Uma Arquitetura de Virtualização de Funções de Rede para Proteção Automática e Eficiente contra Ataques

Leopoldo A. F. Mauricio[1,3], Igor D. Alvarenga[1], Marcelo G. Rubinstein[2] e Otto Carlos M. B. Duarte[1]

[1]Universidade Federal do Rio de Janeiro - GTA/PEE/COPPE
[2]Universidade do Estado do Rio de Janeiro - FEN/DETEL/PEL
[3]Globo.com

leopoldo@corp.globo.com,{alvarenga,otto}@gta.ufrj.br,rubi@uerj.br

Resumo. *Sistemas de prevenção de intrusão são caros e ineficientes ao bloquearem todo o tráfego de um endereço de origem. Logo, a capacidade de aplicar contramedidas somente ao tráfego malicioso se torna um importante desafio de segurança. Este artigo propõe uma arquitetura de virtualização de funções de rede (Network Functions Virtualization - NFV) que oferece proteção automática e eficiente contra ataques. Os fluxos maliciosos são dinamicamente desviados por uma cadeia de funções virtuais de rede (Virtual Network Functions - VNFs) contendo um firewall de aplicação e um IDS. Um protótipo foi construído utilizando-se a plataforma Open Platform for NFV (OPNFV). A arquitetura é capaz de bloquear 99,12% dos ataques sem afetar 93,6% do tráfego benigno.*

Abstract. *Intrusion Prevention Systems (IPSs) are expensive and inefficient when blocking all traffic from a source address. Therefore, the ability to apply countermeasures only to malicious traffic becomes an important security challenge. This article proposes a Network Function Virtualization (NFV) architecture to provide automatic, and efficient protection against attacks. Malicious flows are dynamically diverted through a Virtual Network Functions (VNFs) chain containing an application firewall and an IDS. A prototype was built using the Open Platform for NFV (OPNFV). The architecture is capable of blocking 99.12% of the attacks without affecting 93.6% of the benign traffic.*

1. Introdução

Os ataques de segurança, cada vez mais frequentes na Internet, causam grande prejuízo a pessoas e organizações [Akamai 2017, Anderson et al. 2013]. Ataques de negação de serviço (*Denial of Service* - DoS) visam consumir os recursos do alvo atacado para causar indisponibilidade, enquanto outros, por exemplo, exploram vulnerabilidades para desfigurar o conteúdo de sistemas, propagar código malicioso ou roubar dados sensíveis [Gu et al. 2008, Haggard and Lindsay 2015]. Por isso, sistemas de proteção de rede devem ter capacidade de detectar e aplicar contramedidas eficientes contra os ataques de segurança, a fim de proteger dispositivos, aplicações e redes ligadas à Internet. Além disso, os sistemas de proteção devem ser ágeis e eficientes, de forma a reduzirem o tempo entre a detecção e a reação necessária para mitigar ataques [Amann and Sommer 2015].

Muitas vezes os ataques são gerados a partir dos nós de uma *botnet* [Mtibaa et al. 2015]; um conjunto de máquinas zumbis com *malwares* que permitem a

um atacante controlar o dispositivo “infectado”. Nesse caso, os usuários reais das máquinas escravizadas não sabem que existe tráfego malicioso gerado a partir de seus dispositivos, que podem ser computadores, aparelhos de celular, aparelhos de TV, etc. Além disso, o endereço IP de origem de um tráfego identificado como malicioso pode pertencer a um *gateway* NAT (*Network Address Translation*) que pode servir grande quantidade de nós de rede. Em qualquer dos casos, o bloqueio do endereço IP de origem é uma contramedida eficaz contra ataques de negação de serviço. Entretanto, também é preciso impedir os ataques que exploram vulnerabilidades, tais como a injeção de SQL (*Structured Query Language Injection* - SQLI) e o *cross-site scripting* (XSS), sem afetar o tráfego benigno gerado pelos dispositivos. O XSS é um tipo de vulnerabilidade encontrada normalmente em aplicações web, que ativa ataques maliciosos ao injetar *client-side script* dentro das páginas web vistas por outros usuários. Assim, os sistemas de proteção devem ser eficientes para bloquear todas as atividades maliciosas sem impedir o acesso benigno às redes e aplicações.

A virtualização de funções de rede (*Network Functions Virtualization* - NFV), normatizada pelo *European Telecommunications Standards Institute* (ETSI), é uma nova tecnologia que tem como objetivo tornar as redes mais ágeis e flexíveis e reduzir os custos materiais e de operação. O NFV propõe a implantação de funções de rede virtuais (*Virtual Network Functions* - VNFs) em hardware genérico de prateleira, como alternativa ao uso de dispositivos de rede e segurança customizados [ETSI 2012]. Dessa forma, tanto o custo de capital (CAPEX) quanto o custo operacional (OPEX) são reduzidos. Isso ocorre porque dispositivos especializados não são utilizados e as despesas com dissipação de calor, consumo de eletricidade e custo de manutenção são reduzidas, através da diminuição da quantidade de dispositivos físicos utilizados.

Este artigo propõe uma arquitetura de virtualização de funções de rede para proteção automática e eficiente contra ataques de segurança. Na arquitetura proposta, um módulo de controle foi criado para gerenciar e operar VNFs que capturam amostras de tráfego, detectam ameaças e filtram atividades maliciosas de forma automática, sem impedir o tráfego benigno. A arquitetura NFV proposta aumenta a eficiência da mitigação de ataques, porque apenas o tráfego malicioso é detectado e bloqueado através do encadeamento de um sistema de detecção de intrusão (*Intrusion Detection System* - IDS) a um *firewall* de aplicação. A arquitetura foi implementada e avaliada na plataforma de código aberto para virtualização de funções de rede (*Open source Platform for Network Functions Virtualization* - OPNFV) utilizando servidores de prateleira, onde VNFs foram criadas para atuarem como IDS e *firewall* de aplicação [OPNFV 2017]. Os resultados obtidos demonstram que o sistema proposto apresenta uma alta eficiência.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute trabalhos relacionados. As características da arquitetura NFV proposta são descritas na Seção 3. Na Seção 4, os detalhes de implementação do protótipo são discutidos. Os resultados da avaliação do sistema são apresentados na Seção 5. A Seção 6 apresenta as principais conclusões e sugestões para trabalhos futuros.

