquanto que a média dos pacotes recebidos pelo *Host* 1 ficou em torno dos 1000 pacotes.

Uma vez que não foram contabilizados erros ou descartes de pacotes nas portas do *switch* e foram recebidas 1000 respostas para a varredura realizada, acredita-se que os pacotes em direção ao *Host* 2 não foram contabilizados por terem sido enfileirados durante o processo de consulta ao Controlador, não sendo assim, registrados como coincidentes com o registro

recém inserido na Tabela de Fluxos. Este comportamento foi questionado na lista de discussão do Mininet, porém não se teve uma resposta conclusiva, pois possivelmente a causa do problema estaria no protocolo OpenFlow.

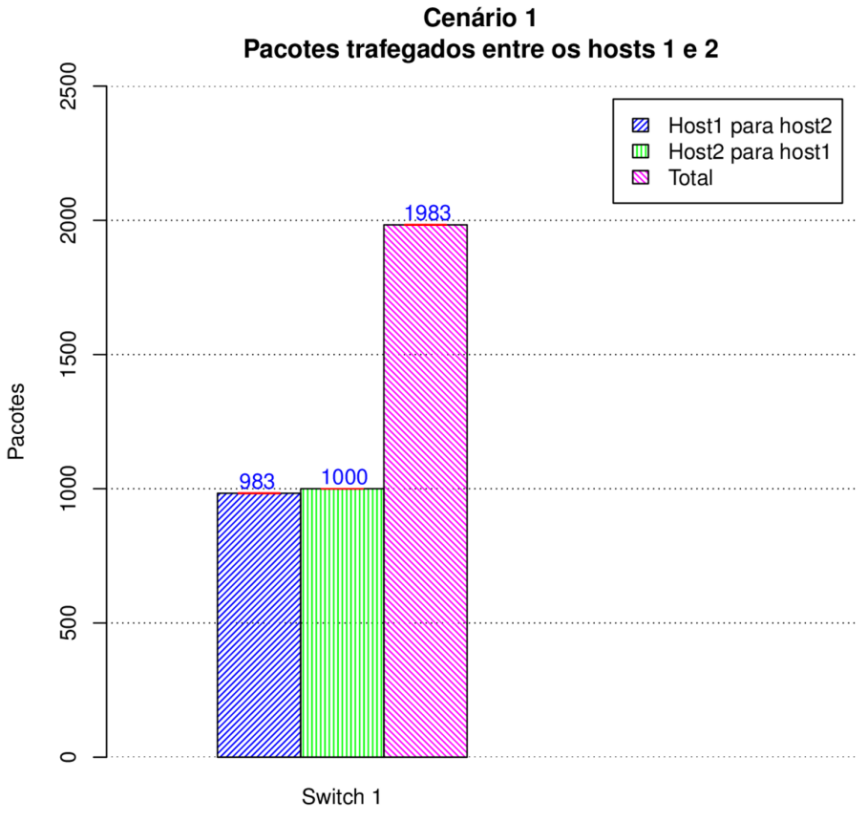


Gráfico 3 - Quantidade de pacotes trafegados entre os *Hosts* 1 e 2 no Cenário 1.

O Gráfico 4 apresenta a quantidade média de *bytes* trafegados entres os dois *hosts*, bem como o somatório do tráfego nos dois sentidos, contabilizados pelo *Switch* 1 no Cenário 1. O tráfego no sentido do *Host* 2 possui ligeiro aumento, pois os pacotes neste sentido foram de 58 *bytes* e os de respostas foram de 54 *bytes*.

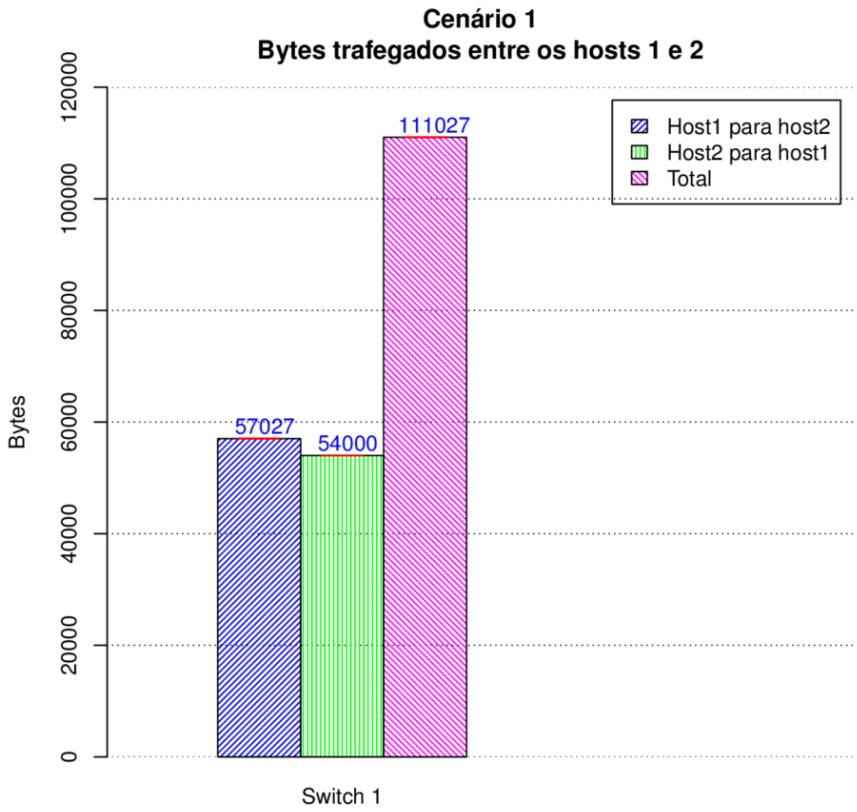


Gráfico 4 - Quantidade de *bytes* trafegados entre os *Hosts* 1 e 2 no Cenário 1.

6.1.2 Cenário 2

De forma semelhante ao Gráfico 3, o Gráficos 5 apresenta a quantidade de pacotes trafegados que foram contabilizados pelos *switches* no Cenários 2. Neste cenário também foi constatada a não contabilização de alguns pacotes no sentido do *Host* 1 para o *Host* 2. A pequena variação na quantidade de pacotes é por causa da presença de outros protocolos como o *Address Resolution Protocol* (ARP), necessário para a localização de *hosts* em uma rede baseada em TCP/IP.

Também de forma semelhante ao Gráfico 4, o Gráfico 6 exibe a quantidade de *bytes* equivalente ao número de pacotes que foram contabilizados nos *switches* que compõem o Cenários 2. Nestes cenários também foi constatado a diferença de tamanho nos pacotes dos dois sentidos entre os *Hosts* 1 e 2.

