Uma Arquitetura Elástica para Prevenção de Intrusão em Redes Virtuais usando Redes Definidas por Software*

Antonio Gonzalez Pastana Lobato,
Ulisses da Rocha Figueiredo,
Martin Andreoni Lopez,
Otto Carlos Muniz Bandeira Duarte

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{antonio,ulisses,martin,otto}@gta.ufrj.br

Resumo. *A arquitetura pluralista é uma das propostas mais promissoras para a Internet do Futuro. A técnica de virtualização de redes permite a coexistência de diversas redes em um mesmo substrato físico, atendendo a diferentes requisitos. Contudo, a virtualização de redes ainda apresenta grandes desafios de segurança e gerenciamento de recursos. Este artigo propõe uma arquitetura elástica para prevenção de intrusão em redes virtuais. A proposta combina um Sistema de Detecção de Intrusão, que identifica e alerta eventos anômalos, com as Redes Definidas por Software, que oferecem uma maior programabilidade para bloquear ataques. A elasticidade é obtida ao se criar ou destruir dinamicamente instâncias de detecção de intrusão. As propriedades das Redes Definidas por Software são também usadas para distribuir ou agrupar o tráfego entre as instâncias de detecção de intrusão.*

Abstract. *The pluralistic architecture is one of the most promising proposals for the Future Internet. The technique of network virtualization allows multiple networks to coexist on the same physical substrate, catering to different requirements. However, network virtualization still presents major challenges in security and resources management. This paper proposes an elastic architecture for intrusion prevention in virtual networks. The proposal combines an Intrusion Detection System, which identifies and alerts anomalous events, with Software Defined Networks, which offer greater programmability to block attacks. Elasticity is achieved by dynamically creating or destroying instances of intrusion detection. The Software Defined Networks are also used to distribute or cluster traffic between the instances of intrusion detection.*

1. Introdução

Os ambientes virtualizados permitem compartilhar os recursos das máquinas físicas em diversas máquinas virtuais. Porém, esses ambientes apresentam diversos desafios como garantir o isolamento entre as máquinas virtuais e prover a segurança no cenário de redes virtuais [Pearce et al. 2013, Ray and Schultz 2009]. O isolamento nas redes virtuais significa que uma rede não afeta as outras que executam na mesma infraestrutura

*Este trabalho foi realizado com recursos da CNPq, CAPES, FINEP, FUNTTEL e FAPERJ.

[Wu et al. 2010, Mattos and Duarte 2012]. Além desses novos desafios, problemas tradicionais relacionados com a segurança da Internet continuam existindo em ambientes virtualizados.

Mais de 60% dos ataques à segurança da rede é de origem interna [Lynch 2006]. Uma das formas de se proteger desses ataques internos é monitorar o tráfego em busca de atividades maliciosas ou violação de políticas. Para realizar o monitoramento de pacotes da rede, a ferramenta mais utilizada é um Sistema de Detecção de Intrusão (*Intrusion Detection System*-IDS) que realiza o monitoramento passivo dos pacotes na rede. Porém, esse tipo de análise não permite que sejam tomadas ações para prevenir que atividades maliciosas ocorram, portanto, se torna necessário um mecanismo que seja não só capaz de monitorar o fluxo de pacotes, mas também possa agir ativamente na rede de forma a bloquear o ataque, logo, um Sistema de Prevenção de Intrusão (*Intrusion Prevention System*-IPS) [Ruohomaa and Kutvonen 2005]. Os Sistemas de Detecção e Prevenção de Intrusão (IDPS) utilizam a técnica de Inspeção Profunda de Pacotes (*Deep Packet Inspection*-DPI) para realizar a análise de pacotes, consumindo recursos nesse processo.

Atualmente, para que um Sistema de Prevenção de Intrusão possa atuar sobre o tráfego da rede, é necessário que ele seja colocado diretamente na rota do pacote. Esta abordagem acaba introduzindo problemas, como, por exemplo: latência, já que cada pacote deverá ser processado individualmente antes de ser encaminhado para seu destino; acurácia, pois o IPS só consegue ver os pacotes do enlace onde ele está localizado, então, por desconhecer o comportamento dos demais tráfegos da rede, ele pode classificar uma atividade como sendo maliciosa de forma equivocada; flexibilidade, visto que não são instaladas regras nos comutadores de forma dinâmica em relação ao bloqueio do tráfego, o que pode afetar fluxos legítimos da rede.

Este artigo apresenta uma arquitetura elástica de detecção e prevenção de intrusão que realiza a detecção fora da rota do pacote. Para isso, as características de um ambiente virtualizado são utilizadas, o que permite uma rede lógica independente da rede física. A abordagem proposta no artigo é usar Redes Definidas por *Software* (*Software Defined Networks*-SDN), cujo protocolo mais utilizado é o OpenFlow [McKeown et al. 2008]. No protocolo OpenFlow, o controlador é centralizado e pode se comunicar com os comutadores da rede, portanto é capaz de analisar todo tráfego que passa por estes nós virtuais, podendo tomar ações sobre todos os fluxos de pacotes de forma dinâmica, conforme os alarmes gerados pelo Sistema de Detecção de Intrusão.

A arquitetura proposta utiliza o protocolo OpenFlow para monitorar o tráfego da rede, espelhando os pacotes para uma máquina virtual que inspeciona estes pacotes individualmente. A ferramenta de análise de tráfego utilizada é o Bro [Paxson 1999], que gera um alarme caso julgue o fluxo como malicioso. Para que este fluxo afete o mínimo do desempenho da rede, o controlador OpenFlow toma medidas para bloquear este fluxo o mais perto possível de sua origem. Desta forma, os pacotes são encaminhados tanto para seu destino quanto para a máquina que inspeciona os pacotes, que não está localizada na rota do fluxo.