2. Trabalhos Relacionados

Diversos autores utilizam propriedades das redes definidas por software (*Software Defined Networks* - SDNs) e de virtualização de funções de rede para propor soluções de gerenciamento de rede e de segurança. Deng *et al.* propuseram o *framework* VNGuard,

que utiliza os conceitos de SDN e de NFV para prover e gerenciar *firewalls* virtuais denominados VNF-FWs [Deng et al. 2015]. Os componentes do VNGuard foram implementados no ClickOS [Martins et al. 2014], que é uma plataforma de software flexível, extensível e escalável. Foram realizados experimentos que mostraram que o tempo médio de processamento por pacote na VNF-FW cresce em função do aumento da quantidade de regras do *firewall*. Todavia, a implementação de um sistema automático para proteção contra ataques não é investigada pelo autores.

Zanna *et al.* mostraram que é possível integrar um sistema de detecção de intrusão com um controlador SDN para realizar detecção e bloqueio de ataques de negação de serviço [Zanna et al. 2014]. O IDS Bro foi configurado para enviar uma requisição de criação de fluxo de bloqueio para a interface de programação (*Application Programming Interface* – API) REST do controlador Ryu, a fim de filtrar os IPs geradores do tráfego malicioso. Porém, a proposta não aborda a criação e a remoção dinâmica de regras para proteção contra ataques de exploração de vulnerabilidades.

Andreoni Lopez *et al.* propuseram o BroFlow, que combina o IDS Bro e o controlador POX para bloquear automaticamente ataques com base nos IPs de origem e de destino [Andreoni Lopez et al. 2014]. Foi implementado um algoritmo para gerar ataques de negação de serviço e um módulo foi criado no IDS Bro para programar os fluxos de bloqueio na rede. Contudo, apenas contramedidas adequadas para evitar ataques de negação de serviço são implementadas, através do bloqueio de todo tráfego gerado pelo IP de origem identificado.

Lin *et al.* propuseram uma arquitetura SDN capaz de classificar tráfego e realizar prevenção de intrusão [Lin et al. 2015]. A proposta implementa uma arquitetura capaz de detectar tanto ataques de negação de serviço quanto os que exploram vulnerabilidades, através da identificação de padrões maliciosos em requisições HTTP. Contudo, o foco do artigo é a diminuição da sobrecarga de tráfego que é enviado para os controladores SDN. A automação da detecção e do bloqueio de ataques não é abordada.

Este artigo propõe uma solução automática e integrada de detecção e mitigação de ataques de segurança usando as tecnologias NFV/SDN. A solução inclui um módulo Controlador de Segurança que recebe e analisa os alertas enviados por uma VNF-IDS e decide bloquear apenas o tráfego de ataque, sem prejudicar o tráfego benigno dos usuários que se servem do mesmo IP de origem.

3. A Arquitetura NFV Proposta

A arquitetura proposta estende a arquitetura NFV do ETSI. A seguir são descritos de forma resumida os principais componentes da arquitetura NFV do ETSI apresentada na Figura 1.

Cada VNF da arquitetura NFV está associada a um sistema de gerenciamento de elementos (*Element Management System* - EMS) que monitora a utilização de seus recursos. A infraestrutura NFV (*NFV Infrastructure* - NFVI) fornece hardware e software com os recursos de rede, de computação (memória e CPU) e de armazenamento necessários para implantar, gerenciar e executar VNFs. O Gerenciamento e Orquestração (*Management and Orchestration* - MANO) do NFV é responsável pelo gerenciamento do ciclo de vida dos recursos físicos e de software da NFVI e pela gestão do ciclo de vida das VNFs.