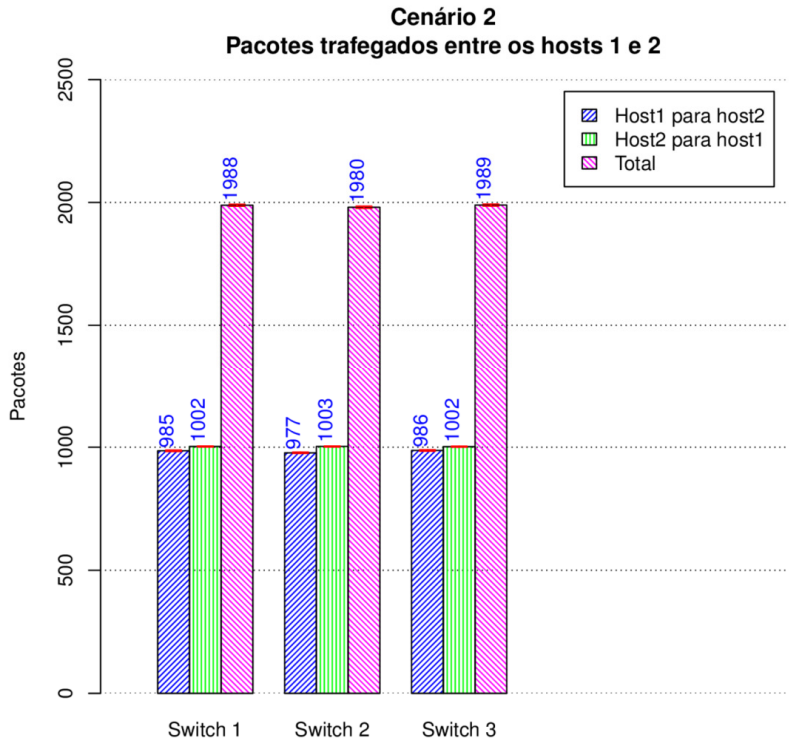


Gráfico 5 - Quantidade de pacotes trafegados entre os *Hosts* 1 e 2 no Cenário 2.

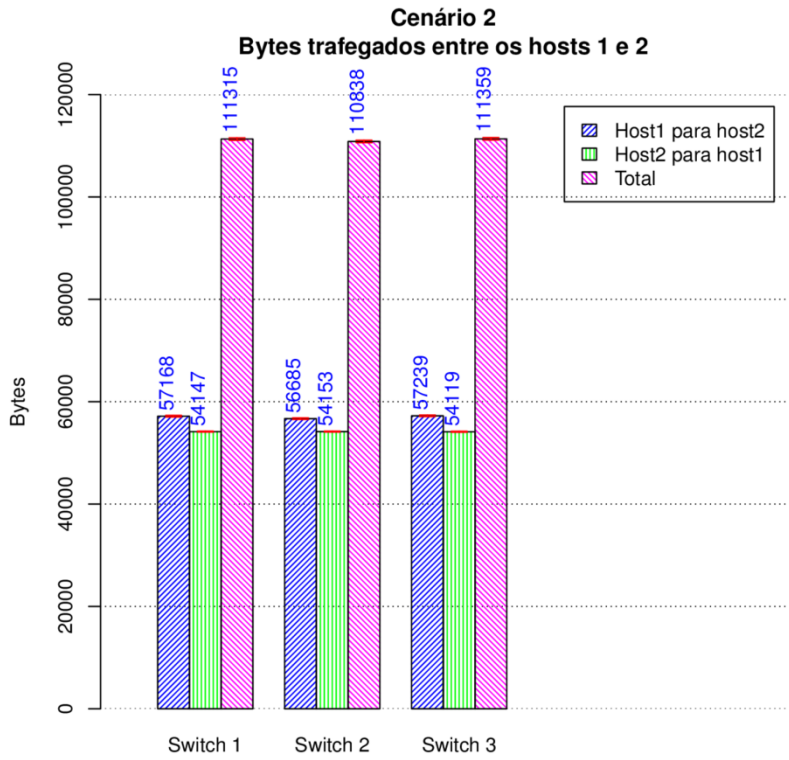


Gráfico 6- Quantidade de *bytes* trafegados entre os *Hosts* 1 e 2 no Cenário 2.

6.1.3 Cenário 3

O Gráfico 7 apresenta a quantidade de pacotes trafegados que foram contabilizados pelos *switches* no Cenário 3. Analogamente aos Cenários 1 e 2, neste cenário também foi constatada a não contabilização de alguns pacotes no sentido do *Host* 1 para o *Host* 2. Novamente é possível perceber uma pequena variação na quantidade de pacotes, provocada pela presença de outros protocolos como o *Address Resolution Protocol* (ARP), necessário para a localização de *hosts* em uma rede baseada em TCP/IP.

De forma semelhante aos Gráficos 4 e 6, o Gráfico 8 exibe a quantidade de *bytes* equivalente ao número de pacotes que foram contabilizados nos *switches* que compõem o Cenários 3. Neste cenário também foi constatado a diferença de tamanho nos pacotes dos dois sentidos entre os *Hosts* 1 e 2.

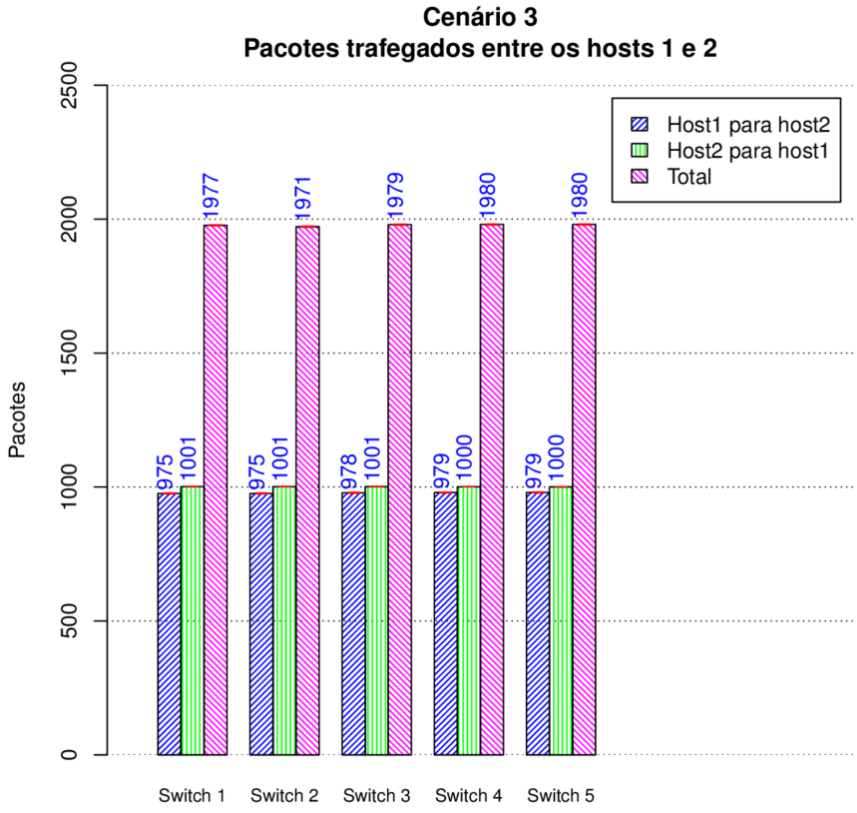


Gráfico 7- Quantidade de pacotes trafegados entre os *Hosts* 1 e 2 Cenário 3.

