Mitigando Ataques com a Orquestração de VNFs Baseadas em Contêineres Usando Aprendizado Supervisionado

Fernando Silva[1], **Alberto E. Schaeffer-Filho**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{fsilva,alberto}@inf.ufrgs.br

Abstract. *The virtualization of network functions (Network Function Virtualization - NFV) decouples network functions from physical devices, simplifying new services deployment. Resilience enables VNFs to deal with possible problems, adapting them to changes through sensitive and immediate responses to specific changes. This paper proposes a mechanism called Intel-OCNF, which, through supervised learning, allows the identification of which network functions should be instantiated based on monitoring data to ensure the mitigation of attacks. The developed prototype was integrated with the NFVO orchestrator and operates in an automated way without dependence on the network operator's actions.*

Resumo. *A virtualização de funções de rede (Network Function Virtualization - NFV) desacopla as funções de rede dos dispositivos físicos, simplificando a implantação de novos serviços. A resiliência é o que possibilita às VNFs lidarem com possíveis problemas, adaptando-as a mudanças através de respostas sensíveis e imediatas a determinadas alterações. Neste artigo é proposto um mecanismo, chamado Intel-OCNF, que através do uso de aprendizado supervisionado permite identificar quais funções de rede devem ser instanciadas com base em dados de monitoramento, de forma a assegurar a mitigação de ataques em rede. O protótipo desenvolvido foi integrado ao orquestrador NFVO, e opera de forma automatizada e sem dependência de ações do operador de rede.*

1. Introdução

A *Network Function Virtualization (NFV)* está surgindo como um novo paradigma para o fornecimento de funções elásticas de rede executadas em plataformas de computação virtualizadas. A NFV usa técnicas de virtualização para consolidar categorias de equipamentos de rede em servidores de propósito geral. Em uma rede tradicional, cada função distinta é normalmente implementada como um *appliance* especializado baseado em *hardware* proprietário (Dominicini et al., 2017). A NFV substitui dispositivos de rede dedicados por um *software* que executa em máquinas virtuais ou contêineres. Uma função de rede, como *firewall* ou balanceador de carga, implantado no ambiente NFV, é conhecida como uma *Virtualized Network Function (VNF)*.

Tradicionalmente, o problema de identificar qual VNF instanciar é tratado manualmente por um operador de rede, que seleciona o conjunto mais apropriado de VNFs e o utiliza como entrada para abordagens automatizadas de posicionamento e encadeamento. A predição de VNF, que busca identificar qual função deve ser instanciada, a instanciação de uma nova função e a orquestração de serviços também são problemas que precisam

ser resolvidos para prover escolha e orquestração inteligente de VNFs para garantir a operação da rede, de forma correta e eficiente. Os principais desafios são como e com base em quais informações identificar automaticamente qual VNF deve ser instanciada, e posteriormente orquestrar esta VNF (Saraiva de Sousa et al., 2019).

Esse artigo apresenta um mecanismo, chamado *Intelligent Orchestration of Containerized Network Functions for Anomaly Mitigation (Intel-OCNF)*, que através do uso de aprendizado supervisionado permite identificar quais funções de rede devem ser instanciadas com base em dados de monitoramento. O objetivo principal do Intel-OCNF é definir quais funções devem ser instanciadas de forma a assegurar a resiliência da rede contra tráfego malicioso. Para minimizar o uso de recursos, e agilizar o processo de instanciação e gerenciamento, o Intel-OCNF utiliza virtualização baseada em contêineres. Esse mecanismo tem o objetivo de identificar anomalias no tráfego de rede e implantar VNFs a priori para garantir a qualidade do serviço. O protótipo desenvolvido foi integrado ao orquestrador *Network Functions Virtualization Orchestrator (NFVO)*, para fornecer orquestração inteligente de funções de rede para mitigação de anomalias. As principais contribuições desse artigo são: (i) Utilização de técnicas de *machine learning* para classificar anomalias no tráfego de rede e determinar o conjunto ideal de funções virtualizadas para mitigação; (ii) Emprego de virtualização baseada em contêineres para minimizar o uso de recursos e agilizar o processo de instanciação e gerenciamento; (iii) Monitoramento das condições da rede para determinar quais funções manter instanciadas, gerenciando assim de forma automática o ciclo de vida das VNFs.

A estrutura desse artigo é a seguinte. A Seção 2 apresenta a fundamentação teórica do trabalho. A Seção 3 é dedicada aos principais trabalhos relacionados. A Seção 4 detalha a solução de orquestração inteligente de funções de rede. A Seção 5 traz o protótipo e a avaliação dos resultados obtidos. Por fim, a Seção 6 aponta as considerações finais.

2. Fundamentação Teórica

Nesta seção, é apresentado o estado da arte com relação à virtualização de funções de rede, gerenciamento e orquestração de NFV.

2.1. Virtualização de Função de Rede

A virtualização de funções de rede visa definir um padrão de arquitetura para a substituição de dispositivos de *hardware* por dispositivos virtuais, para permitir a consolidação de muitas categorias de equipamentos de rede em servidores (Mijumbi et al., 2016). NFV envolve a implementação de funções de rede em *software* que podem ser executadas em uma variedade de *hardware* de servidores de propósito geral, capazes de serem movidas ou instanciadas em vários locais da rede, conforme necessário, sem a necessidade de instalar novos equipamentos.