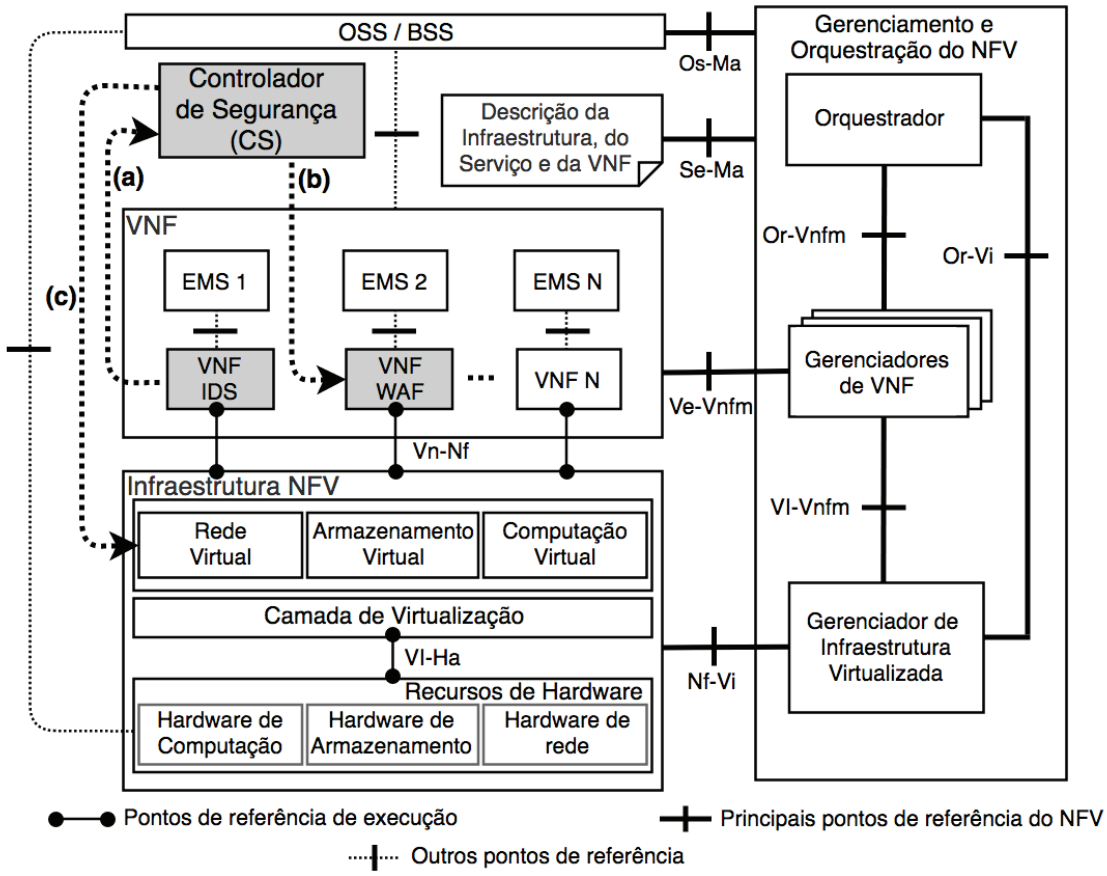


Figura 1. Arquitetura NFV estendida com os componentes de segurança propostos em cor escura. As principais interações do Controlador de Segurança proposto são: a) recebimento de alarmes; b) envio das regras atualizadas para instalação no *firewall* e c) envio de fluxos OpenFlow para serem bloqueados.

Por isso, as ferramentas necessárias para criação, atualização, migração e encerramento de VNFs estão inseridas nele.

O sistema de suporte operacional (*Operations Support System* - OSS) e o sistema de suporte empresarial (*Business Support System* - BSS) gerenciam o inventário de rede, o provisionamento de serviços, os relatórios de falhas e de faturamento etc. A interação entre os componentes da arquitetura NFV é realizada através das interfaces definidas pelo ETSI: Nf-Vi, Vi-Ha, Vn-Nf, Ve-Vnfm, Se-Ma, Os-Ma, Or-Vnfm, Or-Vi e Vi-Vnfm [ETSI 2013, ETSI 2012].

O principal objetivo da arquitetura NFV proposta é oferecer um sistema de proteção automático e eficiente contra ataques. São propostos três novos componentes na arquitetura, apresentados em cor escura na Figura 1. O principal componente é o módulo Controlador de Segurança. Ele interage com duas funções virtuais de rede específicas e com a interface de rede virtual (ver Figura 1). As duas funções virtuais de rede são um sistema de detecção de intrusão denominado VNF-IDS e um *firewall* de aplicação web denominado VNF-WAF.

A função virtual de rede de sistema de detecção de intrusão analisa os dados, detecta ameaças e envia alertas para o Controlador de Segurança. Ao receber os alertas, o

Controlador de Segurança os analisa e decide se uma atualização das regras de *firewall* é necessária. Em caso afirmativo, novas regras são enviadas à função virtual de *firewall*. O Controlador de Segurança caracteriza os alarmes e decide por uma possível estratégia de mitigação da ameaça de segurança, enviando informações para a interface de rede virtual de como determinados fluxos devem ser tratados (bloqueados etc.). Este procedimento é realizado de forma completamente automática, o que permite uma rápida reação a ataques.

4. Implementação do Sistema Proposto

A arquitetura proposta foi implementada na plataforma *Open Platform for Network Function Virtualization* (OPNFV) versão Colorado, que provê parte das funcionalidades propostas na arquitetura de virtualização de redes do ETSI. Mais especificamente, a versão Colorado provê a Infraestrutura de Virtualização de Redes (*Network Function Virtualization Infrastructure* (NFVI) e o Gerenciador de Infraestrutura Virtualizada (ver Figura 1). A OPNFV é uma plataforma de código aberto mantida por uma comunidade de desenvolvedores, que utiliza o OpenStack como Sistema Operacional (SO) de nuvem para controlar os recursos físicos da Infraestrutura NFV [OPENSTACK 2017].

Em relação ao provisionamento de redes virtuais, a plataforma OPNFV permite o uso dos controladores de redes definidas por software *OpenDaylight* (ODL) [ODL 2017] ou *Open Network Operation System* (ONOS) [ONOS 2017]. Nesta implementação, a OPNFV foi instalada com o controlador SDN *OpenDaylight*. A escolha do ODL se baseou na sua maturidade, na sua documentação de apoio e nas características de sua API. A nuvem OPNFV implantada possui três nós de computação e um nó de rede. O Controlador de Segurança e a VNF-WAF da arquitetura proposta foram implantados nos nós de computação, um componente por nó. O VNF-IDS foi implantada em uma outra máquina. Além disso, um comutador OpenFlow [Moraes et al. 2014] Open vSwitch (OVS) foi configurado em cada nó de computação. O ODL foi instalado no nó de rede da nuvem [OPENSTACK 2017].

A Figura 2 apresenta os módulos da proposta que foram implementados para mitigar ataques contra servidores web vulneráveis. As interações que ocorrem entre os módulos e a rede virtual também são apresentadas. A função virtual de rede de sistema de detecção de intrusão (VNF-IDS) proposta possui dois módulos: o módulo Detector de ameaças e o módulo Notificador de ameaças (Figura 2). O detector de ameaças foi implementado utilizando-se o Bro, que é um sistema de detecção de intrusão de rede de código aberto capaz de monitorar passivamente o tráfego de rede e procurar por atividades suspeitas [Sommer 2003]. O módulo de detecção foi configurado através de *scripts* para ser capaz de analisar e extrair informações da camada de aplicação, de modo a compará-las às atividades com padrões classificados como ataques. Dessa forma, é possível identificar ataques do tipo *cross-site scripting* (XSS) e injeção de SQL.