Se existir um cenário de sobrecarga de uma máquina de detecção, parte do tráfego pode deixar de ser analisado o que acarretaria em um risco para a rede. Portanto, o artigo propõe o uso de uma ferramenta de monitoramento das máquinas que realizam DPI, para

fornecer recursos de acordo com a demanda de forma elástica. Se houver sobrecarga, uma mensagem é enviada ao sistema de gerência da rede, que cria uma réplica da máquina de DPI. O controlador OpenFlow é então informado para que seja instaurada uma nova regra para a distribuição dos fluxos espelhados. Caso após algum tempo o tráfego diminua e não sejam mais necessária tantas máquinas de DPI, um outro alerta é enviado para que estas máquinas sejam desativadas, garantindo assim um melhor aproveitamento de recursos.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados na literatura; a Seção 3 apresenta os modelos de virtualização usados na proposta; na Seção 4 são apresentados os modos de operação para Sistemas de Detecção de Intrusão na rede; a Seção 5 apresenta a arquitetura proposta e a Seção 6 trata dos resultados do protótipo implementado; por último, a Seção 7 conclui o artigo e apresenta os trabalhos futuros.

2. Trabalhos Relacionados

Outras propostas que visam o uso de Redes Definidas por *Software* para a implementação de aplicações de segurança em ambientes virtualizados são apresentadas nesta seção. Estas propostas se servem da flexibilidade provida pelo paradigma da separação de planos para realizar a interação entre os alarmes e os comutadores. Shim *et al.* propõem uma nova linguagem de alto nível baseada em OpenFlow específica para criação de aplicações para segurança de forma modular [Shin et al. 2013]. Kim *et al* apresentam uma abordagem baseada nos estados da rede, que usa a alta programabilidade oferecida por SDN para decidir que aplicações executar dependendo do evento [Kim et al. 2013], nele a linguagem Pyretic, que modulariza funções de rede, é utilizada.

OpenSAFE [Ballard et al. 2010] propõe um sistema para redirecionamento de fluxos para aplicações de monitoramento e segurança. OpenSAFE também especifica uma linguagem para um gerenciamento simplificado de fluxos em aplicações de segurança. O OpenSAFE se baseia no protocolo OpenFlow para implementar funções básicas para manipulação de fluxos, como por exemplo filtros, que selecionam fluxos específicos e também desvios, para encaminhar o tráfego para aplicações de monitoramento e segurança. Há ainda um componente OpenFlow que interpreta as políticas definidas pela linguagem de gerenciamento de fluxo para que as ações sejam realizadas.

Especificamente para um Sistema de Prevenção de Intrusão (*Intrusion Prevention System*–IPS), Nagahama *et al.* realizam um estudo de caso para implementação de um IPS baseado em OpenFlow com uma captura seletiva de fluxos para o módulo de detecção de intrusão [Nagahama et al. 2012]. Porém, não foram consideradas possíveis sobrecargas neste módulo, o que poderia acarretar em não analisar parte do tráfego. Além disso, como a captura é seletiva, nem todos os fluxos passam por este módulo de inspeção, que caso mal configurado pode gerar uma falha de segurança.

O SnortFlow [Xing et al. 2013] é outra ferramenta de detecção de intrusão, onde é utilizado o analisador de tráfego Snort [Roesch et al. 1999], localizado no Domínio 0 do hipervisor Xen [Egi et al. 2008], para realizar a inspeção de pacotes da rede. Meha compara o desempenho das ferramentas de detecção de intrusão Bro e Snort, sendo que a primeira apresenta um melhor desempenho na inspeção de pacotes [Mehra 2012]. Ao se instalar o Snort no Domínio 0, conectado a um comutador ligado às máquinas virtuais de

uma máquina física, o Snort só consegue analisar o tráfego dessas máquinas virtuais. Para uma visão global da rede seria necessário alguma forma de comunicação e sincronismo, que não é apresentada no artigo do SnortFlow. Na arquitetura proposta, a máquina com o Bro consegue ver todos os pacotes da rede, visto que o controlador desvia todo o tráfego da rede para a máquina de análise de pacotes. Caso haja réplicas da máquina de DPI, o controlador distribui os fluxos considerando sua origem para facilitar a detecção de um possível atacante.

3. Modelos de Virtualização

A proposta consiste em um ambiente híbrido de virtualização. Para criar esse ambiente virtual foram utilizadas a virtualização de máquinas provida pelo Xen e a comutação de fluxos programável provida pelo OpenFlow. A vantagem da utilização do Xen é a criação de máquinas virtuais isoladas entre si e com um controle distribuído. Essa vantagem possibilita que sejam desenvolvidas novas aplicações para o gerenciamento da infraestrutura virtual. A vantagem da utilização do protocolo OpenFlow em um comutador programável é que a migração de fluxos é feita de forma simplificada e, consequentemente, também é possível a migração de enlaces virtuais, o que permite o remapeamento de topologias lógicas sobre a topologia física. Na proposta, também são coletadas estatísticas de consumo de recursos das máquinas geradas pelo Xen para o monitoramento do ambiente virtual.

3.1. A Plataforma de Virtualização de Máquinas Xen

O Xen [Egi et al. 2008] é uma plataforma de virtualização de computadores pessoais. Essa plataforma de virtualização é usada em servidores de centro de dados (*data-centers*) em empresas como a *Amazon*, onde a virtualização do Xen teve seu desempenho avaliado [Wang and Ng 2010]. A arquitetura do Xen se baseia em uma camada hipervisora de virtualização logo acima do *hardware*, que monitora o comportamento das máquinas virtuais, como mostrada na Figura 1. As máquinas virtuais são executadas sobre o hipervisor, que garante um acesso independente a recursos como CPU, memória, rede e disco.

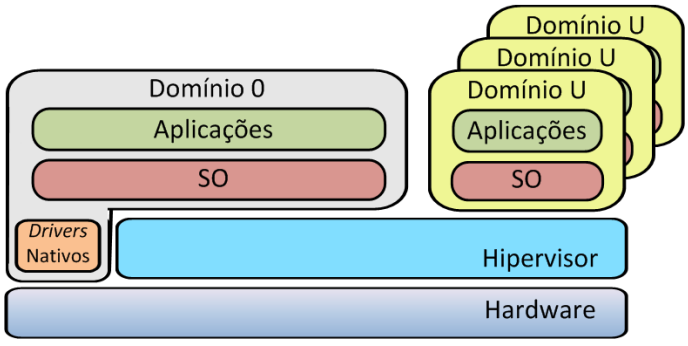


Figura 1. Arquitetura Xen com a camada hipervisora e a separação entre o domínio privilegiado e os demais domínios.