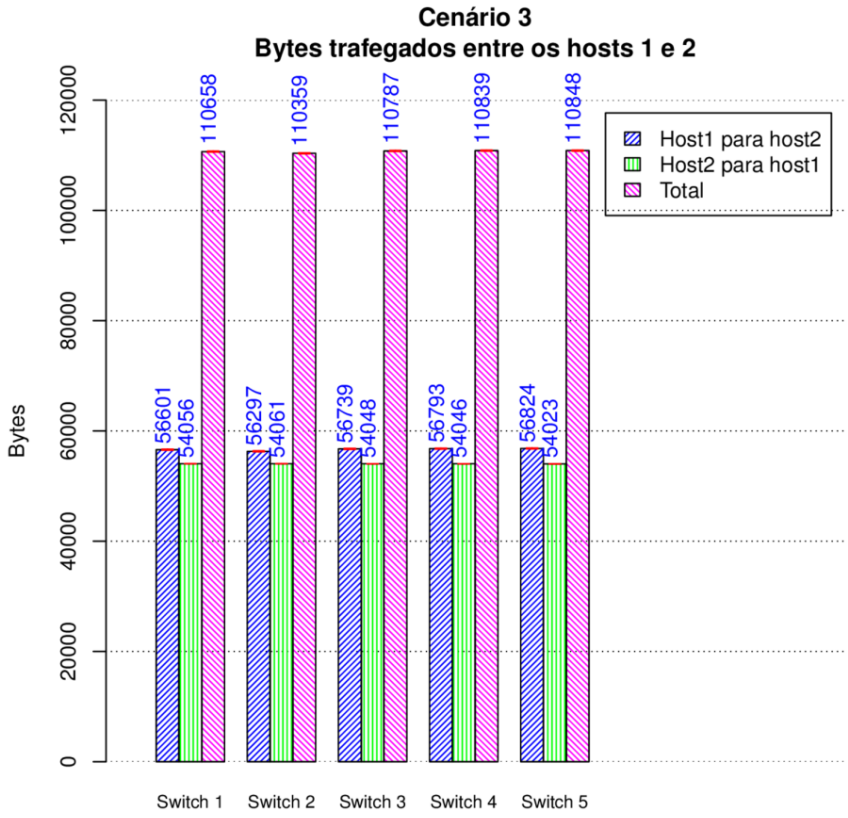


Gráfico 8- Quantidade de *bytes* trafegados entre os *Hosts* 1 e 2 Cenário 3.

6.2 Cenários com o tráfego sendo analisado pelo IDS

Os Gráficos de 9 a 16 são referentes aos Cenários 4, 5 e 6, nos quais o IPSFlow foi habilitado. O Gráfico 9 apresenta a quantidade de portas detectadas como fechadas e filtradas pelo Nmap. Uma vez que o Nmap reporta como filtradas as portas nas quais não recebeu nenhuma resposta, conclui-se que o IPSFlow efetuou o bloqueio de boa parte dos pacotes utilizados na varredura do *host* 2. Com o uso do IPSFlow, conseguiu-se bloquear 85% dos pacotes da varredura, índice este relacionado diretamente com a configuração da regra configurada no Snort.

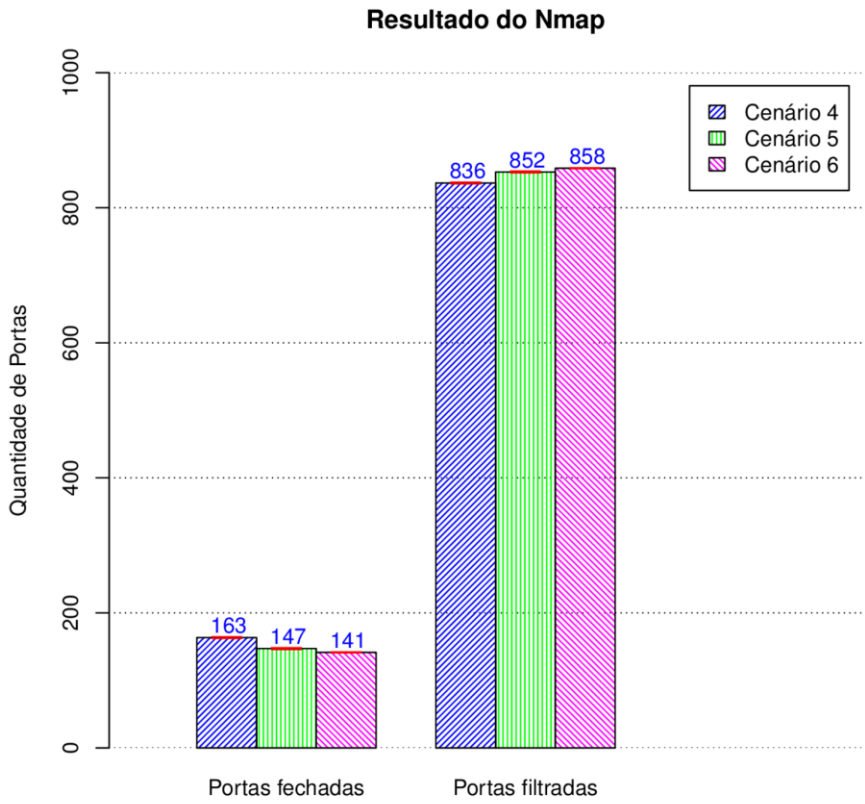


Gráfico 9 - Quantidade de portas detectadas como fechadas ou filtradas pelo Nmap nos Cenários 4, 5 e 6.

No Gráfico 10, são apresentados os tempos demandados para realizar a varredura do Nmap nos Cenários com o IPSFlow habilitado (Cenários 4, 5 e 6), bem como a latência média registrada na comunicação entre os *Hosts* 1 e 2. A diferença dos tempos de varredura apresentados nos Gráficos 2 e 10 para os três Cenários aumentam mais a possibilidade das variações registradas serem provocadas pelo Mininet. Já para o tempo de latência, o mesmo comportamento se registra para os Cenários 1, 2 e 3.

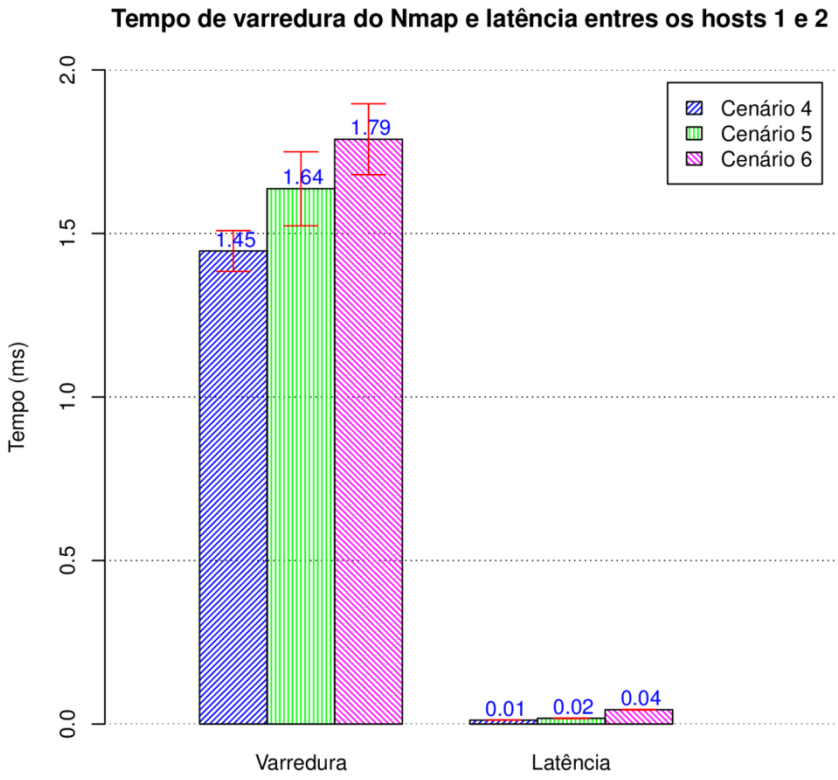


Gráfico 10 - Tempo de varredura do Nmap e latência entre os *Hosts* 1 e 2 nos Cenários 4, 5 e 6.