TAP é um tipo de operação da rede. Nesse tipo de operação, uma cópia de cada evento de rede ocorrido em uma interface é enviada para o sistema que monitora e analisa os dados. A interface de rede da VNF-IDS foi configurada em modo TAP para permitir o envio de amostras do tráfego da rede para o detector de ameaças, como mostrado na interação (1) da Figura 2. O notificador de ameaças da VNF-IDS, que foi escrito na linguagem Python, foi desenvolvido para enviar mensagens HTTP com alertas de ameaças para o controlador de Segurança (2). Na mensagem enviada, a VNF-IDS informa o

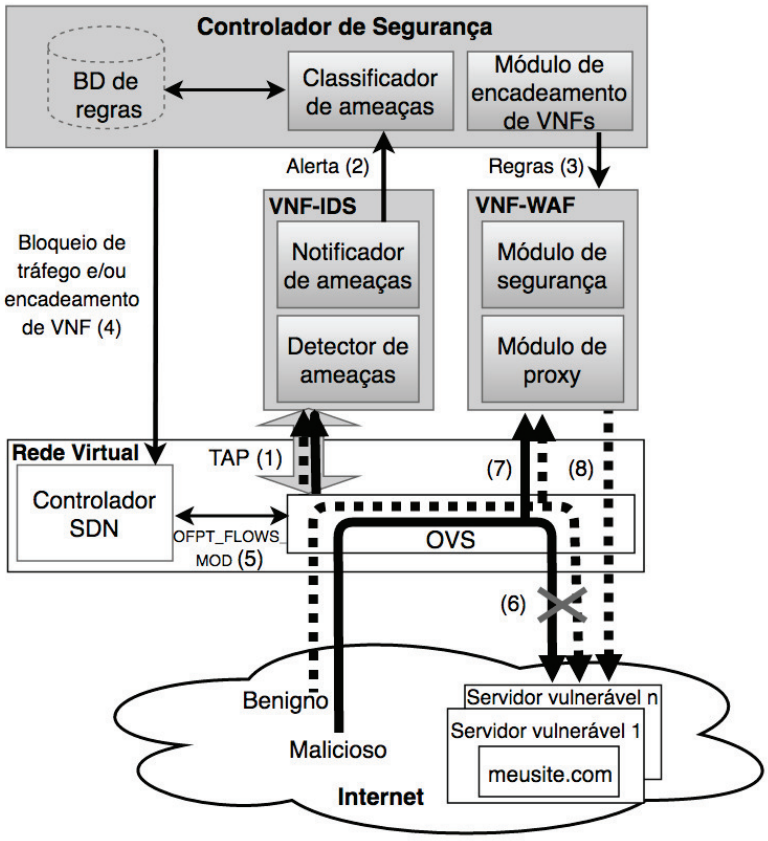


Figura 2. Arquitetura NFV implementada. Os módulos mais escuros são os propostos como extensão da arquitetura NFV do ETSI.

endereço IP que gerou o tráfego malicioso, o endereço IP atacado, a *Uniform Resource Locator* (URL) relativa ao ataque, a *Uniform Resource Identifier* (URI) e o tipo de ameaça detectada.

Um módulo classificador de ameaças, um módulo de encaminhamento de VNFs e uma base de dados (BD) de regras são implementados no controlador de segurança. Na base de dados, são armazenados os alertas de ameaça e as regras que podem ser utilizadas posteriormente no *firewall* de aplicação para bloquear ataques (Figura 2). O programa de banco de dados de código aberto MongoDB foi utilizado na criação da base de dados. A mitigação das atividades maliciosas de forma automática pode ser realizada através de uma operação de bloqueio de tráfego em todos os comutadores OVS da nuvem OPNFV ou através de uma operação de encadeamento com a VNF-FW para bloquear as atividades maliciosas, sem impedir o acesso benigno.

O Algoritmo 1 determina o tratamento dos pacotes após um ataque ser detectado. Ao receber um alerta de ataque (2), o módulo classificador de ameaças do controlador de segurança verifica se o tipo é de negação de serviço (Ψ) ou se busca explorar uma vulnerabilidade na rede ou nos sistemas protegidos pela arquitetura NFV (Φ). Se o ataque é de negação de serviço, o controlador de segurança executa uma operação de bloqueio de tráfego em todos os comutadores OVS da nuvem OPNFV. Nesse tipo de operação, o controlador de segurança envia mensagens HTTP para a API REST do *OpenDaylight* (4), que insere fluxos *OpenFlow* no OVS (5) para bloquear todo fluxo de dados gerado pelo

endereço IP de origem do ataque de negação de serviço ($\delta \rightarrow src, *$).

Algoritmo 1: TRATAMENTO DE ATAQUE.

Entrada:
$\delta = \{src, dst, tipo\}$: Ataque detectado
Ψ: Conjunto de ataques de negação de serviço
Φ: Conjunto de ataques que exploram vulnerabilidades

```
1  início
2      se δ → tipo ∈ Ψ então
3          BLOQUEIA(δ → src, *);
4      senão se δ → tipo ∉ Φ então
5          BLOQUEIA(δ → src, δ → dst);
6      senão
7          DESVIA(δ → dst);
8      fim
9      REGISTRA(δ);
10 fim
```

Se o ataque não é de negação de serviços, o classificador de ameaças verifica se a base de dados possui uma regra adequada para bloquear o ataque. Caso não exista ($\delta \rightarrow tipo \notin \Phi$), o controlador de segurança bloqueia o tráfego de ataque nos comutadores OVS ($\delta \rightarrow src, \delta \rightarrow dst$) executando os passos (4) e (5). Quando existe regra adequada na base de dados do controlador de segurança para mitigar o ataque de exploração de vulnerabilidade, o classificador de ameaças aciona o módulo de encadeamento de VNF para que uma operação de redirecionamento de tráfego seja executada, de forma a permitir que o tráfego passe pelo *firewall* instanciando na VNF-FW. Por conseguinte, o módulo de encadeamento de VNF insere a regra que mitiga o ataque no Módulo de segurança (3), configura o módulo de *proxy* da VNF-WAF e aciona o controlador SDN (4) para inserir fluxos OpenFlow de redirecionamento de tráfego no OVS ($\delta \rightarrow dst$) (5), com o propósito de desviar todo tráfego antes destinado de forma direta ao site web atacado (6) para a VNF-WAF (7).

O módulo de segurança da VNF-WAF foi construído com o ModSecurity [Ristic 2010], que é um *firewall* de aplicação web. Quando a VNF-WAF é encadeada ao fluxo de dados (7), todas as requisições HTTP destinadas ao site web vulnerável passam a ser tratadas primeiro pelo módulo de segurança e pelo módulo de *proxy* (Figura 2). O módulo de segurança remove o tráfego malicioso. O módulo de *proxy* é configurado pelo módulo de encadeamento de VNFs do controlador de segurança para atuar como "*proxy* reverso" [APACHE 2017], repassando o tráfego de rede benigno que recebe para os servidores que hospedam o site web vulnerável (8). Deste modo, a VNF-WAF remove o tráfego malicioso, sem afetar o tráfego benigno gerado a partir do mesmo endereço de origem.