No Xen, cada ambiente virtual está isolado dos demais, ou seja, o funcionamento de uma máquina virtual não afeta o de outras. Além disso, permite que essas máquinas

virtuais possuam sistemas operacionais distintos. O Xen possui um ambiente virtual privilegiado, o Domínio 0, que executa as operações de gerência do hipervisor e detém a exclusividade no acesso aos dispositivos físicos, como os dispositivos de Entrada/Saída (E/S). Desta forma, as máquinas virtuais só conseguem acesso aos dispositivos físicos através de *drivers* virtuais que se comunicam com o Domínio 0.

3.2. Monitoramento de Recursos

Em ambientes virtuais é possível gerenciar a alocação de recursos de acordo com a demanda ou de acordo com políticas pré-estabelecidas. Essa gerência é obtida através do remapeamento das instâncias virtuais nas diversas máquinas físicas, de acordo com a quantidade de recursos disponíveis e as políticas estabelecidas [Carvalho and Duarte 2012]. Portanto, monitorar o consumo de recursos de máquinas virtuais e físicas permite a criação de métricas para alocação das máquinas virtuais de acordo com um objetivo estabelecido. Um exemplo disso, é a computação em nuvem, na qual os provedores de serviço gerenciam de forma dinâmica os recursos fornecidos aos clientes de forma a entregar os recursos contratados com o menor custo possível.

Na proposta, o monitoramento foi feito a partir da coleta de dados do Xen através da Interface de Programação de Aplicação *Libvirt* [Bolte et al. 2010]. A *Libvirt* tem acesso às informações do hipervisor através de uma conexão segura e permite que essas informações sobre o consumo das máquinas físicas e virtuais possam ser usadas em diversas linguagens de programação.

3.3. O Comutador de Fluxos OpenFlow

A abordagem das Redes Definidas por Software (*Software Defined Networking* – SDN) permite o gerenciamento do tráfego de redes. Elas permitem a definição de fluxos de pacotes de forma dinâmica. O paradigma de SDN é o da separação de planos, em que a rede é dividida em plano de dados e plano de controle. O plano de controle é responsável pela definição da próxima rota de um fluxo e, para isso, executa algoritmos de controle da rede. Enquanto o plano de dados faz o encaminhamento dos pacotes de acordo com políticas definidas pelo controlador. Essa separação fornece a facilidade de que vários comutadores podem compartilhar o mesmo plano de controle, que detém a visão global da rede. Desta forma, é possível redefinir as rotas de fluxos de múltiplos comutadores apenas mudando a aplicação no controlador.

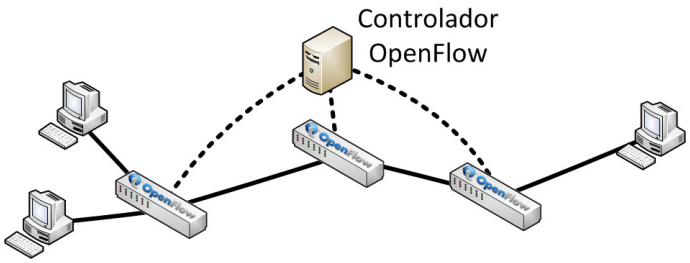


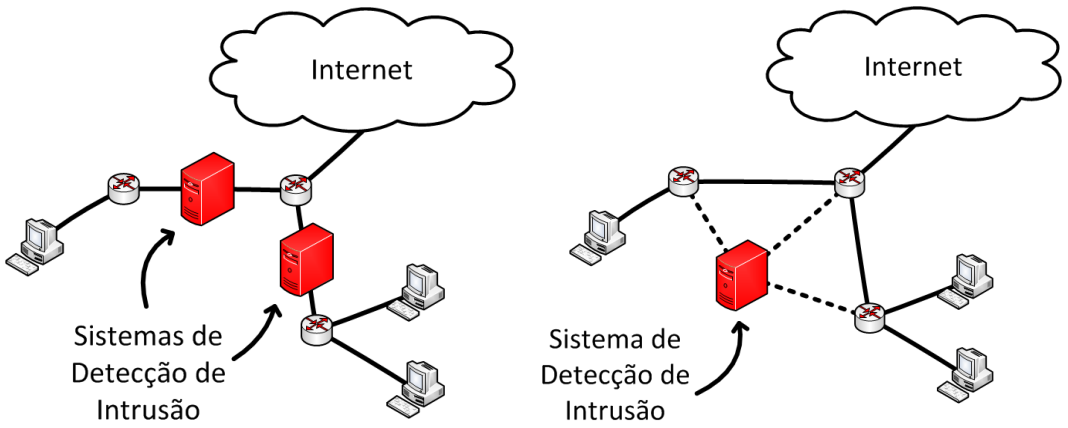
Figura 2. Arquitetura OpenFlow na qual diversos comutadores se comunicam com o controlador para definição de fluxos.

O protocolo OpenFlow [McKeown et al. 2008] define regras e padrões para SDN. Na arquitetura OpenFlow, existe uma tabela de fluxos nos comutadores que é atualizada

pelo plano de controle. Quando chega um pacote de um fluxo que já está definido nessa tabela, ele é simplesmente comutado. Caso chegue um pacote de um fluxo não definido, ele é enviado para o controlador, onde é definido qual será o próximo destino desse pacote. O destino é escolhido de acordo com as aplicações sendo executadas e também de acordo com o cabeçalho do pacote. O controlador então adiciona essa regra na tabela de fluxo do elemento encaminhador. É importante ressaltar que só o primeiro pacote de um fluxo não definido é encaminhado ao controlador, sendo a partir dele criada a regra na tabela de fluxo, o que faz que os demais pacotes desse fluxo sejam somente comutados, afetando pouco o desempenho da rede. A arquitetura OpenFlow está ilustrada na Figura 2.