A estrutura arquitetônica da NFV concentra-se nas alterações que provavelmente ocorrerão na rede de uma operadora devido ao processo de virtualização da função de rede (Bondan et al., 2019). Ou seja, a estrutura arquitetônica concentra-se nos novos blocos funcionais e pontos de referência trazidos pela virtualização das redes de uma operadora.

2.2. Gerenciamento e Orquestração de NFV

A estrutura da arquitetura de gerenciamento e orquestração (*NFV Management and Orchestration - MANO*) de funções virtualizadas de rede da *European Telecommunications*

Standards Institute (ETSI) define uma plataforma de serviço composta por um orquestrador (NFVO) e um gerenciador (*VNF Manager (VNFM)*) de VNFs, responsáveis por fornecer serviços de rede. Os serviços de rede podem ser definidos com a instanciação conjunta de VNFs e os encadeamentos entre as diferentes VNFs para realizar conjuntamente uma função mais complexa (Schardong; Nunes; Schaeffer-Filho, 2021). A funcionalidade do NFVO pode ser dividida em duas grandes categorias: orquestração de recursos e orquestração de serviços. A orquestração de recursos é usada para fornecer serviços que suportam o acesso aos recursos da *Network Functions Virtualization Infrastructure (NFVI)* de maneira abstraída, independente de quaisquer *Virtualized Infrastructure Managers (VIMs)*, bem como a governança das instâncias da VNF, que compartilham recursos da NFVI. Já a orquestração de serviços lida com a criação de serviços de ponta a ponta, compondo diferentes VNFs e o gerenciamento de topologia das instâncias de serviço de rede. Por fim, o VNFM trabalha em conjunto com os blocos funcionais VIM e NFVO, para padronizar as VNFs e aumentar a interoperabilidade de elementos de rede definidos por software, e é responsável pelo *Life Cycle Management (LCM)* das VNFs. Atualmente existem implementações de interfaces que possibilitam a orquestração das VNFs, no entanto, essas VNFs e seus encadeamentos devem ser definidos a priori por um operador. O propósito do Intel-OCNF é utilizar aprendizado de máquina para automatizar a seleção de VNFs e sua instanciação.

3. Trabalhos Relacionados

Alguns estudos recentes investigaram a identificação de qual VNF deve ser instanciada, a necessidade de novas instâncias de uma mesma VNF e o dimensionamento nas *Service Chains*. Nesta seção, são apresentados trabalhos que abordam estes conceitos.

Schardong, Nunes e Schaeffer-Filho (2018) propõem um mecanismo baseado em leilões, no qual agentes fazem lances sobre a alocação de VNFs. Agentes cognitivos são incorporados com preferências de um especialista, permitindo que eles decidam quais ações devem ser tomadas com base nas percepções do ambiente. É realizada a orquestração de funções, instanciando novas funções manualmente, prevendo qual VNF é necessária reativamente e utilizando a técnica de BDI (crença-desejo-intenção).

Zhang et al. (2017) apresentam um mecanismo baseado em uma adaptação das técnicas de aprendizado online para prever futuras cargas de trabalho das *Service Chains*, possibilitando estimar efetivamente as próximas taxas de tráfego e ajustar a implantação proativa da VNF, com orquestração de recursos das VNFs. Nessa estratégia, a instanciação de novas funções é realizada manualmente, identificando qual *Network Function (NF)* é necessária reativamente e empregando técnicas de Online Learning e Online Optimization.

Zhou e Guo (2017) propõem a mitigação de ataques DDoS com NFV e *Software Defined Networking (SDN)*, através da alocação e implantação flexíveis de recursos. Os autores incorporam monitoramento de eventos, coleta de informações e análise de dados para facilitar a detecção de atividades anômalas. Esta abordagem foca na orquestração de serviços e recursos, realizando a instanciação de novas funções de forma automática, identificando qual VNF é necessária reativamente e empregando técnicas de detecção de anomalias.

Pattaranantakul et al. (2016) relatam brevemente uma análise de ameaças no con-

texto da NFV e identificam os requisitos de segurança correspondentes, com o objetivo de estabelecer uma taxonomia abrangente de ameaças e fornecer uma diretriz para desenvolver contramedidas eficazes de segurança. Assim, apresentam um framework conceitual (SecMANO) para gerenciamento de segurança baseado em NFV e orquestração de serviços, com o objetivo de implantar e gerenciar de forma dinâmica e adaptativa as funções de segurança de acordo com as demandas dos usuários e clientes.

Alawe et al. (2018) propõem uma solução para prever a evolução do tráfego na rede com base em redes neurais, buscando escalabilidade para atender às necessidades, como provisionamento de recursos, sem degradar a Qualidade de Serviço (QoS). A solução trata de generalizar redes neurais enquanto acelera o processo de aprendizado usando agrupamento K-means e um método de Monte-Carlo. Essa solução é capaz de prever a carga futura e, combinada com um orquestrador, oferece provisionamento dinâmico e proativo de recursos.