5. Avaliação de Desempenho da Arquitetura Proposta

O desempenho da arquitetura proposta é avaliado através de dois experimentos. No primeiro experimento é avaliada a eficiência da detecção de ataques do módulo IDS.

Assim, é realizado uma simulação de ataques, na qual diversos tipos de ataque são realizados contra um servidor web vulnerável, de forma a verificar se os ataques praticados são detectados e se as requisições benignas continuam a ser atendidas. Em um segundo experimento, é verificado o impacto na capacidade do servidor atender às requisições HTML, ou o quanto a métrica taxa de respostas HTTP recebidas por segundo é afetada ao se utilizar a arquitetura proposta.

5.1. Capacidade da Proposta em Filtrar Ataques

De modo a avaliar o comportamento da arquitetura proposta, um cliente web envia requisições HTTP benignas e maliciosas para um servidor web. São observados quais ataques foram filtrados e quais requisições benignas passaram quando a arquitetura proposta é utilizada. Um pote de mel (*honeypot*) [Provos et al. 2004], que é uma ferramenta capaz de emular aplicações e sistemas operacionais vulneráveis, foi instalado para atuar como o servidor web. O principal propósito desse tipo de ferramenta é ser atacada e comprometida, para que novos tipos ou tendências de ataques possam ser identificados através da análise minuciosa das ações e do comportamento dos atacantes. Foram inseridas no pote de mel vulnerabilidades como XSS e injeção de SQL.

O cliente web foi configurado para enviar 220 requisições HTTP benignas e 340 requisições de ataque para o pote de mel. As requisições HTTP benignas foram enviadas com o objetivo de acessar a página principal do site web vulnerável, através do comando *WGET http://sitevulneravel/index.html*. As requisições HTTP maliciosas foram enviadas para explorar as vulnerabilidades configuradas no pote de mel. A VNF-WAF foi instalada sem regras previamente configuradas em seu módulo de segurança e nenhum *site* vulnerável foi previamente cadastrado para receber repasses de tráfego web do módulo de *p*roxy. As regras armazenadas na base de dados do Controlador de Segurança foram as necessárias para impedir todos os tipos de requisições maliciosas enviadas pelo cliente web.

A nuvem OPNFV é implantada em quatro servidores de prateleira: um nó de rede e três nós de computação. Todas as máquinas são instaladas com o sistema operacional Ubuntu 14.04.5 LTS. Nos nós de computação, a ferramenta de virtualização KVM, única opção suportada pelo OPNFV, foi utilizada. O nó de rede e um dos nós de computação da nuvem OPNFV são máquinas com processador Intel(R) Core(TM) i7-4770 CPU @ 3,40 GHz de oito núcleos, 32 GB de RAM e três interfaces Ethernet de 1 Gb/s. O segundo nó de computação possui processador Intel(R) Core(TM) i7-2660 CPU @ 3,40 GHz de oito núcleos, 32 GB de RAM e três interfaces Ethernet de 1 Gb/s. O terceiro e último nó de computação é uma máquina com dois processadores Intel(R) Xeon(R) CPU E5-2650 v2 @ 2,60 GHz de dezesseis núcleos, 32 GB de RAM e duas interfaces de 1 Gb/s.

No conjunto de nós de computação, foram criadas cinco máquinas virtuais (VMs) com acesso à Internet, uma para cada componente a seguir: Controlador de Segurança, cliente web, servidor web vulnerável, VNF-WAF e base de dados do Controlador de Segurança. Cada VM foi criada com duas CPUs virtuais e 4 GB de RAM. Por último, a VNF-IDS foi implantada numa quinta máquina com processador Intel(R) Core(TM)2 Quad CPU Q6600 @ 2,4 GHz, 4 GB de RAM e uma interface de 1 Gb/s.

Para realizar os ataques de exploração de vulnerabilidade foi desenvolvido um *script*, no qual são enviados os ataques e as requisições benignas em uma mesma ordem

Tabela 1. Exemplos de ataques detectados e não detectados.

Método	Requisição HTTP para explorar vulnerabilidade	Resposta
GET	http://siteA/?rHPbc8c=../../../../etc/passwd	200 OK
PUT	http://siteA/oRnQL com o arquivo:"9zseqh"	201 OK
GET	http://siteA/?XzuFzsw=SELECT TOP 1 name FROM sysusers	200 OK
GET	http://siteA/?rHPbc8c=ps -aux	403 Proibido

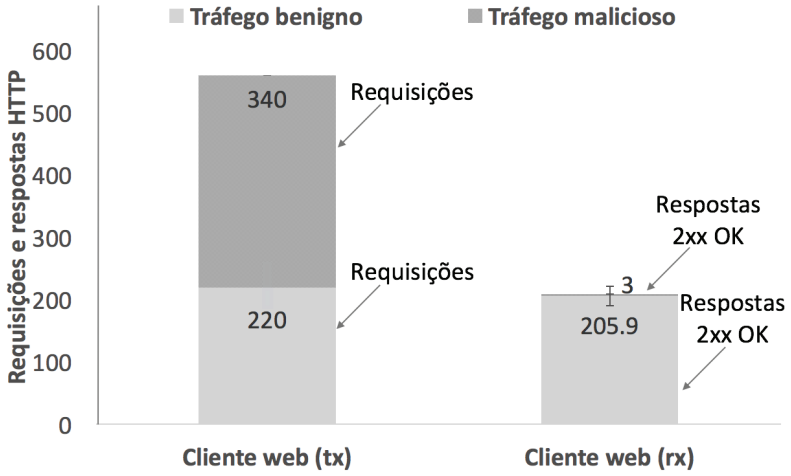
em todos os experimentos. As primeiras requisições de ataque enviadas pelo *script* em todos os experimentos estão apresentadas na Tabela 1. A primeira linha da tabela mostra que as contas de usuários existentes no sistema operacional do servidor web atacado são indevidamente listadas (200 OK) através de um GET HTTP. Na segunda linha, um ataque de inclusão remota de arquivo no servidor web atacado é realizado e comandos SQL arbitrários são enviados com sucesso para o banco de dados na terceira linha. A quarta linha mostra que uma mensagem HTTP 403 é enviada quando a arquitetura proposta bloqueia um ataque de injeção arbitrária de comandos no sistema operacional do servidor.