6.2.1 Cenário 4

Os Gráficos 11 e 12 apresentam a quantidade de pacotes e de *bytes* trafegados no único *switch* do Cenário 4, originadas no *Host* 1 e destinadas para o *Host* 2, originadas no *Host* 2 e destinadas para o *Host* 1, e suas somatórias. Nestes gráficos, é importante notar que o volume de tráfego no sentido do *Host* 1 para o *Host* 2 é em torno de 6 vezes maior que o volume no sentido oposto, pois mesmo bloqueando os pacotes do fluxo no *switch*, este continua recebendo os pacotes e os contabilizando nos respectivos contadores da Tabela de Fluxos.

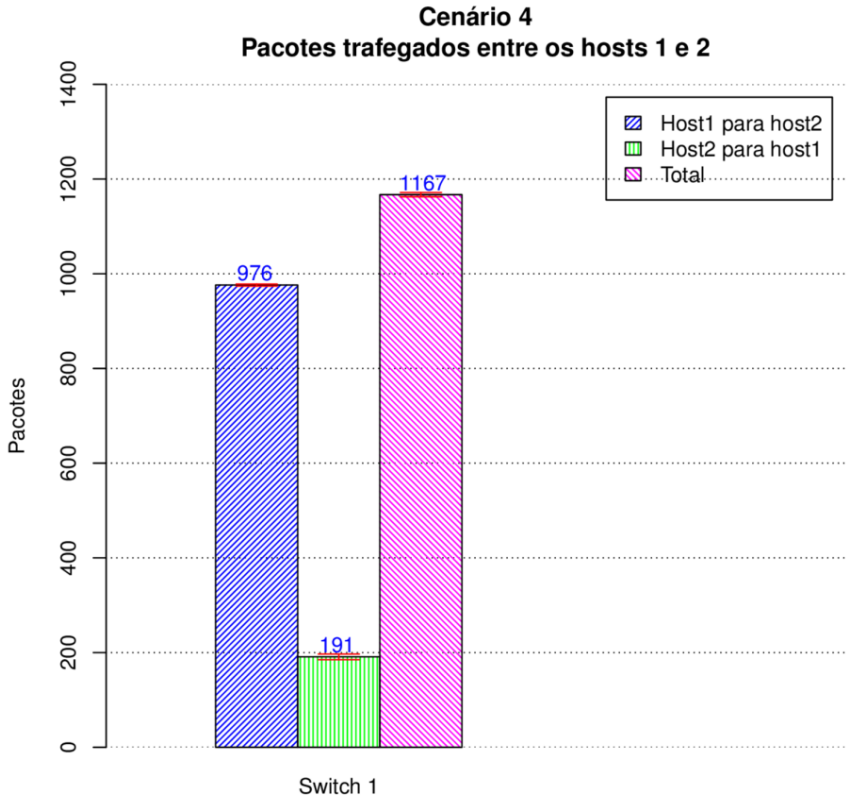


Gráfico 11 - Quantidade de pacotes trafegados entre os *Hosts* 1 e 2 Cenário 4.

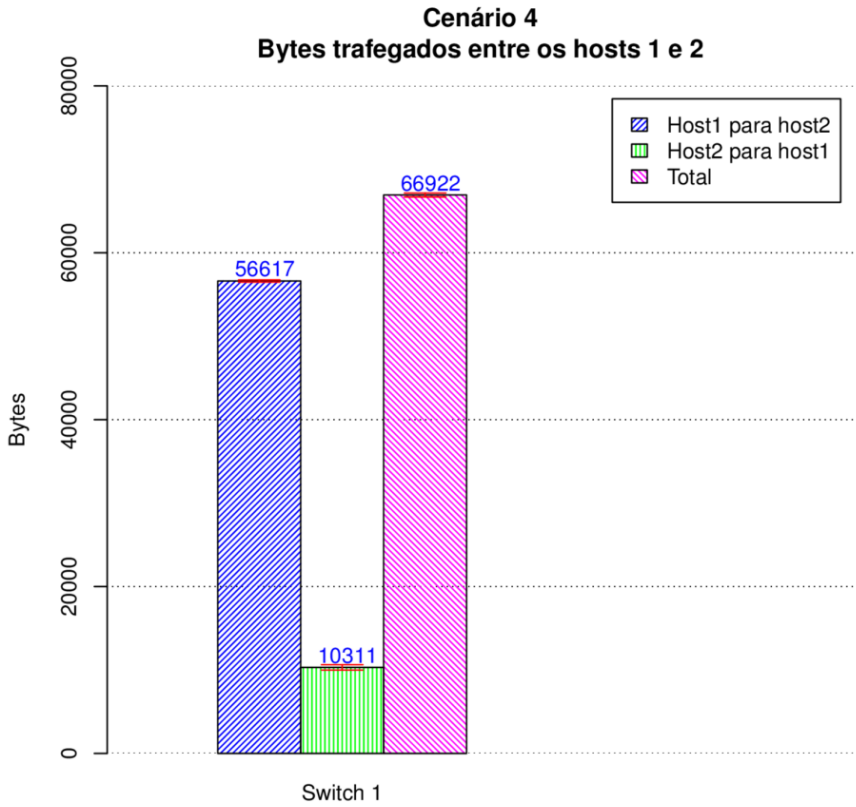


Gráfico 12 - Quantidade de *bytes* trafegados entre os *Hosts* 1 e 2 Cenário 4.

A capacidade de efetuar o bloqueio no *switch* mais próximo do originador da varredura pode ser evidenciada melhor nos Gráficos 13, 14, 15 e 16, onde o *switch* 1, no qual o *Host* 1 estava diretamente conectado, registrou uma grande quantidade de pacotes enquanto que os demais registraram apenas o montante de tráfego anterior ao bloqueio.

6.2.2 Cenário 5

De forma idêntica ao Gráfico 11, o Gráfico 13 apresenta a quantidade de pacotes que foram registrados nos *switches* do cenário 5. O bloqueio efetuado no *switch* mais próximo do *Host* 1 pode ser melhor evidenciado neste gráfico, pois apenas parte do tráfego destinado ao *Host* 2 é registrado nos *Switches* 2 e 3.

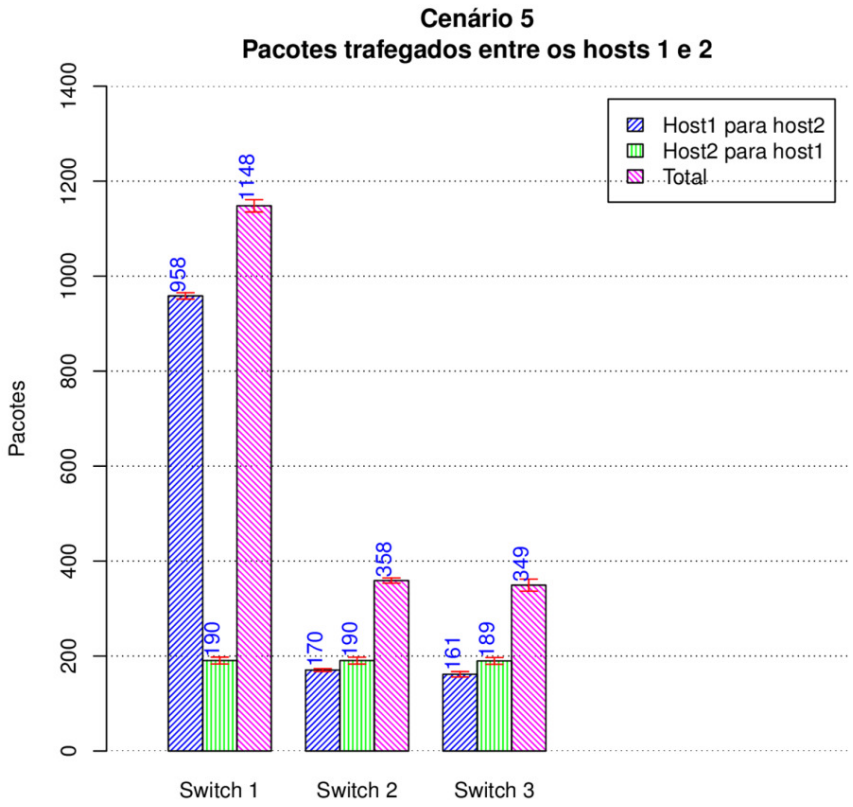


Gráfico 13 - Quantidade de pacotes trafegados entre os *Hosts* 1 e 2 no Cenário 5.