Os trabalhos citados na sua maioria tratam do provisionamento de novas instâncias de VNFs pré-existentes, utilizando virtualização baseada em máquinas virtuais, não tratando da identificação de qual VNF deve ser instanciada e nem do seu ciclo de vida. Diferentemente, neste artigo, propomos a utilização de aprendizado supervisionado para identificação de anomalias e seleção dinâmica de VNFs de mitigação, gerenciando de forma automática o ciclo de vida das VNFs, e também usando contêineres como virtualização para uma instanciação mais rápida e menor uso de recursos.

4. Orquestração Inteligente de Funções de Rede em Contêineres para Mitigação de Anomalias

Nessa seção descreveremos o Intel-OCNF, um mecanismo para orquestração de VNFs baseado em aprendizado supervisionado para proteção da infraestrutura contra tráfego malicioso. O Intel-OCNF tem o objetivo de identificar anomalias no tráfego de rede e implantar VNFs à priori para garantir a qualidade do serviço e minimizar custos de recursos. Nas próximas subseções será detalhada a proposta, começando pelo cenário de uso, a visão geral de como funcionará a instanciação de funções sob demanda e por fim a especificação dos componentes que compõem a arquitetura do Intel-OCNF.

4.1. Cenário de Uso

Buscando diminuir os custos em *Operational Expenditure (OPEX)* e *Capital Expenditure (CAPEX)*, a virtualização de serviços de rede representa uma grande oportunidade de baixo custo para as empresas, e de negócios para os provedores de serviços. Com base neste cenário, este artigo propõe que ao contratar um serviço de infraestrutura uma empresa terá ao seu dispor serviços baseados em NFV que possibilitam instanciar funções de rede conforme a necessidade. Para isso o provedor disponibilizará também um serviço que identifica possíveis fluxos maliciosos de forma automatizada, podendo reconhecer um ataque e instanciar funções para mitigá-lo. A disponibilização de VNFs para a empresa como serviço é comparável à abordagem de *Software as a Service (SaaS)*.

Um exemplo de caso de uso, seria o Intel-OCNF identificar uma anomalia no tráfego que explora uma vulnerabilidade de *SQL Injection* em determinada aplicação. Isso levaria por exemplo à seleção de uma VNF de *Web Application Firewall (WAF)* capaz de mitigar o ataque, e o correspondente *deploy* desta função. O Intel-OCNF também

é responsável pelo gerenciamento do ciclo de vida da VNF, identificando por exemplo se foi detectada uma anomalia na qual já existe uma VNF de mitigação instanciada e se ela não é mais necessária ou precisa ser escalada. No modelo proposto, a empresa pagaria somente sob demanda pelo consumo de recursos de computação utilizados para executar as instâncias de VNFs. Assim, o Intel-OCNF seria responsável por identificar a necessidade, implantar, configurar e gerenciar a operação de novas instâncias de VNFs.

4.2. Visão Geral da Estratégia: Funções de Resiliência Sob Demanda

Em um cenário onde podemos automatizar medidas de mitigação a possíveis ataques que podem impactar a resiliência da infraestrutura, instanciando funções sob demanda e de baixo custo, o Intel-OCNF vem com a proposta de se integrar às implementações existentes de NFVO. Seu objetivo é fornecer orquestração inteligente de funções de rede para mitigação de anomalias, utilizando virtualização baseada em contêineres para minimizar o uso de recursos, e agilizar o processo de instanciação e gerenciamento.

Para automatizar o processo de instanciação de novas VNFs, o componente Intel-OCNF estende as funcionalidades do NFVO através das APIs disponíveis por implementações baseadas nas especificações da arquitetura de APIs RESTful da NFV-MANO *NFV-SOL-005* (Chatras, 2018), passando a ser o responsável por identificar quais funções de rede devem ser instanciadas e requisitar o *deploy* das mesmas. Esta identificação se dá através da coleta de informações do tráfego de rede, que passa por análises capazes de identificar possíveis anomalias e determinar qual VNF é necessária instanciar. Estas análises podem ser feitas utilizando inúmeras técnicas, no entanto, neste artigo é proposto a utilização de aprendizado de máquina *supervised learning*, pois em trabalhos como de Salman et al. (2017) é comprovada sua eficiência em aplicações de categorização de anomalias em ambientes *multi-cloud*, assim como em sistemas de detecção de intrusão (MODI et al., 2013).

4.3. Arquitetura do Intel-OCNF

Nesta seção é apresentada a arquitetura proposta, detalhando o Intel-OCNF, responsável por aplicar técnicas de aprendizado de máquina para detectar possíveis anomalias na rede e identificar qual VNF deve ser instanciada. Esta arquitetura é ilustrada na Figura 1, exemplificando como pode ser feita a captura do tráfego de rede e a integração com a arquitetura NFV da ETSI (ETSI, 2013), porém focando no detalhamento do *Traffic Categorizer and Mitigation Selection* e no *Mitigation Deployment and NS Life Cycle Management*[1].

4.3.1. Traffic Categorizer and Mitigation Selection

O *Traffic Categorizer and Mitigation Selection* é o responsável por categorizar o tráfego malicioso, buscando identificar o tipo de anomalia. Além de determinar a precisão da detecção de anomalias, ele também distingue diferentes tipos de ataques individualmente para obter uma contramedida e defesa ideal, podendo assim, identificar qual VNF poderá mitigar o ataque detectado.

[1]Nesse artigo, devido a restrições de espaço, não será descrito o componente Network Traffic Capture, responsável pela coleta de tráfego para análise no contexto do Intel-OCNF.