4. A Análise de Tráfego e a Detecção de Intrusão na Rede

A análise de tráfego e a detecção de intrusão na rede são essenciais para a segurança. Deve ser ressaltado que a detecção de intrusão é mandatória para a detecção de ameaças provocadas por usuários já autenticados ou que conseguiram passar pelo *firewall*. O sistema de detecção de intrusão analisa pacotes para monitorar eventos que ocorrem na rede em busca de um comportamento suspeito que indique sinais de intrusão, que podem ser detectados por assinatura, quando o ataque é conhecido, ou por anomalia, quando a rede se comporta de uma forma diferente da habitual.



(a) Captura e análise na rota do pacote. (b) Captura e análise do pacote por espelhamento.

Figura 3. A captura e análise dos pacotes para detecção de intrusão. A interceptação na rota só analisa os pacotes no enlace no qual a captura é realizada. Com a duplicação dos pacotes e o envio de mensagens, um único sistema de detecção de intrusão é capaz de analisar os pacotes de toda a rede.

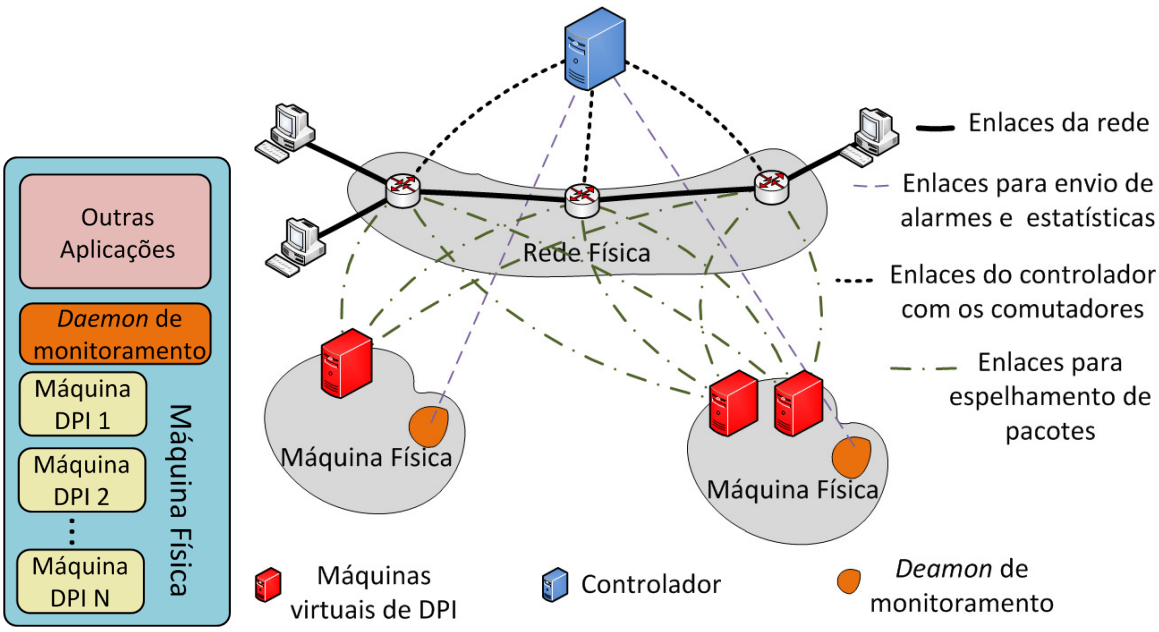
Um Sistema de Detecção de Intrusão (*Intrusion Detection System*-IDS) em rede pode atuar de dois modos distintos: na rota dos pacotes, no qual o IDS captura e analisa os pacotes e depois os encaminha para o próximo salto de sua rota; e fora da rota, na qual os pacotes são espelhados em um comutador, sendo encaminhados tanto para um Sistema de Detecção de Intrusão, quanto para o seu próximo destino na rota. Esses modos de operação estão ilustrados na Figura 3. Na arquitetura proposta, o modo utilizado foi o fora da rota do pacote, pois desta forma evita-se os problemas de latência devido ao processamento antes de encaminhar o pacote para seu destino, como acontece no modo na rota do pacote. No modo fora da rota, uma maior acurácia também é obtida, pois um

único IDS fora da rota pode receber pacotes de diversos enlaces da rede. Enquanto na rota, o IDS estaria restrito aos pacotes no enlace em que ele está localizado.

5. Arquitetura Proposta

O artigo propõe uma arquitetura elástica para detecção de intrusão em redes virtuais, com alocação dinâmica de máquinas virtuais de análise de pacotes de acordo com a demanda. Além da elasticidade de criação e destruição de máquinas para IDSs, a arquitetura provê um mecanismo de redefinição dos fluxos para balancear o tráfego direcionado às máquinas que realizam a detecção de intrusão e a tomada de ações em caso de alguma atividade maliciosa. A arquitetura de detecção de intrusão proposta consiste em três módulos principais, interligados, para controlar a rede: o módulo de detecção de intrusão, o módulo de gerenciamento de fluxo e o módulo de gerenciamento de recursos.

Nesta arquitetura, as máquinas físicas podem alocar máquinas virtuais de DPI e executar aplicações em paralelo, como mostrada na Figura 4(a), além disso, apresentam um *daemon* para monitoramento dos recursos consumidos tanto pela máquina física quanto pelas máquinas de DPI.



(a) Máquina física com máquinas virtuais de DPI.

(b) Exemplo da arquitetura proposta e a comunicação entre seus elementos.

Figura 4. Na arquitetura proposta, uma máquina física pode conter diversas máquinas de DPI, Figura 4(a). Em 4(b), pode ser visto o controlador ligado aos comutadores para a redefinição de fluxos levando em conta as estatísticas e alarmes das máquinas de DPI.

Na Figura 4(b), está ilustrado um exemplo de uma rede que utiliza a arquitetura proposta para a detecção de intrusão. O tráfego da rede é espelhado nos comutadores para as máquinas de detecção de intrusão. Existe uma comunicação do controlador com as máquinas físicas para o envio de estatísticas sobre o consumo de recursos e alarmes em caso de atividades maliciosas.