Também de forma semelhante ao Gráfico 12, o Gráfico 14 apresenta a quantidade de *bytes* equivalente ao número de pacotes que foram contabilizados nos *switches* que compõem o Cenários 5.

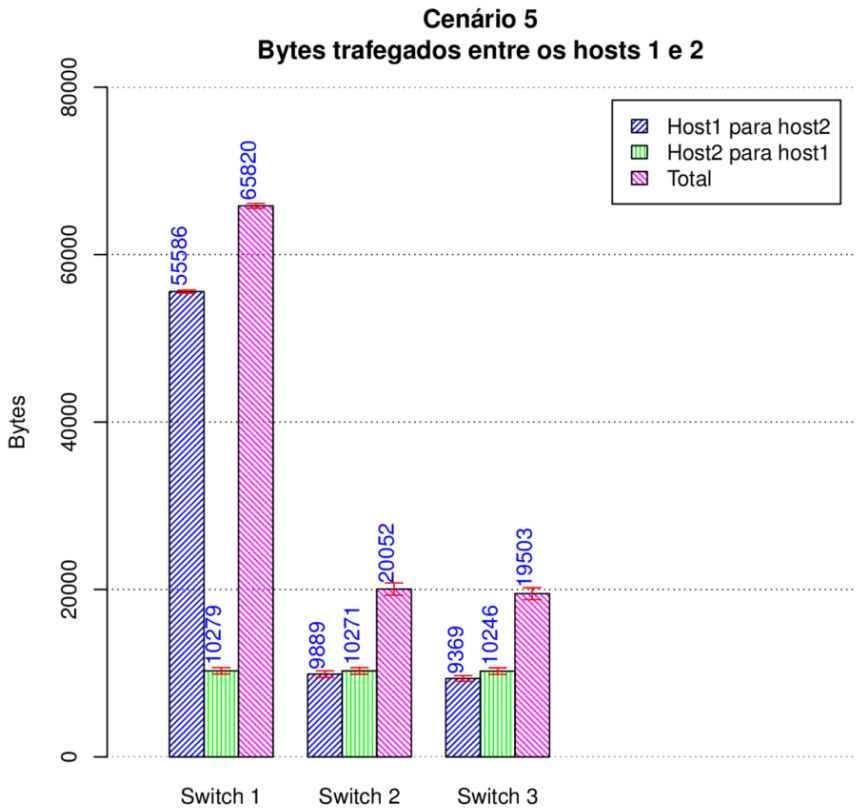


Gráfico 14 - Quantidade de *bytes* trafegados entre os *Hosts* 1 e 2 Cenário 5.

6.2.3 Cenário 6

Para o Cenário 6, registrou-se a quantidade de pacotes e de *bytes* apresentados nos Gráficos 15 e 16, respectivamente, que foram registrados nos *switches* do cenário 6. Novamente pode ser observado que o bloqueio do tráfego de varredura foi realizado no *switch* no qual o *Host* 1 estava diretamente conectado, no caso, o *Switch* 1.

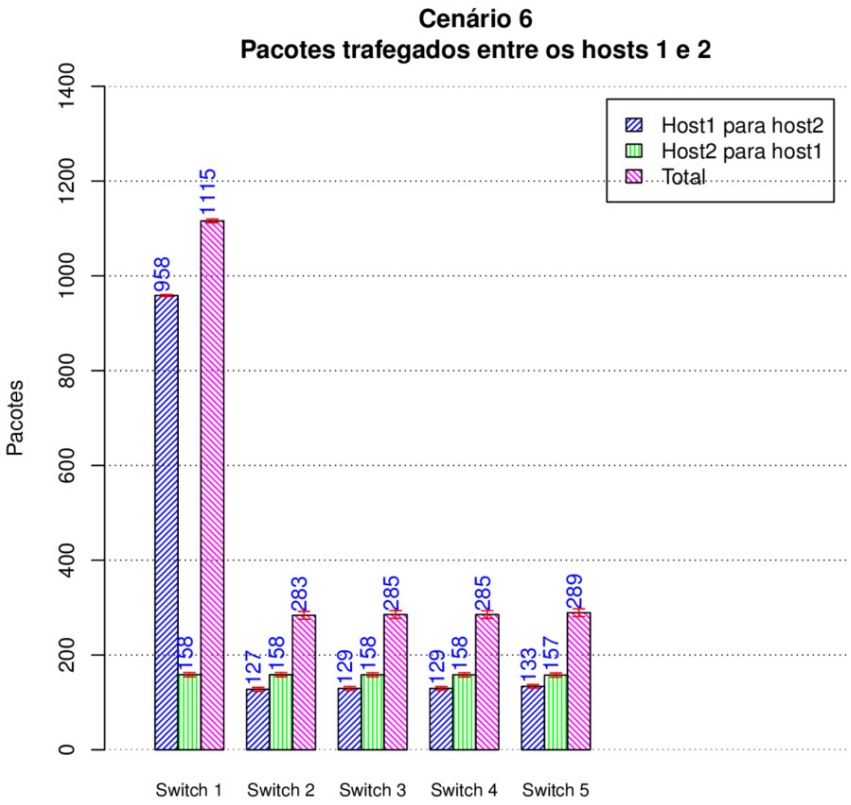


Gráfico 15 - Quantidade de pacotes trafegados entre os *Hosts* 1 e 2 no Cenário 6.