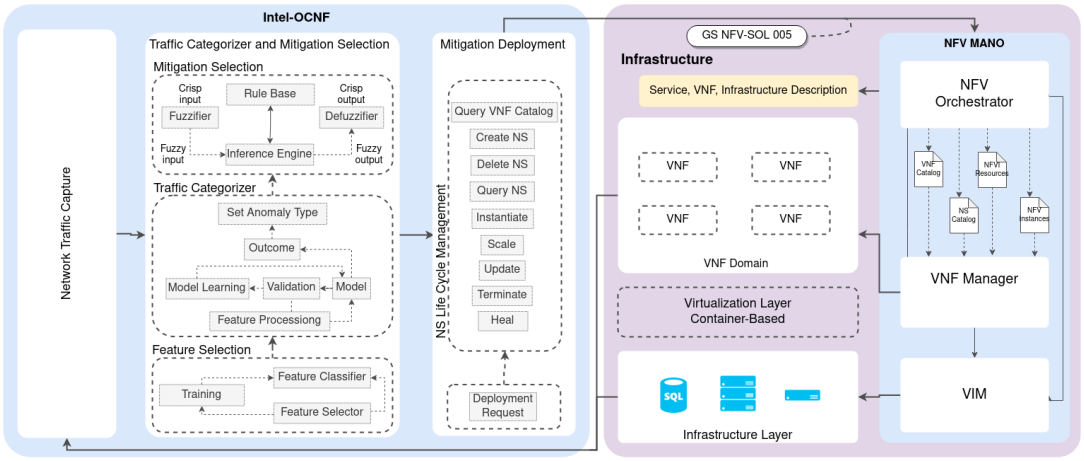


Figura 1: Detalhamento da arquitetura do Intel-OCNF

O ponto-chave para a construção de um classificador baseado em *Machine Learning (ML)* é a identificação do menor conjunto de características necessárias para atingir os objetivos de precisão, um processo conhecido como *Feature Selection* (LI; LI; LIU, 2017). A *Feature Selection* visa encontrar o melhor subconjunto de características que são relevantes para uma tarefa. Neste artigo é proposto a utilização do algoritmo *Random Forest Regressor* (RFR) baseado em *Randomized Decision Trees* para identificar as melhores características. O algoritmo *Random Forest Regressor* (SEGAL, 2004) é usado para calcular o peso de importância das características. Neste processo, na fase de treinamento do algoritmo propomos uma abordagem em duas etapas para otimização da seleção de características (KOSTAS, 2018), conforme ilustrado na Figura 2 e descrito abaixo:

- *Seleção de características entre fluxo de ataque e benigno*: Esta abordagem na seleção de características é aplicada para todo o conjunto de dados, coletando todos os tipos de ataques em um único rótulo "attack". Portanto, as características selecionadas são utilizadas para classificar qualquer tipo de ataque.
- *Seleção de características de acordo com os tipos de ataque*: Após a classificação dos ataques é feita a separação dos fluxos por tipo, isolando um tipo de ataque de outros. Assim, é possível selecionar as características ideais de cada um.

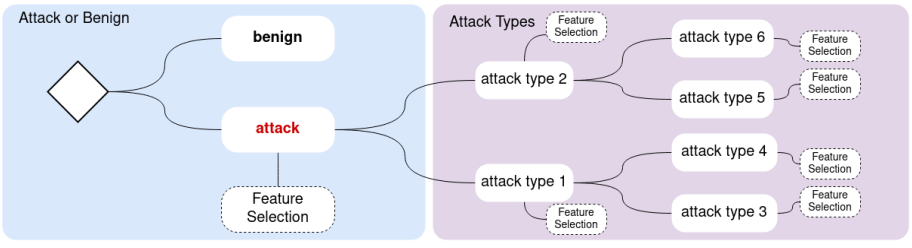


Figura 2: Processo de seleção de características

O conjunto de características a ser usado consiste em combinar as 4 características com o maior peso de importância alcançado para cada ataque. Este conjunto único de características é usado para criar os modelos de detecção de anomalias e seleção da VNF de mitigação. Para tal, são utilizados algoritmos baseados em *supervised learning*, devido a seus modelos simples de aprendizado e bom desempenho. Embora vários tipos de

técnicas de ML estejam disponíveis, as abordagens de aprendizado supervisionado são as técnicas mais populares e comumente usadas (Saravanan; Sujatha, 2018).

Algorithm 1 Traffic Categorizer and Mitigation Selection Pseudocode

```
1: dataset := (x_1, y_1),...,(x_n, y_n) the dataset is labeled with anomalies and their mitigators
2: train, test := train_test_split(dataset) split the dataset into training dataset and testing dataset
3: features := (x_1, x_2, x_3,...)
4: algorithms := (a_1, a_2, a_3,...) list of algorithm that will be applied
5: function TRAFFICCATEGORIZER(train, features, algorithms)
6:     result ← θ
7:     for i ∈ 1, ..., algorithms do apply each algorithm
8:         r_i ← LEARN(train, features, algorithms^(i))
9:         result ← result ≤ {r_i}
       end for
10:    result ← MITIGATIONSELECTION(result[anomaly], result[mitigators])
11:    return result
   end funcion
12: function LEARN(train, features, algorithm)
13:    algorithm(train, features)
14:    return The learned
   end funcion
15: function MITIGATIONSELECTION(anomaly, mitigators)
16:    apply Fuzzy Logic System
17:    result ← [a, {v_1, v_2, ..., v_N}]
18:    return result The Anomaly and Mitigation VNF
   end funcion
```