Em uma primeira rodada de teste, foram enviados os 340 ataques e as 220 requisições benignas para o servidor web vulnerável sem habilitar os módulos da arquitetura NFV proposta. A quantidade de respostas 2xx OK (Tabela 1) foi igual a de requisições enviadas, conforme esperado, pois não há detecção de ataques. Há necessidade de execução do teste várias vezes em função do não determinismo relacionado à remoção e à instalação de novos fluxos nos comutadores SDN. Por esse motivo, o *script* envia as requisições em dez rodadas do experimento, com os módulos da arquitetura NFV proposta habilitados. Todos os resultados apresentados são médias com intervalo de confiança de 95%. Alguns intervalos de confiança não são visualizados, por serem muito pequenos.

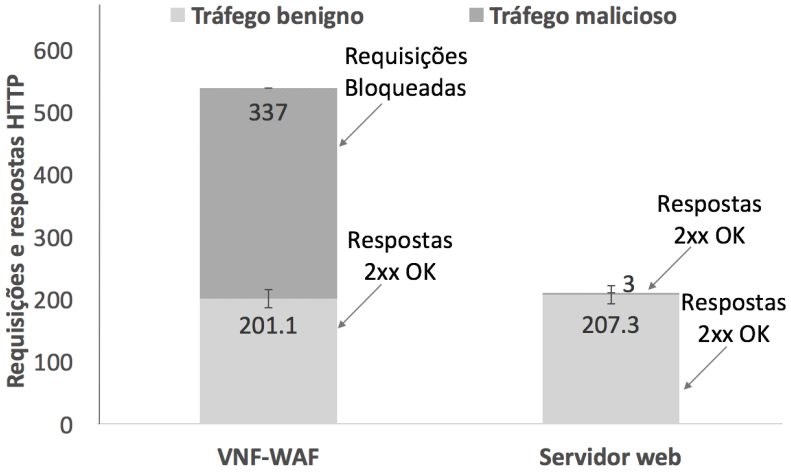
Pode-se observar pela Figura 3(a) que dos 340 ataques transmitidos pelo cliente web, apenas três alcançaram o servidor web e foram bem sucedidos (respostas 2xx OK recebidas pelo cliente web na figura). Todos os outros ataques foram detectados e bloqueados. Quando os primeiros ataques foram enviados, a VNF-IDS identificou as requisições maliciosas e notificou o Controlador de Segurança. O Controlador de Segurança automaticamente desviou o tráfego para a VNF-WAF aplicando fluxos OpenFlow no OVS, inseriu as regras de bloqueio na VNF-WAF e configurou seu Módulo de proxy. As três respostas HTTP positivas para os ataques (Figura 3(b)) foram enviadas pelo servidor web enquanto a VNF-WAF ainda não havia sido automaticamente encadeada.

Os resultados indicam que existem perdas na rede quando o Controlador de Segurança remove do comutador OVS os fluxos que permitiam a comunicação entre o cliente web e o servidor vulnerável. Esses fluxos são substituídos por outros que direcionam o tráfego para a VNF-WAF. Durante esta operação, acontecem perdas de pacotes, semelhantes às ocorridas quando um roteador de uma rede de comutação de pacotes falha. Por esse motivo, a quantidade de respostas HTTP benignas recebidas no cliente web, que em média é igual a 205,9, é menor do que as 220 requisições por ele transmitidas (ver Figura 3(a)). No entanto, 207,3 das 220 requisições benignas transmitidas alcançaram o servidor sem problemas. Portanto, a quantidade de requisições HTTP perdidas entre o cliente e o servidor vulnerável é em média igual a 12,7.

Ainda nessa linha, os resultados também mostram que 201,1 das 207,3 respostas



(a) Quantidade de requisições transmitidas e de respostas recebidas de ataques e de tráfego benigno pelo cliente.



(b) Quantidade de ataques que alcançaram o servidor web, bloqueados na VNF-WAF e quantidade de respostas enviadas pelo servidor web e que passaram pela VNF-WAF para o tráfego benigno.

Figura 3. Ataques filtrados quando a arquitetura NFV proposta é utilizada.

HTTP enviadas pelo servidor, dentro da média observada, foram respondidas quando a VNF-WAF já estava encadeada ao fluxo de dados (ver Respostas 2xx OK benignas na Figura 3(b)). Por conseguinte, aproximadamente seis respostas HTTP benignas voltaram do servidor para o cliente web sem passar pela VNF-WAF. Além disso, é possível perceber, que em média 205,9 dessas 207,3 respostas HTTP chegaram no cliente web. Portanto, a quantidade de respostas HTTP benignas perdidas entre o servidor e o cliente web é em média igual a 1,4. Por fim, a Figura 3(b) ainda mostra que, em média, depois do encadeamento da VNF, todos os 337 ataques realizados foram bloqueados.

Os resultados mostram que a arquitetura NFV proposta é eficiente, pois 99,12% dos ataques gerados foram dinamicamente bloqueados e 93,59% do tráfego benigno gerado não foi afetado quando a VNF-WAF foi dinamicamente encadeada ao fluxo de dados.

5.2. Impacto da Proposta no Desempenho da Aplicação Web

O segundo experimento consiste em variar a quantidade de requisições HTTP por segundo enviadas para o servidor web. O objetivo é verificar de que forma o encadeamento da VNF-WAF afeta a taxa de requisições e respostas HTTP por segundo. O desempenho da VNF-WAF foi avaliado no segundo experimento, quando todas as regras, adequadas para mitigar os ataques gerados pelo *script* de teste, estavam aplicadas na VNF.