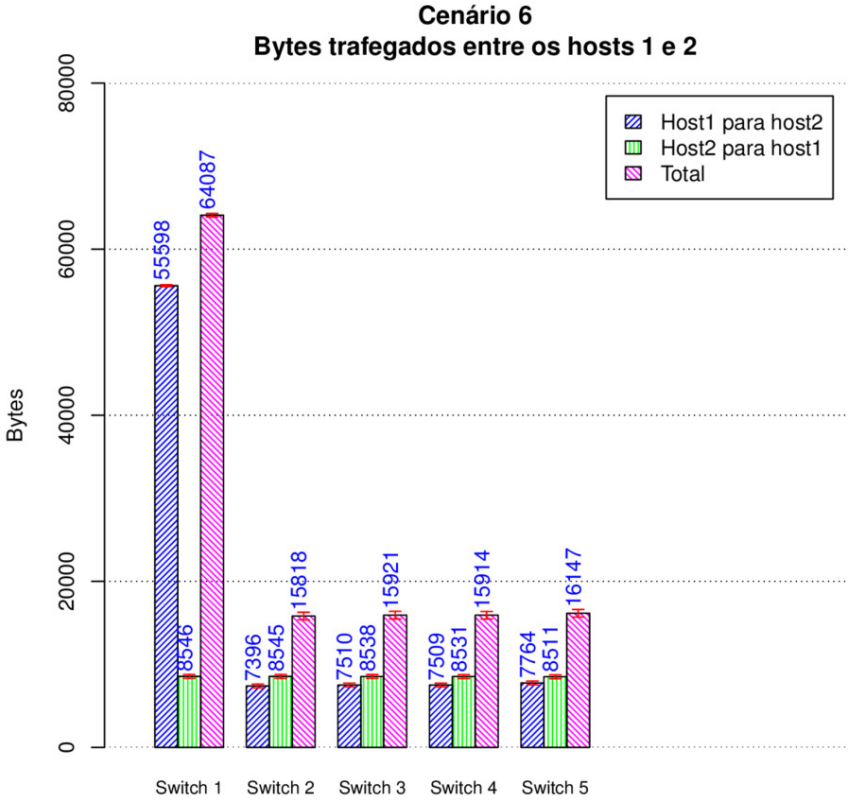


Gráfico 16 - Quantidade de *bytes* trafegados entre os *Hosts* 1 e 2 no Cenário 6.

Diante dos Gráficos de 9 a 16, percebe-se que há uma discrepância entre a quantidade de portas detectadas como fechadas no Snort e a quantidade de pacotes registrados do *Host* 2 para o *Host* 1 nos Cenários 4, 5 e 6. Além disso, também não foram registrados os 990 pacotes no *switch* 1 como foram contabilizados nos Cenários 1, 2 e 3.

Tomando como base os resultados apresentados pelo Nmap no Gráfico 9 e levando em consideração a não contabilização de todos os pacotes nos Cenários 1, 2 e 3, acredita-se que todas as discrepâncias registradas ao longo dos experimentos estejam relacionadas ao Mininet. Porém, vale lembrar que o objetivo principal do IPSFlow é bloquear um ataque, ou um possível ataque, no ponto mais próximo da origem. Desta forma, a exatidão das estatísticas acaba se tornando de menor relevância para os experimentos.

Sendo assim, de acordo com os gráficos apresentados, a execução dos experimentos revelou o funcionamento adequado do IPSFlow, uma vez que cumpriu com o objetivo de efetuar o bloqueio do tráfego no *switch* mais próximo de sua origem tomando como base um alerta gerado por um IDS.

7 Considerações Finais

Com o crescimento das redes, a diversificação dos dispositivos de rede utilizados e o aumento de usuários conectados em rede, o aumento de incidentes de segurança tende a ser inevitável. Desta forma, cada vez mais ferramentas como IDSs e IPSs se fazem necessárias na administração da segurança de uma rede. Embora sejam ferramentas reconhecidas, as pesquisas neste ramo têm como foco principal as técnicas de análise.

Diante do iminente crescimento dos incidentes de segurança, mecanismos de resposta automática se fazem cada vez mais necessários para reduzir os impactos causados por um ataque. Porém, a fraca integração entre os IDSs/IPSs com a maior parte dos equipamentos de rede continua sendo um grande obstáculo, pois a implementação de novas funcionalidades em *switches* e roteadores somente pode ser realizado por seus fabricantes.

A elaboração de padrões de SDN, como o OpenFlow, surgiu como uma grande oportunidade, pois retiraram esta grande dependência que pesquisadores e administradores de redes tinham com os fabricantes de equipamentos de rede, permitindo assim que novas soluções pudessem surgir de acordo com as necessidades de cada um. A flexibilidade das SDNs não trouxe apenas a oportunidade de criar novas propostas ou soluções, trouxe também a oportunidade de melhorar soluções já existentes, como foi o caso do IPSFlow em relação aos IDSs e IPSs nas redes convencionais, uma vez que surgiu a partir da possibilidade de analisar o problema sob uma perspectiva diferente.

Por ser baseado no Openflow, o IPSFlow permite uma administração simplificada e centralizada, com solução aberta e flexível no qual permite que a análise do tráfego possa ser realizada por um ou mais IDSs. Com atuação em todos os *switches* da rede, o IPSFlow permite ampliar a cobertura de um IDS, sem dedicar portas do *switch* para espelhamento de tráfego, e realizar o bloqueio do tráfego malicioso o mais próximo de sua origem.

Embora os experimentos realizados para esta dissertação possam indicar um bom desempenho do IPSFlow, os índices de bloqueio registrados são diretamente relacionados com a resposta dada pelo IDS para detecção de um tráfego malicioso. No caso do Snort utilizado nos experimentos, estes índices estão relacionados com a regra elaborada.

Vale lembrar que o objetivo principal desta dissertação não foi propor o IPSFlow para detectar ou realizar um bloqueio no menor tempo possível. Mas sim, para realizar o bloqueio mediante o recebimento de um alerta de um IDS. Desta forma, conclui-se que em seu estágio inicial, o IPSFlow cumpriu com as expectativas ao permitir que o bloqueio do tráfego fosse realizado o mais próximo de sua origem.

O desenvolvimento do IPSFlow demandou grande esforço e pesquisa nos códigos fontes do Floodlight, pois verificou-se que este Controlador carece de uma documentação mais detalhada de seu funcionamento interno. Percebeu-se também que o nível de experiência na linguagem Java e no ambiente de desenvolvimento Eclipse impactam diretamente no desenvolvimento de aplicações para o Floodlight. Devido a pouca experiência para o desenvolvimento do IPSFlow, pretende-se disponibilizar o código utilizado para a criação de uma comunidade de desenvolvimento.

Na ausência de equipamentos com suporte a OpenFlow, o Mininet acaba sendo uma boa opção para validar as funcionalidades de uma proposta. Porém, diante do comportamento estranho apresentado nesta dissertação, acredita-se que o Mininet não seja um bom ambiente de validação quando se deseja certa precisão dos resultados.

Devido o isolamento dos dispositivos no Mininet, a sincronização de atividades entre o ambiente virtual e o *host* real se torna um complicador, pois só pode ser realizada através do uso do sistema de arquivos ou da hora do sistema.

No decorrer do desenvolvimento do IPSFlow, foram gerados dois artigos publicados cujas comprovações estão nos Anexos: um publicado no III Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF) ocorrido em 2012 em Outro Preto-MG e outro no XII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg) ocorrido em 2012 em Curitiba. Com os resultados coletados nesta dissertação e nos próximos experimentos, pretende-se submeter novos artigos em eventos acadêmicos nacionais e internacionais.

7.1 Trabalhos Futuros

Durante o desenvolvimento desta dissertação e a realização dos experimentos, foram detectadas algumas limitações e melhorias que se tem como tarefas para os trabalhos futuros, os quais serão detalhados a seguir.

Diante dos valores coletados nos experimentos, inicialmente, tem-se para trabalho futuro a construção de um *testbed* com equipamentos OpenFlow para a avaliação do IPSFlow e comparação dos dados obtidos nos ambientes virtual e real.

Embora a proposta do IPSFlow preveja o uso de diversos IDSs, seja para balanceamento de carga ou para análise por IDSs especializados, nesta versão inicial, só implementou-se e utilizou-se um IDS. Desta forma, há necessidade de desenvolver o encaminhamento do tráfego para múltiplos IDSs, bem como elaborar um mecanismo de

correlação ou tomada de decisão nas situações onde mais de um IDS gere alertas para um mesmo fluxo.