Como abordagem para obter os melhores resultados na classificação, propomos a utilização de um conjunto de algoritmos, baseado no conceito de *ensemble learning* (KRAWCZYK et al., 2017). De acordo com o pseudocódigo ilustrado no Algoritmo 1, como pré-condição o algoritmo espera um conjunto de dados de treinamento ***dataset*** (linha 1), um conjunto de características ***features*** (linha 3) e a listagem dos algoritmos que serão utilizados ***algorithms*** (linha 4). O *dataset* de entrada é separado entre dados de treinamento e dados de teste (linha 2). O algoritmo funciona da seguinte maneira: serão utilizados quatro algoritmos (Naive Bayes, Random Forest, K Nearest Neighbours e Quadratic Discriminant Analysis (QDA)), os mesmos serão aplicados na fase de treinamento dos modelos e será utilizado o algoritmo com melhor resultado obtido após o treinamento (linha 6-9 e linha 12-14). Após o tráfego categorizado é feita a seleção das VNFs de mitigação (linha 15-18) aplicando o conceito de *fuzzy logic* (PAUNOVIĆ et al., 2018), retornando como resultado a anomalia encontrada e as funções de mitigação.

Mais especificamente, com base nos modelos de treinamento, é determinado o tipo de anomalia e através de um *Fuzzy Logic System (FLS)* as VNFs de mitigação, p. ex., *S = (a, V) onde V =* $\{v_1, v_2, \ldots, v_N\}$, *a* é a identificação da categoria da anomalia e v_1 a VNF de mitigação. Cada anomalia identificada é submetida ao FLS, compreendendo a etapa de *fuzzification*, responsável por transformar as entradas do sistema em etapas fuzzy; *rule base*, usado para armazenar o conjunto de regras e as condições If-Then usadas para controlar a tomada de decisão; *inference engine*, responsável, por processar as informações; e *defuzzification*, responsável por transformar os valores de saída. O FLS identifica as VNFs que mais se adequem para mitigar a anomalia, com base nas regras fornecidas por um especialista, p. ex., `IF anomaly=(DoS com grau de incerteza 0.75) THEN vnf=(Shield DoS e Firewall)`, onde `anomaly` pode ser descrita por um conjunto *A=(*$a_1, a_2, ..., a_n$*)*, sendo A, composto por um ou mais termos e seu grau de incerteza, conectados por operadores lógicos AND ou OR, e `vnf` pode ser descrita pelo conjunto *P=(*$p_1, p_2, ..., p_n$*)*, sendo P o conjunto de VNFs de mitigação. Não necessariamente cada anomalia será mitigada por uma única VNF. Por exemplo, identificado

um ataque *DoS* que pode ser mitigado por duas VNFs, uma chamada *Shield DoS* e outra chamada *Firewall*, a = *DoS* e $V = \{v_1, v_2\}$, onde v_1 é o *Shield DoS* e v_2 é o *Firewall*. Com o resultado da categorização de tráfego, identificação das anomalias e as VNFs de mitigação, é possível instanciar dinamicamente as VNFs.

4.3.2. Mitigation Deployment and NS Life Cycle Management

O *Mitigation Deployment* é o componente que faz a conexão com o NFVO e requisita as ações necessárias para o *deploy* das VNFs. A conexão com o NFVO é realizada através das APIs disponíveis por implementações baseadas nas especificações da arquitetura de APIs RESTful da NFV-MANO *NFV-SOL-005* (Chatras, 2018). Uma das especificações da *NFV-SOL-005* é a interface LCM, que implementa o conjunto de ações de gerenciamento do ciclo de vida da *Network Service (NS)*, desde a criação de uma instância até sua remoção. A Figura 3 ilustra a sequência de troca de mensagens da integração entre o componente *Mitigation Deployment* e o NFVO durante a criação de uma NS e o gerenciamento do seu ciclo de vida.