5.1. Módulo de Detecção de Intrusão

Este módulo é composto por máquinas virtuais que executam uma ferramenta de detecção de intrusão por Inspeção Profunda de Pacotes (DPI). Essas máquinas analisam todo o tráfego da rede e geram alarmes para o controlador OpenFlow. Na proposta, a ferramenta utilizada é o Bro. Como o Bro não pode executar em *multi-threading*, ele só usa um único núcleo da CPU, portanto, cada uma dessas máquinas virtuais foi configurada de forma a ter acesso a somente um núcleo, para evitar desperdício de recursos.

A comunicação deste módulo com o Módulo de Gerenciamento de Fluxo em caso de alarme é feita através de um canal seguro. Quando uma das máquinas de DPI detecta um ataque, um alarme é disparado contendo as informações sobre o fluxo malicioso, como sua origem, que são utilizadas para a posterior tomada de ações.

O Módulo de Detecção de Intrusão se adapta à demanda. O número de máquinas virtuais com Bro pode variar de acordo com a quantidade de fluxos da rede, sendo instanciadas dinamicamente. Desta forma, a arquitetura apresenta um comportamento elástico.

5.2. Módulo de Gerenciamento de Fluxo

Este módulo é responsável pela distribuição dos fluxos entre as máquinas com o Bro e também por tomar ações contra possíveis ataques alertados. O módulo é composto por uma aplicação do controlador OpenFlow POX, que se comunica tanto com o Módulo de Detecção de Intrusão, quanto com o Módulo de Gerenciamento de Recursos. O espelhamento de pacotes da rede para as máquinas de DPI é feita através de um túnel GRE *Generic Routing Encapsulation* e a análise dos pacotes é feita após o desencapsulamento, para garantir que estes pacotes não tenham sido alterados.

Caso o Módulo de Detecção de Intrusão possua mais de um máquina DPI, a distribuição dos fluxos é feita levando em conta dois aspectos: a quantidade de recursos para análise de pacotes disponíveis em cada máquina virtual de DPI; e a origem dos pacotes. Os pacotes de mesma origem têm prioridade para serem alocados na mesma máquina de DPI, para evitar que um ataque desta origem passe despercebido. Ao distribuir os fluxos entre diversos IDSs, um ataque Negação de Serviço Distribuída (*Distributed Denial of Service* - DDoS) pode ter sua detecção dificultada. Entretanto, como o controlador OpenFlow determina todos os fluxos da rede, ele pode auxiliar na detecção desse ataque, como mostrado em [Braga et al. 2010].

A tomada de ações de bloqueio de fluxo é feita analisando a origem do fluxo malicioso. A origem é informada no alarme gerado pelo Módulo de Detecção de Intrusão. Assim, todos os fluxos provenientes desta origem já instalados nos comutadores OpenFlow são então listados e bloqueados. Além disso, uma outra regra é adicionada aos comutadores para que os futuros fluxos desta origem sejam bloqueados automaticamente. A regra para o bloqueio é instalada com um tempo de duração previamente determinado, deixando o atacante em quarentena. É importante listar os fluxos já instalados, pois no protocolo OpenFlow regras com correspondência mais completa tem prioridade sobre as demais. Caso contrário, a regra de apenas bloqueio da origem não surtiria efeito sobre os fluxos já existentes.

5.3. Módulo de Gerenciamento de Recursos

O modulo de gerenciamento dos recursos se encontra no domínio privilegiado, Domínio 0, de todas as máquinas físicas da rede. Nesse módulo são monitorados os consumos de banda, de processamento e de memória da máquina física e também o quanto desses recursos é consumido por cada máquina virtual. Este monitoramento é feito através da coleta de dados do Xen realizadas pela *Libvirt*. Cada máquina física executa um *daemon* para coleta de dados e das estatísticas de todas as máquinas físicas da rede que são agregadas no controlador. Desta forma, o controlador tem a informação da quantidade de máquinas de DPI da rede e o consumo de recursos de cada uma.

Para evitar sobrecarga ou a presença de recursos ociosos, o sistema analisa principalmente as máquinas do Módulo de Detecção de Intrusão. Em caso de sobrecarga, este módulo analisa os recursos disponíveis nas máquinas físicas e decide então onde instanciar uma nova máquina de DPI. De maneira análoga, também é feito o monitoramento conjunto de todas as máquinas de DPI, para detectar quando for possível uma redistribuição de fluxos que permita a desativação de uma dessas máquinas. Desta forma, garantindo a elasticidade da proposta.

6. Resultados

Para validar a proposta da arquitetura elástica de detecção de intrusão, foi desenvolvido um protótipo na plataforma FITS[1] (*Future Internet Testbed with Security*) [Mattos et al. 2012] que é uma rede de testes interuniversitária para experimentação de propostas para a Internet do Futuro. O FITS é baseado nos mecanismos Xen e OpenFlow para prover uma arquitetura pluralista, permitindo a coexistência de múltiplas redes em paralelo executando aplicações diferentes. Na plataforma FITS o plano de controle executa nas máquinas virtuais Xen e o encaminhamento de pacotes é desempenhado pelo OpenFlow. A arquitetura está ilustrada na Figura 5, na qual os nós das redes são compostos por máquinas virtuais Xen isoladas agindo como comutadores OpenFlow.