Dado que este trabalho teve como foco validar o funcionamento do IPSFlow diante de um alerta gerado por um IDS, utilizou-se a detecção simples de varreduras de portas para esta validação. Desta forma, devido a infinidade de ataques existentes, também se tem como trabalho futuro a avaliação do IPSFlow diante de outros ataques como DoS, DDoS etc.

Uma vez que o bloqueio de um fluxo pode implicar na geração de outro fluxo, alocar filas de prioridade mais baixas que reduzam as taxas de transferência de um fluxo pode impedir a criação de um segundo fluxo. Desta forma, seria interessante avaliar e verificar a integração do IPSFlow com o QoSFlow proposto por Ishimori (2012), no qual permite dinamicamente alocar filas de prioridades distintas para os fluxos em vez de bloqueá-los de imediato e incentivar uma ação de evasão por parte do atacante.

Poderá haver situações em que o administrador deva ser notificado ou consultado para a tomada de uma decisão. Sendo assim, mecanismos adicionais de troca de mensagens deste tipo deve ser elaborado no Controlador IPSFlow. Do mesmo modo, uma ferramenta gráfica de administração que permita o administrador interagir com o Controlador IPSFlow se faz necessária.

REFERÊNCIAS

BAKER, A. R.; CASWELL, B.; POOR, M. **Snort 2.1 Intrusion Detection**. 2 ed. Syngress, 2004.

BALLARD, J.; RAE, I.; AKELLA, A. Extensible and Scalable Network Monitoring Using OpenSAFE. In: Proceedings of 2010 INTERNET NETWORK MANAGEMENT WORKSHOP / WORKSHOP ON RESEARCH ON ENTREPRISE NETWORKING. 2010, San Jose, CA. USENIX Association Berkeley, 2010. p. 8.

BEACON. Disponível em: < https://openflow.stanford.edu/display/Beacon/Home >. Acesso em: 10 fev. 2013.

BLUM, R., BRESNAHAN, C. **Linux Command Line and Shell Scripting**. 2 ed. Wiley Publishing, Inc, 2011.

BRAGA, R. S.; MOTA, E.; PASSITO, A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proceedings of 2010 IEEE 35TH CONFERENCE ON LOCAL COMPUTER NETWORKS. 2010, Denver, CO. IEEE, 2010. p. 408-415.

CERT: Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. Disponível em: <http://www.cert.br>. Acesso em: 25 mai. 2013.

DEBAR, H.; CURRY, D.; FEINSTEIN, B. **The Intrusion Detection Message Exchange Format**. 2007. Disponível em: <http://tools.ietf.org/html/rfc4765>. Acesso em 20 dez. 2012.

DEBUS, Keith. **Snort Rule Writing for the IT Professional: Part 2**. Disponível em: <http://resources.infosecinstitute.com/snort-rule-writing-for-the-it-professional-part-2-2/> . Acesso em: 24 jun. 2013.

DEITEL, P. J.; Deitel, H. M. **Java**: How to Program. 9 ed. Prentice Hall, 2012.

FLOODLIGHT. Disponível em: <http://floodlight.openflowhub.org>. Acesso em: 17 dez. 2012.

FULLER, V. **Classless Inter-domain Routing (CIDR)**: The Internet Address Assignment and Aggregation Plan. 2006. Disponível em: <http://www.rfc-editor.org/rfc/rfc4632.txt>. Acesso em: 15 jul. 2013.

GUDE, N.; et al. NOX**:** towards an operating system for networks. In: Newsletter of ACM SIGCOMM COMPUTER COMMUNICATION REVIEW. 2008, New York, NY, USA, ACM, v. 38, n. 3, p. 105-110.

HORSTMANN, C. S.; Cornell, G. **Core Java.** 9 ed, v. 1. Prentice Hall, 2012.

HP Networking OpenFlow Workshop. Disponível em: <http://eventos.redclara.net/indico/getFile.py/access?contribId=1&resId=3&materialId=slides&confId=197> . Acesso em 20 jun. 2013.

ISHIMORI, A., SALVATTI, J., FARIAS, F., GASPARY, L., GRANVILLE, L., CERQUEIRA, E., ABELEM, A. **QoSFlow**: Gerenciamento Automático da Qualidade de Serviço em Infraestruturas de Experimentação Baseadas em Framework OpenFlow. In: III Workshop de Pesquisa Experimental da Internet do Futuro – WPEIF. Anais. Ouro Preto, MG. SBC, 2012.

JAHANGIRI, A. **How to Test Snort with Penetration Testing Tools**. Disponível em: <http://blog.alijahangiri.org/2012/04/how-to-test-snort-with-penetration-testing-tools/>. Acesso em: 25 jun. 2013.

JAIN, R. **The art of computer systems performance analysis**: techniques for experimental design, measurement, simulation and modeling", Ed. Wiley, England, 1991.

JAVA e Orientação a Objetos. Disponível em: <http://www.caelum.com.br/apostila-java-orientacao-objetos/>. Acessado em: 01 mai 2013.

KERRISK, M. **The Linux programming interface**: a Linux and UNIX system programming handbook. San Francisco. No Starch Press, 2010.

KOHLENBERG, T; et al. **Snort**: IDS and IPS Toolkit. Burlington, MA. Syngress, 2007.

KUROSE, J. F.; Ross K. W. **Redes de Computadores e a Internet: uma abordagem top-down**. Addison Wesley, 5 ed, 2010.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of 9TH ACM SIGCOMM WORKSHOP ON HOT TOPICS IN NETWORKS. New York, NY, USA. ACM, 2010.

LANTZ, B., Handigol, N. Heller, B., Jeyakumar, V. **Introduction to Mininet**. Disponível em: **<**https://github.com/mininet/mininet/wiki/Introduction-to-Mininet> . Acesso em: 28 jun. 2013.

MAESTRO. Disponível em: <https://code.google.com/p/maestro-platform/>. Acesso em: 10 fev. 2013.

MCKEOWN, N.; et. al. OpenFlow**:** enabling innovation in campus networks. In: Newsletter of ACM SIGCOMM COMPUTER COMMUNICATION REVIEW. New York, NY, USA, v. 38, n. 2, ACM, 2008. p. 69-74.

MEHDI, S; KHALID, J.; KHAYAM, S. Revisiting Traffic Anomaly Detection Using Software Defined Networking. In: Proceedings of 14TH INTERNATIONAL SYMPOSIUM, RAID 2011. Menlo Park, CA, USA, v. 6961, Springer, 2011. p. 161-180,

MININET: An Instant Virtual Network on your Laptop (or other PC). Disponível em: http://mininet.org/ Acesso em: 10 mar. 2013.

MUKHOPADHYAY, I.; CHAKRABORTY, M.; CHAKRABARTI, S. A Comparative Study of Related Technologies of Intrusion Detection & Prevention Systems. In: JOURNAL OF INFORMATION SECURITY. v. 2. 2011. p. 28-38.

NMAP. Disponível em: <http://nmap.org/> Acesso em: 9 abr. 2013.

NOX. Disponível em: <http://www.noxrepo.org/nox/about-nox/>, Acesso em: 10 fev. 2013.

OPENFLOW. The OpenFlow Switch Specification. Disponível em: <http://OpenFlowSwitch.org>. Fevereiro 2011. Acesso em 27 set. de 2011.

OPENFLOW Switch Specification. 2009. Disponível em: <http://www.openflow.org/documents/openflowspec-v1.0.0.pdf>. Acesso em: 10 jan. 2013.

OPENFLOW Tutorial. Disponível em: <http://www.openflow.org/wk/index.php/OpenFlow_Tutorial>. Acesso em: 10 jan. 2012.