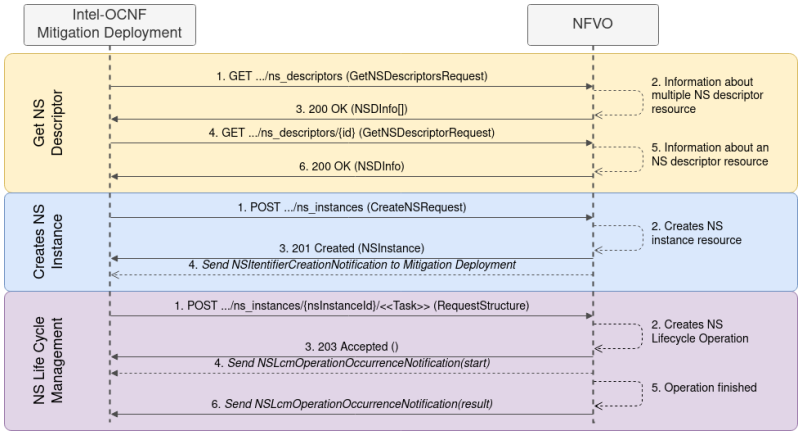


Figura 3: Fluxo de operações de integração com NFVO

O processo de *deploy* é iniciado enviando uma requisição ao NFVO, para consultar os *NS Descriptors (NSD)*. Essa requisição retorna todos os *NS Descriptors* disponíveis. Cada *NS Descriptor* é identificado através de uma *tag* composta por tipo e versão. O *Mitigation Deployment* seleciona o *NS Descriptor* que deve ser utilizado e envia uma nova requisição ao NFVO, passando o *NSD Info ID*. Esta nova requisição busca as informações do *NS Descriptor*, e com estas informações é iniciado o processo de criação da instância da NS. Para isso é enviada uma requisição ao recurso *ns_instances*. O corpo da mensagem contém, entre outros itens, uma referência ao *VNF Descriptor (VNFD)* que o VNFM precisará ler para determinar a sequência de ações a serem executadas. Após a criação da instância da NS é retornado o *NS Instance ID*, utilizado nas próximas requisições para gerenciar o ciclo de vida das *NSs*. O ciclo de vida da instância de uma NS é gerenciado enviando requisições aos recursos da *ns_instances* criada.

5. Protótipo e Avaliação

Nesta seção, são apresentados o protótipo e a avaliação preliminar do Intel-OCNF.

5.1. Implementação

O protótipo[2] foi construído usando a linguagem Python, a plataforma Open Source MANO - (OSM) v. 8.0.4, e as ferramentas Docker, Kubernetes e OpenStack. Python foi usado para implementar os algoritmos de seleção de características, detecção de anomalias e mitigação. O OSM é usado, pois, implementa a *stack MANO* e sua plataforma já suporta emulação de VIM (Vim-emu) (Peuster; Karl; van Rossem, 2016) baseado em contêineres utilizando Docker e Kubernetes, e também implementa a *North Bound Interface (NBI)* API baseada na especificação ETSI SOL005.

Para implementar o algoritmo de seleção de características é utilizada a classe *RandomForestRegressor* da biblioteca Sklearn[3] baseada no algoritmo *Random Forest*, na *Categorização do Tráfego* são utilizadas as classes *GaussianNB*, *RandomForestClassifier*, *KNeighborsClassifier* e *QDA*, ambas da biblioteca Sklearn baseadas nos algoritmos (Naive Bayes, Random Forest, K Nearest Neighbours e QDA).

5.2. Avaliação Experimental

Nesta seção, é apresentada em detalhes a avaliação experimental do protótipo, iniciando com o detalhamento do cenário, topologia, *dataset*, as métricas aplicadas para medir os experimentos e por fim, trazendo uma avaliação dos resultados.

5.2.1. Setup: Cenário, Topologias, Datasets

O cenário utilizado para a avaliação experimental usa dois computadores para emular o ambiente, um para executar o OSM e outro para executar o Intel-OCNF (onde são executados os algoritmos de aprendizado de máquina). O computador utilizado para executar o OSM possui um processador *AMD EPYC 7282 16-Core Processor CPU @ 2.79GHz X 4*, *8 GB* de memória e sistema operacional *Linux Ubuntu 18.04 64-bit*. Já o computador onde é executado o componente Intel-OCNF possui um processador *Intel® Core™ i7-8550U CPU @ 1.80GHz X 8*, *8 GB* de memória e sistema operacional *Linux Fedora 33 (Workstation Edition) 64-bit*. O cenário se baseia na ideia de prover um serviço que identifique possíveis fluxos de tráfego maliciosos em uma rede, reconhecendo um ataque e instanciando funções de rede para mitigá-lo.

Para a realização dos experimentos, foi definida uma topologia *FatTree* conforme Figura 4. Esta topologia é composta por um *Monitor* que coleta o tráfego do *switch* e envia para o Intel-OCNF que faz a classificação, identificação do ataque e da VNF que pode mitigá-lo. Como conjunto de dados para realização dos experimentos foi utilizado o *dataset* CIC-IDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Este *dataset* foi usado na fase de implementação para treinamento e testes dos algoritmos de aprendizado de máquina. Este conjunto de dados consiste em fluxos de rede rotulados, incluindo cargas úteis de pacotes completos em formato *pcap*. Trata-se de um *dataset* atualizado, oferece uma ampla diversidade de ataques e está publicamente disponível para pesquisadores. O conjunto de dados contém 3.119.345 fluxos de tráfego, incluindo diversos tipos de ataques. A distribuição desses fluxos pode ser vista na Tabela 1.