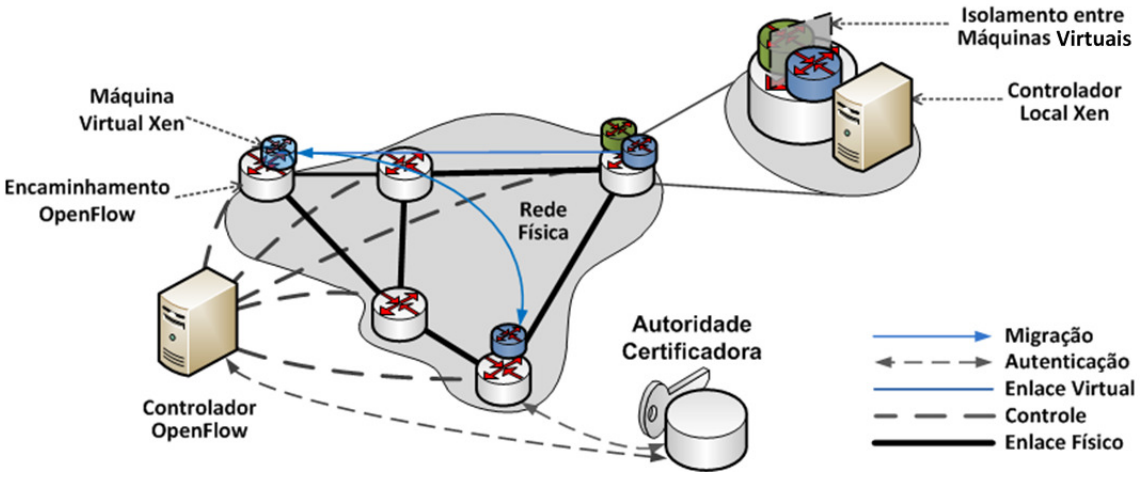


Figura 5. A arquitetura da plataforma FITS: roteadores virtuais com isolamento executam nos roteadores físicos usando a plataforma de virtualização Xen e encaminhamento de dados usando OpenFlow.

[1] www.gta.ufrj.br/fits

Foram realizados os seguintes experimentos para avaliar o correto funcionamento e o bom desempenho da arquitetura elástica de detecção de intrusão proposta: bloqueio de um fluxo malicioso; avaliação dos recursos consumidos para análise de pacotes; e elasticidade de sobrecarga e descarga no Módulo de Detecção de Intrusão.

Os experimentos foram realizados em um servidor com processador Intel Xeon X5690 com 24 núcleos, com frequência de 3.47GHz de *clock* e com 48 GB de memória RAM. Para os testes foram geradas taxas de pacotes do tipo SYN entre máquinas clientes da rede para simular um ataque de negação de serviço por inundação. Ao passarem pelo comutador esses fluxos são espelhados para o Módulo de Detecção de Intrusão para análise.

6.1. Experimento de Bloqueio de Fluxo Malicioso

Este experimento analisa o efeito do bloqueio de um fluxo considerado malicioso sobre os demais fluxos da rede. O resultado das medidas está ilustrado na Figura 6 e mostra que quando o bloqueio do ataque acontece, o fluxo considerado legítimo não é afetado. Quando uma intrusão é detectada o Módulo de Detecção de Intrusão manda um alerta para o controlador que atualiza a tabela de fluxos bloqueando o fluxo malicioso em todos os comutadores OpenFlow da rede. Esta visão global da rede, característica das redes definidas por software, resulta em que o ataque seja bloqueado o mais perto possível de sua origem.

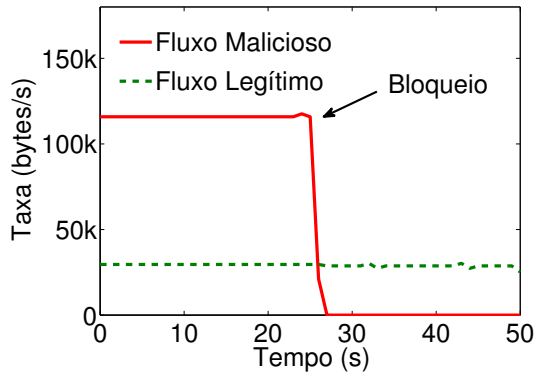
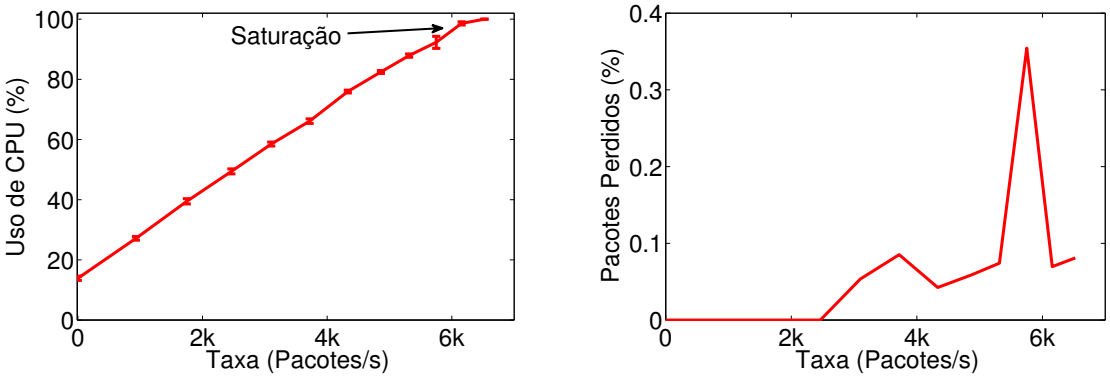


Figura 6. Ação eficaz de bloqueio de um fluxo malicioso de negação de serviço por inundação sem afetar o tráfego legítimo.

6.2. Avaliação dos Recursos Consumidos pelo Bro em uma Máquina Virtual

O Sistema de Detecção de Intrusão utilizado captura e analisa em tempo real os pacotes espelhados e, portanto, há uma sobrecarga associada a esta função. Assim, avaliou-se o consumo de recursos da análise de pacotes pelo Sistema de Detecção de Intrusão para se determinar qual dos aspectos, banda ou processamento, é o mais crítico para a detecção deste ataque de negação de serviço por inundação. É importante ressaltar que o processamento exigido pela análise dos pacotes depende da política de segurança e de que tipos de ameaças são analisadas. A técnica de Inspeção Profunda de Pacotes (*Deep Packet Inspection*–DPI) requer uma quantidade considerável de processamento. Neste experimento foram geradas taxas crescentes de pacotes e foram analisados tanto o processamento gasto pela máquina de DPI, quanto o percentual de pacotes perdidos pela máquina.