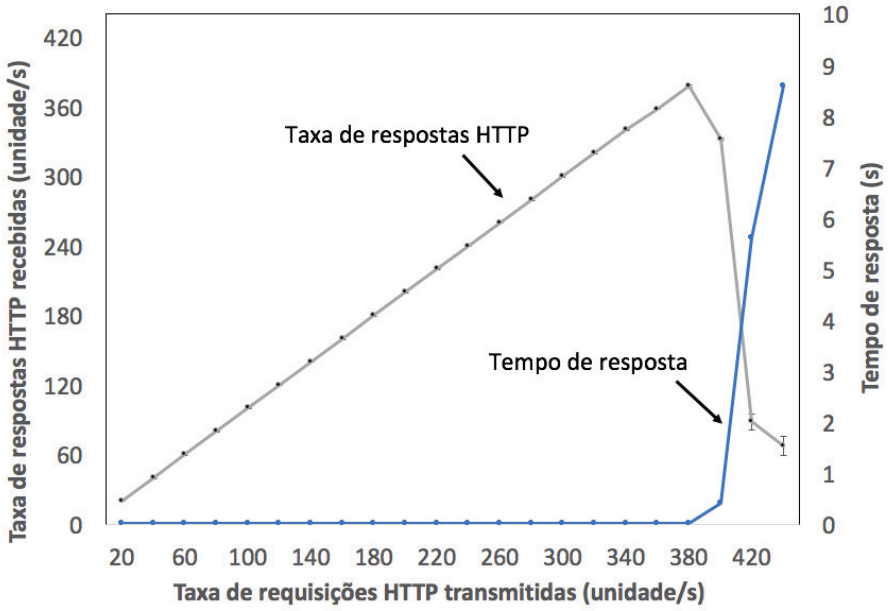


Figura 4. Desempenho da aplicação web quando a VNF-WAF não está encadeada ao fluxo de dados.

O httperf foi utilizado para gerar as requisições HTTP. Esse aplicativo permite gerar e manter taxas sustentáveis de requisições HTTP de forma a sobrecarregar um servidor. Nele é possível definir a taxa de conexões TCP por segundo, quantas requisições HTTP devem ser realizadas em cada conexão e o número máximo de requisições HTTP que o cliente deve realizar. Também é possível aumentar a taxa de conexões TCP por segundo durante um teste, definindo uma taxa inicial, o valor do incremento e a taxa máxima de conexões TCP [Mosberger and Jin 1998].

De forma a automatizar o uso do httperf, o Autobench foi utilizado. O Autobench foi configurado para variar a taxa de conexões TCP entre 10 e 220, com incremento de 10 em 10 conexões TCP por segundo [AUTOBENCH 2017]. Foram enviadas duas requisições HTTP persistentes em cada conexão TCP estabelecida com o servidor web. Portanto, a quantidade de requisições HTTP requisitadas variou entre 20 e um valor máximo de 440. Estes valores foram utilizados porque, em diversos testes preliminares, observou-se que este é um limite para postergar ao máximo o aumento significativo do tempo de resposta das transações HTTP, que é o tempo decorrido entre a requisição de um objeto e o recebimento do mesmo. Os resultados apresentados a seguir são médias com intervalo de confiança de 95%. Alguns intervalos de confiança não são visualizados nas figuras, por serem muito pequenos.

Na Figura 4 pode-se observar que a taxa máxima de requisições HTTP suportada

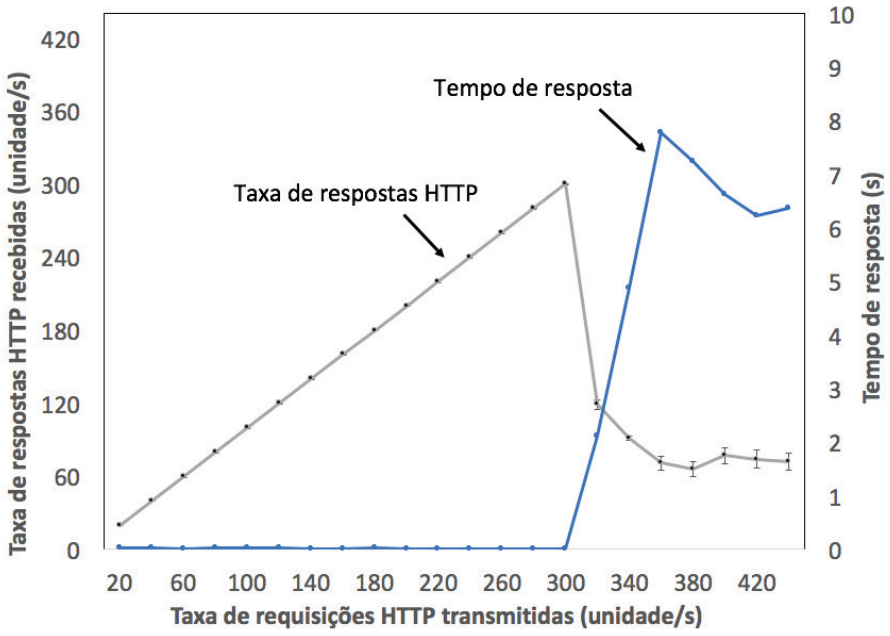


Figura 5. Desempenho da aplicação web quando a VNF-WAF está encadeada ao fluxo de dados.

pelo servidor web, sem que ocorra aumento significativo no tempo de resposta, é igual a 380 requisições por segundo. A partir deste valor, o tempo de resposta aumenta consideravelmente, saindo de aproximadamente zero para cerca de 9 s quando o cliente tenta enviar 420 requisições HTTP por segundo para o servidor.

A Figura 5 mostra que o encadeamento da VNF-WAF fez com que a capacidade de o servidor responder sem o tempo de resposta aumentar consideravelmente caísse para cerca de 300 requisições HTTP por segundo. Essa redução da capacidade foi causada pela VNF-WAF, em função das regras de ataque instanciadas. A partir desta quantidade de requisições, o tempo de resposta subiu consideravelmente, chegando a alcançar cerca de 8 s. Portanto, o encadeamento da VNF-WAF reduziu em cerca de 21% a taxa de conexão e de requisições HTTP por segundo atendidas. Entretanto, Mauricio *et al.* mostraram que é possível utilizar a escalabilidade da computação em nuvem quando é necessário escalar funções de rede virtuais criadas para atuarem como *firewalls* [Mauricio et al. 2016]. É possível escalar a VNF de forma vertical, aumentando os recursos de memória, CPU ou disco da VNF-WAF em função da demanda, ou de forma horizontal, quando o número de VNFs da arquitetura é aumentado.