OPPENHEIMER, P. **Top-Down Network Design**, 3 ed. Cisco Press, 2010.

PANDA, M.; ABRAHAM, A.; PATRA, M. R. A Hybrid Intelligent Approach for Network Intrusion Detection. In: INTERNATIONAL CONFERENCE ON COMMUNICATION TECHNOLOGY AND SYSTEM DESIGN. 2011. v. 30. Procedia Engineering, 2012, p. 1-9.

POSTEL, J. **Internet Control Message Protocol**. 1981. Disponível em: <http://tools.ietf.org/html/rfc792>. Acesso em: 15 jul. 2013.

POSTEL, J. **Transmission Control Protocol**. 1981. Disponível em: <http://tools.ietf.org/html/rfc793>.Acesso em: 15 jul. 2013.

POX. Disponível em: <http://www.noxrepo.org/pox/about-pox/>, Acesso em: 10 fev. 2013.

REHMAN, R. U. **Intrusion Detection Systems with Snort:** Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID. Upper Saddle River, New Jersey. Prentice Hall PTR, 2003.

SCARFONE, K.; MELL, P. **Guide to Intrusion Detection and Prevention Systems (IDPS)**. 2007. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf>. Acesso em 20 jun 2012.

SNORT. Disponível em: <http://www.snort.org/> Acesso em: 1 mar. 2013.

SNORT Users Manual 2.9.5. 2006. Disponível em: <http://s3.amazonaws.com/snort-org/www/assets/166/snort_manual.pdf>. Acesso em: 15 jun. 2013.

SNYDER, J. **Guide to Network Intrusion Prevention Systems**. 2008. Disponível em: <http://www.pcworld.com/businesscenter/article/144634/guide_to_network_intrusion_prevention_systems.html>. Acesso em: 20 fev. 2012.

SOFTWARE-Defined Networking: The New Norm for Networks. Disponível em: <https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf>. Acesso em 13 abr. 2012

STALLINGS, W. **Cryptography and Network Security Principles and Practice.** 5 ed. Prentice Hall, 2011.

TANENBAUM, A. S. **Redes de Computadores**. 4 ed. Elsevier , 2003.

WIRESHARK. Disponível em: <http://www.wireshark.org/>. Acesso em: 15 jun. 2013.

XING, T., Huang, D., Xu, L., Chung, C., Khatkar, P. **SnortFlow**: A OpenFlow-based Intrusion Prevention System in Cloud Environment. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6601422>. Acesso em 7 set. 2013.

Anexo A - E-mail de aceitação do artigo aceito no III WPEIF.

From: SBRC 2012 - WPEIF <marcosrs@cpqd.com.br>
To: [redacted]
Cc: Lisandro Zambenedetti Granville <[redacted]@inf.ufrgs.br>; Luciano Paschoal Gaspary <[redacted]@inf.ufrgs.br>; Antônio Abelém <[redacted]@ufpa.br>; Elisangela Aguiar <[redacted]@serpro.gov.br>; Fernando Farias <[redacted]@ufpa.br>; Eduardo Cerqueira <[redacted]@ufpa.br>
Sent: Friday, March 30, 2012 11:45 PM
Subject: Your SBRC 2012 - WPEIF paper 97642

Dear Mr. Fábio Nagahama:

Congratulations - your paper "IPSFlow - Uma Proposta de Sistema de Prevenção de Intrusão Baseado no Framework OpenFlow" for SBRC 2012 - WPEIF has been accepted.

The reviews are below or can be found at submissoes.sbc.org.br/PaperShow.cgi?m=97642

Please address the reviews and upload the camera-ready version into JEMS preferably by March 31 and no longer than April 2.

The copyright form (filled and signed) must also be uploaded to JEMS with the camera-ready version of your paper. The form can be found at: www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=view.download&catid=74

Please remember that at least one author of the paper must be registered in both WPEIF and SBRC main symposium by April 2 for the paper to appear in the proceedings of WPEIF.

Last but not the least, please make sure the authors information in the paper and in JEMS are the same.

Regards,
Conference Chairs

Anexo B - E-mail de aceitação do artigo aceito no XII SBSeg.

From: SBSeg 2012 Artigos Curtos <sbseg2012.tpc@gmail.com>
To: [redacted]
Cc: Antônio Abelém <[redacted]@ufpa.br>; Elisangela Aguiar <[redacted]@serpro.gov.br>; Fernando Farias <[redacted]@ufpa.br>; Eduardo Cerqueira <[redacted]@ufpa.br>
Sent: Monday, September 3, 2012 11:05 PM
Subject: Your SBSeg 2012 Artigos Curtos paper 104781

Dear Mr. Fábio Nagahama:

Congratulations! Your paper "IPSFlow Uma Proposta de IPS Distribuído para Captura e Bloqueio Seletivo de Tráfego Malicioso em Redes Definidas por Software.", submitted to SBSeg 2012 Artigos Curtos, has been accepted for presentation in the XII Brazilian Symposium on Information and Computer System Security (SBSeg 2012), to be held in Curitiba, November 19-22, 2012.

The reviews are below or can be found at https://submissoes.sbc.org.br/PaperShow.cgi?m=104781.

Please, read these carefully, following the reviewers" recommendations as closely as possible.

The final camera-ready manuscript is due on Sept. 16, 2012 (FIRM), and one of the authors must be registered for SBSeg 2012 in order to upload the paper. ***NOTE: short papers must have up to 7 pages. ***

Please go to http://sbseg2012.ppgia.pucpr.br/?pg=chamadas for author instructions.

Final camera-ready manuscripts and copyright are uploaded via JEMS. You must use PDF format.

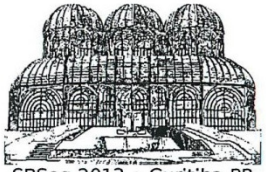
Please, observe that your camera-ready must follow the SBC template for papers: http://sbseg2012.ppgia.pucpr.br/@docs/artigos.rar

Also, you must fill and sign the copyright form: http://sbseg2012.ppgia.pucpr.br/@docs/copyrights_sbc.doc

Thank you again for contributing to SBSeg 2012. We look forward to seeing you in Curitiba.

Regards,
TPC Chairs

PUCPR UFPR UNIVERSIDADE FEDERAL DO PARANÁ UTFPR UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ



Certificado

Certificamos que **Fabio Yu Nagahama** participou do SBSeg 2012 – **XII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**, promovido pela Sociedade Brasileira de Computação e realizado de 19 a 22 de novembro de 2012 em Curitiba PR.

Participação nas seguintes atividades no âmbito do simpósio:

- Minicurso 1: *Análise de vulnerabilidades em Sistemas Computacionais Modernos: Conceitos, Exploits e Proteções.*
- Minicurso 2: *Introdução à Segurança de Dispositivos Móveis Modernos - Um Estudo de Caso em Android.*
- Minicurso 3: *Segurança em Redes Centradas em Conteúdo: Vulnerabilidades, Ataques e Contramedidas.*
- Minicurso 4: *Encriptação Homomórfica.*
- Apresentação do artigo "*IPSFlow – uma proposta de IPS distribuído para captura e bloqueio seletivo* de tráfego malicioso em redes definidas por software." nas sessões técnicas do SBSeg 2012. - Artigos Curtos

Curitiba PR, 19 de novembro de 2012

Coordenação Geral do SBSeg 2012
Aldri dos Santos, UFPR
Altair Santin, PUCPR
Carlos Maziero, UTFPR

SBC