Conforme ilustrado na Figura 7, a máquina de DPI saturou o processamento antes de ter uma perda significativa de pacotes. No ponto crítico de processamento a perda de pacotes foi menor que 0.5%. Portanto, para os demais experimentos, é possível definir o critério para saturação de uma máquina do Módulo de Detecção de Intrusão como consumo de CPU.



(a) Consumo de CPU da máquina de DPI para análise de pacotes.

(b) Percentual de pacotes perdidos pela máquina de DPI.

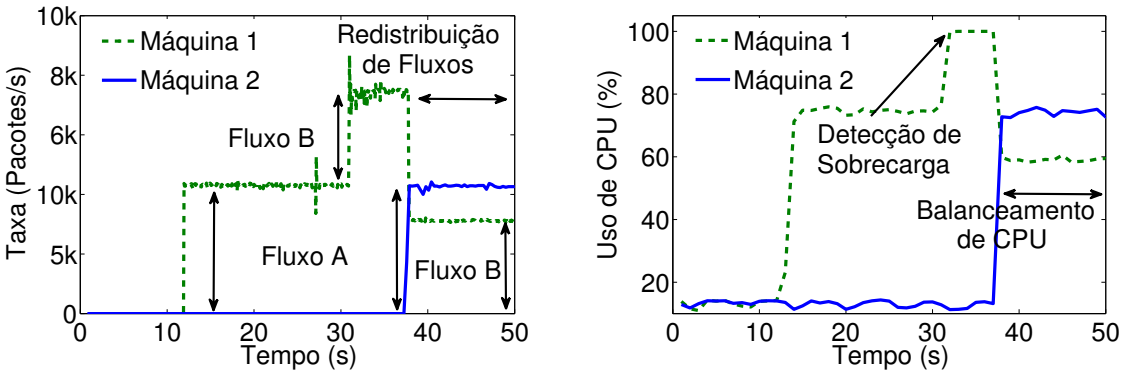
Figura 7. Processamento e percentual de perdas de pacotes em uma máquina de DPI. Comparando as Figuras 7(a) e 7(b), percebe-se que não há perda significativa de pacotes para a taxa em que o processamento é saturado.

6.3. Sobrecarga na Detecção de Intrusão

Este experimento visa avaliar a performance da arquitetura em caso de sobrecarga no Módulo de Detecção de Intrusão, que ocorre caso uma taxa elevada de pacotes precise ser analisada. Para este experimento, foram geradas taxas de pacotes constantes que eram inspecionados pela única máquina de DPI ativa no momento devido à baixa demanda. Em seguida, um novo fluxo foi criado, de forma a sobrecarregar esta máquina. Com isso, a máquina física onde a máquina de DPI estava localizada enviou uma mensagem de sobrecarga para o controlador. Neste experimento, a sobrecarga foi avaliada pelo processamento da máquina com o Bro, pois, como visto na Seção 6.2, o processamento satura antes da banda.

Quando o controlador recebe a mensagem de sobrecarga, ele executa o algoritmo de balanceamento para decidir em qual máquina física deve ser instanciada a nova máquina de detecção de intrusão. Após a ativação da nova máquina, os fluxos são redistribuídos levando em conta sua origem e os recursos de cada máquina de análise de pacotes.

Os resultados ilustrados na Figura 8 mostram que a partir do segundo fluxo ocorre uma sobrecarga na máquina de DPI. Quando a sobrecarga é detectada, existe um intervalo de tempo, devido à instanciação da nova máquina, até que os fluxos sejam redistribuídos. Após o balanceamento dos fluxos, todos os pacotes passam a ser analisados sem saturar o Módulo de Detecção de Intrusão.

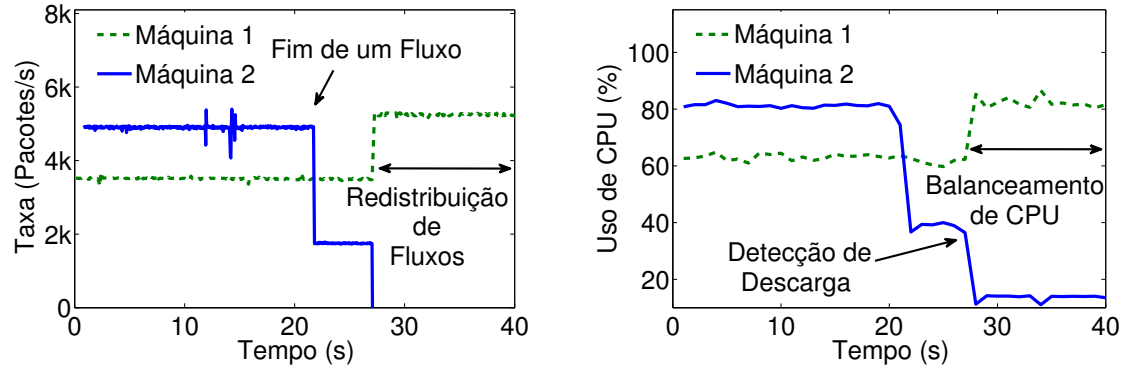


(a) Taxa de Pacotes recebida pelas máquinas de DPI. (b) Consumo de CPU de cada máquina de DPI.

Figura 8. Análise temporal do Módulo de Detecção de Intrusão em caso de sobrecarga. São iniciados dois fluxos na máquina 1, Figura 8(a), causando sobrecarga de CPU, Figura 8(b). Para evitar a saturação da máquina 1, a máquina 2 é ativada e os fluxos são redistribuídos, balanceando o consumo de CPU.

6.4. Descarga na Detecção de Intrusão

Este experimento foi feito para medir a elasticidade da arquitetura. Caso haja recursos ociosos, a proposta visa redistribuir os fluxos de forma que uma máquina possa ser desativada. Os recursos consumidos são constantemente analisados e informados ao controlador para que ele possa detectar quando um remanejamento de fluxos irá permitir a desativação de uma das máquinas do Módulo de Detecção de Intrusão.



(a) Taxa de Pacotes recebida pelas máquinas de DPI. (b) Consumo de CPU de cada máquina de DPI.