6. Conclusão

A arquitetura NFV proposta mostrou-se capaz de aplicar contramedidas de forma dinâmica contra ataques. A função virtual de rede de sistema de detecção de intrusão notificou corretamente o Controlador de Segurança e seus módulos de classificação e de encadeamento conseguiram inserir corretamente o fluxo OpenFlow OFPT_FLOW_MOD no comutador SDN para desviar o tráfego para o *firewall* VNF-WAF, onde as regras de bloqueio para os ataques de exploração de vulnerabilidade foram corretamente instaladas. A arquitetura apresentou elevada eficiência, pois bloqueou 99,12% dos ataques de exploração de vulnerabilidade gerados no ambiente de teste, sem afetar consideravelmente o

tráfego benigno gerado pelo mesmo IP de origem, que continuou alcançando o servidor web. A VNF-WAF do ambiente de teste reduziu a quantidade de conexões por segundo que o servidor web consegue atender em cerca de 21%. Contudo, a escalabilidade da computação em nuvem permite que a VNF-WAF seja vertical ou horizontalmente escalada em função da demanda de tráfego. Comparada a uma solução com equipamentos tradicionais, a arquitetura NFV possui ainda outras vantagens como menor custo e maior adaptabilidade à quantidade de tráfego de rede.

Como trabalhos futuros, deseja-se estender a arquitetura NFV para implementar um módulo de varredura dinâmica de vulnerabilidades. Dessa forma, será possível identificar possíveis vulnerabilidades nas aplicações e aplicar as contramedidas descritas assim que sejam identificadas.

7. Agradecimentos

Este trabalho foi realizado com recursos da Globo (www.globo.com), INCT Internet do Futuro, CNPq, CAPES e FAPERJ.

Referências

Akamai (2017). State of the Internet - Connectivity report - Additional resources. Technical report. https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2016-state-of-the-internet-security-report.pdf.

Amann, J. and Sommer, R. (2015). Providing dynamic control to passive network security monitoring. In *International Workshop on Recent Advances in Intrusion Detection*, pages 133–152.

Anderson, R., Barton, C., Böhme, R., Clayton, R., Eeten, M. J. V., Levi, M., Moore, T., and Savage, S. (2013). *The Economics of Information Security and Privacy*, chapter Measuring the Cost of Cybercrime, pages 265–300. ISBN 978-3-642-39497-3. Springer Berlin Heidelberg.

Andreoni Lopez, M., Figueiredo, U. R., Lobato, A., and Duarte, O. C. M. B. (2014). Broflow: Um sistema eficiente de detecção e prevenção de intrusão em redes definidas por software. In *Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerfomance)*, pages 1919–1932.

APACHE (2017). Apache http server: Reverse proxy guide. https://httpd.apache.org/docs/2.4/howto/reverse_proxy.html. Acessado 07-02-2017.

AUTOBENCH (2017). Autobench: An http benchmarking suite. https://github.com/menavaur/Autobench. Acessado 17-02-2017.

Deng, J., Hu, H., Li, H., Pan, Z., Wang, K.-C., Ahn, G.-J., Bi, J., and Park, Y. (2015). VNGuard: An NFV/SDN combination framework for provisioning and managing virtual firewalls. In *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, pages 107–114.

ETSI (2012). Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. Technical report.

ETSI (2013). Network functions virtualisation (NFV); architectural framework. ETSI GS NFV 002 V1.1.1.

Gu, G., Perdisci, R., Zhang, J., Lee, W., et al. (2008). Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*, pages 139–154.

Haggard, S. and Lindsay, J. R. (2015). North Korea and the Sony hack: Exporting instability through cyberspace. *AsiaPacific Issue*, 117:1–8.

Lin, Y.-D., Lin, P.-C., Yeh, C.-H., Wang, Y.-C., and Lai, Y.-C. (2015). An extended SDN architecture for network function virtualization with a case study on intrusion prevention. *IEEE Network*, 29(3):48–53.

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., and Huici, F. (2014). ClickOS and the art of network function virtualization. In *USENIX Conference on Networked Systems Design and Implementation*, pages 459–473.

Mauricio, L. A. F., Rubinstein, M. G., and Duarte, O. C. M. B. (2016). Proposing and evaluating the performance of a firewall implemented as a virtualized network function. In *International Conference on Network of the Future (NOF)*, pages 1–3.

Moraes, I. M., Mattos, D. M. F., Ferraz, L. H. G., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M. K., de Amorim, M. D., Velloso, P. B., Duarte, O. C. M. B., and Pujolle, G. (2014). FITS: A flexible virtual network testbed architecture. *Computer Networks*, 63:221–237.

Mosberger, D. and Jin, T. (1998). Httperf — a tool for measuring web server performance. *Performance Evaluation Review*, 26(3):31–37.

Mtibaa, A., Harras, K. A., and Alnuweiri, H. (2015). From botnets to mobibots: A novel malicious communication paradigm for mobile botnets. *IEEE Communications Magazine*, 53(8):61–67.

ODL (2017). Opendaylight: Open source SDN platform. https://www.opendaylight.org/. Acessado 01-02-2017.

ONOS (2017). ONOS: Open network operating system. http://onosproject.org/. Acessado 01-02-2017.

OPENSTACK (2017). Open source software for creating private and public clouds. https://www.openstack.org/. Acessado 27-01-2017.

OPNFV (2017). Open platform for NFV. https://www.opnfv.org/. Acessado 24-01-2017.

Provos, N. et al. (2004). A virtual honeypot framework. In *USENIX Security Symposium*, pages 1–14.

Ristic, I. (2010). *ModSecurity Handbook: The Complete Guide to the Popular Open Source Web Application Firewall*. Feisty Duck, 1^{st} edition. ISBN 978-1907117022.

Sommer, R. (2003). Bro: An open source network intrusion detection system. In *DFN-Arbeitstagung über Kommunikationsnetze*, pages 273–288.

Zanna, P., O'Neill, B., Radcliffe, P., Hosseini, S., and Hoque, M. S. U. (2014). Adaptive threat management through the integration of IDS into software defined networks. In *International Conference on the Network of the Future (NOF) - Workshop on Smart Cloud Networks & Systems*, pages 1–5.