[2]https://github.com/fernandodebrando/Intel-OCNF

[3]https://scikit-learn.org/

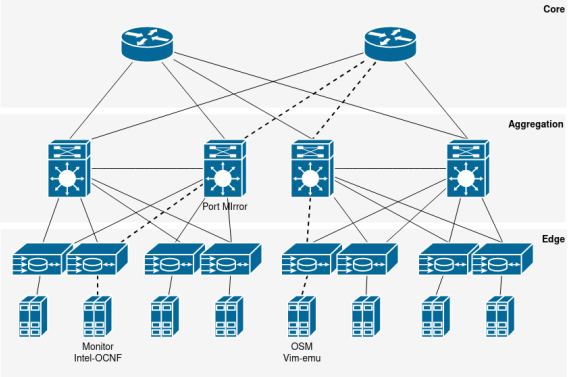


Figura 4: Topologia utilizada para os experimentos

O conjunto de características a ser usado consiste em combinar as 4 características com o maior peso de importância alcançado para cada ataque[4]. Assim, 4 características foram obtidas de cada uma das 12 categorias de ataque, resultando em um grupo de 48 características. Após, foram removidas as repetições, resultando em uma lista de 18 características. Outra forma de selecionar as características é usá-las com alta ponderação de acordo com as pontuações de importância obtidas para todo o conjunto de dados na abordagem seleção de características entre fluxo de ataque e benigno (Seção 4.3.1), sem usar todas as 18 características. Foi definido um valor limite para o peso das características, em 0,8%. Desta forma, 97% do peso total da importância das características será coberto selecionando apenas 7 características. As 11 características restantes constituem apenas 3% do peso total de significância.

Tabela 1: Distribuição de registros de fluxo no conjunto de dados CIC-IDS2017

Tipo	Registros	Tipo	Registros
BENIGN	2.359.289	DoS slowloris	5.796
DoS Hulk	231.073	DoS Slowhttptest	5.499
PortScan	158.930	Web Attack	2.180
DDoS	41.835	Bot	1.966
DoS GoldenEye	10.293	Infiltration	36
FTP-Patator	7.938	Heartbleed	11
SSH-Patator	5.897		

5.2.2. Métricas

Os resultados foram avaliados com base no desempenho do algoritmo de categorização do tráfego e seleção do mitigador, para isso foram utilizados cinco critérios (*accuracy*, *precision*, *f1-score*, *recall* e *processing time*). Todos esses critérios (excluindo o *processing time*) têm um valor entre 0 e 1. Quando se aproxima de 1, o desempenho aumenta, e quando se aproxima de 0, diminui (Bhuyan; Bhattacharyya; Kalita, 2014).

Outro método utilizado para avaliar é a comparação entre os algoritmos de aprendizado de máquina utilizados (*Naive Bayes, Random Forest, K Nearest Neighbours*

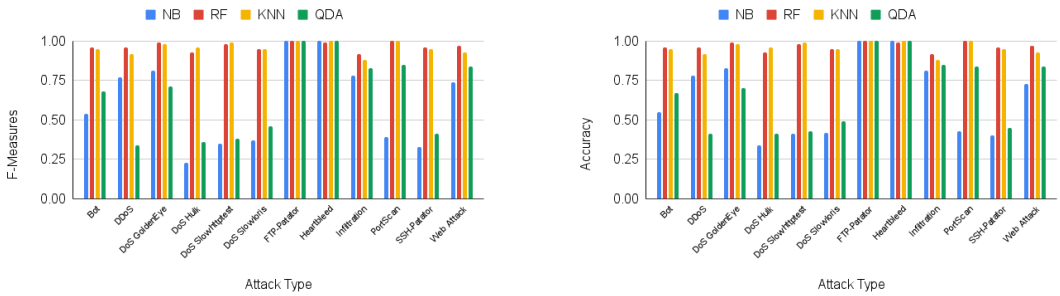
[4]Esta escolha se dá porque as 4 características com maior peso de importância representam, em 9 das 12 categorias de ataques, mais de 95% do peso total da importância das características.

e QDA). Estes algoritmos foram escolhidos considerando sua utilização em diferentes formas de classificação. Por exemplo, normalmente NB é utilizado para classificar frequência das informações, RF para classificação de conjuntos, KNN para medidas de similaridade e QDA para análise discriminante quadrática. Com isso, objetiva-se observar a eficácia e o desempenho de diferentes algoritmos de aprendizado em diferentes categorias de ataques, para utilizar aquele com melhor resultado obtido após o treinamento.

Por fim, também foi avaliado o desempenho das VNFs em contêineres, considerando o tempo de inicialização da VNF, tamanho da imagem, consumo de CPU e memória. O consumo de CPU e memória é avaliado com o apoio da ferramenta de *benchmarking ab*[5], para configurar 10 clientes e enviar 1.000 requisições por cliente para avaliar o impacto das requisições no serviço. Para avaliar o desempenho das VNFs em contêineres, foi utilizada uma VNF do *Snort*[6], um serviço de *Intrusion Detection System (IDS)*, em duas plataformas diferentes: contêineres e VM. O objetivo principal é entender o impacto das plataformas de virtualização no desempenho das VNFs.

5.2.3. Experimentos

Nesta seção, são apresentados os resultados dos experimentos. Iniciando com um comparativo de desempenho entre os quatro algoritmos de aprendizado de máquina, aplicando aos mesmos as 12 categorias de ataques, os resultados são apresentados na Figura 5.



(a) F1-score **(b) Acurácia**

Figura 5: Resultados segundo tipos de ataques e algoritmos de aprendizado

Ao analisar os resultados, nota-se que os algoritmos Random Forest e KNN só não alcaçaram mais de 90% de sucesso na detecção de uma categoria de ataque, levando em consideração as medidas F1-score e acurácia. Entre esses 2 algoritmos, o Random Forest, que é o mais bem-sucedido, classificou 8 das 12 categorias de ataques com a pontuação mais alta. Um ponto interessante é que Naive Bayes se destaca em 5 das categorias de ataques quando se trata de velocidade, como pode ser visto no comparativo da Tabela 2, onde também é apresentado o tempo de processamento de cada algoritmo (os piores resultados são apresentados em sublinhado e os melhores em negrito).