Figura 9. Análise temporal do Módulo de Detecção de Intrusão em caso de descarga. Duas máquinas de DPI estão operando quando um dos fluxos termina, Figura 9(a). Então, o controlador analisa o consumo de CPU das duas máquinas, Figura 9(b), e redistribui os fluxos de forma a poder desativar uma das máquinas.

O teste começa com duas máquinas virtuais de DPI recebendo taxas constantes de pacotes, como mostrada na Figura 9. Após um período de tempo, um desses fluxos foi cancelado, causando uma queda perceptível no processamento de uma das máquinas. Ao detectar esta descarga, o controlador redistribui os fluxos de forma que a máquina com menor processamento passe a receber nenhum fluxo e, portanto, possa ser desativada.

7. Conclusão e Trabalhos Futuros

O artigo propõe uma arquitetura elástica para Prevenção de Intrusão. Os resultados mostram que o processamento de CPU durante a análise de pacotes na detecção de intrusão é um ponto crítico conforme o aumento do tráfego. Desta forma, a arquitetura proposta analisa esse aspecto para que os recursos sejam fornecidos de maneira dinâmica, evitando que pacotes deixem de ser inspecionados. A proposta permite a instanciação e a desativação dinâmica de máquinas de DPI de forma que a inspeção de todos os pacotes da rede seja feita evitando recursos ociosos.

O comportamento da arquitetura proposta foi analisado para os casos de sobrecarga e descarga do Módulo de Detecção de Intrusão. No caso da sobrecarga, a distribuição dos fluxos entre duas máquinas de DPI, permitiu que todos os pacotes fossem analisados, já que ouve um balanceamento de CPU. No caso de descarga, o fluxo foi redistribuído de forma que uma máquina parasse de receber fluxos, o que permite a desativação dessa máquina de DPI. Portanto, a arquitetura fornece recursos de acordo com a demanda, ou seja, realiza a prevenção de intrusão de forma elástica.

A análise mais detalhada do comportamento da arquitetura proposta para o ataque de negação de serviço distribuída será realizada em um trabalho futuro. Além disso, pretende-se integrar a arquitetura ao FITS, para fornecer uma ferramenta de prevenção de intrusão elástica como serviço para novas propostas de experimentação para a Internet do Futuro.

8. Referências

[Ballard et al. 2010] Ballard, J. R., Rae, I., and Akella, A. (2010). Extensible and scalable network monitoring using opensafe. *Proc. INM/WREN*.

[Bolte et al. 2010] Bolte, M., Sievers, M., Birkenheuer, G., Niehörster, O., and Brinkmann, A. (2010). Non-intrusive virtualization management using libvirt. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 574–579. European Design and Automation Association.

[Braga et al. 2010] Braga, R., Mota, E., and Passito, A. (2010). Lightweight DDoS flooding attack detection using NOX/OpenFlow. In *IEEE 35th Conference on Local Computer Networks (LCN)*, pages 408–415.

[Carvalho and Duarte 2012] Carvalho, H. E. and Duarte, O. C. (2012). Voltaic: volume optimization layer to assign cloud resources. In *Proceedings of the 3rd International Conference on Information and Communication Systems*, page 3. ACM.

[Egi et al. 2008] Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Huici, F., and Mathy, L. (2008). Towards high performance virtual routers on commodity hardware. In *Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12. ACM.

[Kim et al. 2013] Kim, H., Gupta, A., Shahbaz, M., Reich, J., Feamster, N., Clark, R., and Tech, G. (2013). Simpler network configuration with state-based network policies. *Georgia Institute of Technology, Atlanta, USA*.

[Lynch 2006] Lynch, D. M. (2006). Securing against insider attacks. *Information Systems Security*, 15(5):39–47.

[Mattos and Duarte 2012] Mattos, D. M. F. and Duarte, O. C. M. B. (2012). QFlow: Um sistema com garantia de isolamento e oferta de qualidade de serviço para redes virtualizadas. In *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2012*.

[Mattos et al. 2012] Mattos, D. M. F., Mauricio, L. H., Cardoso, L. P., Alvarenga, I. D., Ferraz, L. H. G., and Duarte, O. C. M. B. (2012). Uma rede de testes interuniversitária a com técnicas de virtualizacao híbridas. In *Salão de Ferramentas do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'12*.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: in campus networksenabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*.

[Mehra 2012] Mehra, P. (2012). A brief study and comparison of snort and bro open source network intrusion detection systems.

[Nagahama et al. 2012] Nagahama, F. Y., Farias, F., Aguiar, E., Gaspary, L., Granville, L., Cerqueira, E., and Abelém, A. (2012). Ipsflow–uma proposta de sistema de prevenção de intrusão baseado no framework openflow.

[Paxson 1999] Paxson, V. (1999). Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463.

[Pearce et al. 2013] Pearce, M., Zeadally, S., and Hunt, R. (2013). Virtualization: Issues, security threats, and solutions. *ACM Computing Surveys (CSUR)*, 45(2):17.

[Ray and Schultz 2009] Ray, E. and Schultz, E. (2009). Virtualization security. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, page 42. ACM.

[Roesch et al. 1999] Roesch, M. et al. (1999). Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238.

[Ruohomaa and Kutvonen 2005] Ruohomaa, S. and Kutvonen, L. (2005). Trust management survey. In *Trust Management*, pages 77–92. Springer.

[Shin et al. 2013] Shin, S., Porras, P., Yegneswaran, V., Fong, M., Gu, G., and Tyson, M. (2013). Fresco: Modular composable security services for software-defined networks. In *Proceedings of Network and Distributed Security Symposium*.

[Wang and Ng 2010] Wang, G. and Ng, T. E. (2010). The impact of virtualization on network performance of amazon ec2 data center. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

[Wu et al. 2010] Wu, H., Ding, Y., Winer, C., and Yao, L. (2010). Network security for virtual machine in cloud computing. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pages 18–21. IEEE.

[Xing et al. 2013] Xing, T., Huang, D., Xu, L., Chung, C.-J., and Khatkar, P. (2013). Snortflow: A openflow-based intrusion prevention system in cloud environment. In *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*, pages 89–92. IEEE.