Analisando os resultados da avaliação de desempenho das VNFs em diferentes plataformas de virtualização, observamos que o tamanho da imagem do contêiner (120

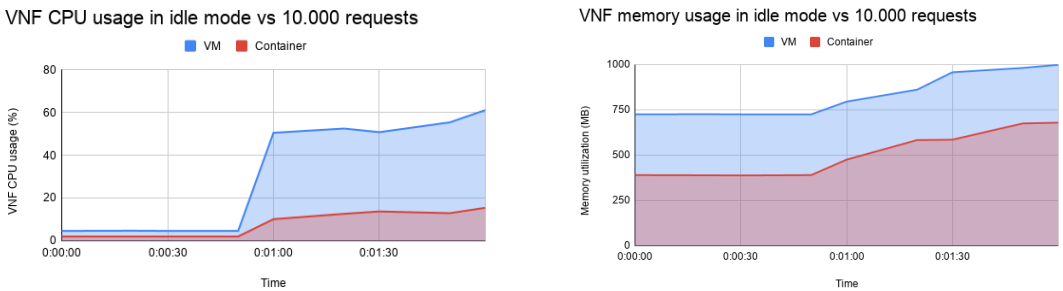
[5]https://httpd.apache.org/docs/2.4/programs/ab.html
[6]https://www.snort.org/

Tabela 2: Resultados segundo tipo de ataque, algoritmo e tempo de execução

Attack Names	Accuracy				F1-Score				Processing Time (s)			
	NB	RF	KNN	QDA	NB	RF	KNN	QDA	NB	RF	KNN	QDA
Bot	0.55	**0.96**	0.95	0.67	0.54	**0.96**	0.95	0.68	**0.0031**	0.0281	0.0176	0.0225
DDoS	0.78	**0.96**	0.92	0.41	0.77	**0.96**	0.92	0.34	0.0491	0.4187	**0.92**	0.0541
DoS GoldenEye	0.83	**0.99**	0.98	0.70	0.81	**0.99**	0.98	0.71	**0.0094**	0.0884	0.1172	0.0141
DoS Hulk	0.34	0.93	**0.96**	0.41	0.23	0.93	**0.96**	0.36	0.3127	3.4215	219.6141	**0.353**
DoS Slowhttptest	0.41	0.98	**0.99**	0.43	0.35	0.98	**0.99**	0.38	0.0073	0.0516	0.0845	**0.062**
DoS Slowloris	0.42	**0.95**	0.95	0.49	0.37	**0.95**	0.95	0.46	**0.0059**	0.0513	0.0419	0.0078
FTP-Patator	**1.0**	1.0	1.0	1.0	**1.0**	1.0	1.0	1.0	**0.0078**	0.0544	0.2547	0.0083
Heartbleed	**1.0**	0.99	1.0	1.0	**1.0**	0.99	1.0	1.0	0.0047	0.0107	**0.0031**	0.0031
Infiltration	0.81	**0.92**	0.88	0.85	0.78	**0.92**	0.88	0.83	0.0031	0.0016	0.0031	**0.0016**
PortScan	0.43	**1.0**	1.0	0.84	0.39	**1.0**	1.0	0.85	**0.2033**	2.2375	49.3462	0.2342
SSH-Patator	0.40	**0.96**	0.95	0.45	0.33	**0.96**	0.95	0.41	0.0078	**0.068**	0.0483	0.0084
Web Attack	0.73	**0.97**	0.93	0.84	0.74	**0.97**	0.93	0.84	0.0044	0.0317	0.0174	**0.0042**

MB) é significativamente menor do que na VM. O principal motivo vem do fato de que os contêineres contêm apenas as dependências necessárias para executar o serviço. A VM tem seu tamanho muito maior devido ao sistema operacional que está sendo executado dentro dela, podendo atingir cerca de 14 GB. Quando analisamos o tempo de inicialização da VNF, a VNF em contêiner mostrou-se muito mais rápida que a VNF baseada em VM – o contêiner registrou um tempo de inicialização de 2,47 segundos, enquanto a VM registrou um tempo de inicialização de 5,95 minutos. Isso ilustra que a VNF em contêiner apresenta um provisionamento muito mais rápido do que a VNF baseada em VM.

Na Figura 6 é comparado o consumo de CPU e de memória, na execução em VM e em contêiner. No primeiro minuto, é mostrada a utilização da CPU e da memória da VNF em modo ocioso. A utilização é comparativamente maior para VM devido ao kernel *guest* e sistema operacional *guest*, respectivamente. Já o contêiner mantém a CPU menos ocupada devido a seu gerenciamento de memória eficiente usando *cgroups*. A partir do primeiro minuto, são exibidas informações após o início da execução do teste de *benchmark*, com 1.000 requisições de 10 clientes em paralelo. Como esperado, o consumo de CPU no contêiner apresenta desempenho entre 70-80% superior que a VM e o consumo de memória entre 40-50% melhor que a VM, devido à VM sofrer penalidade na sobrecarga adicionada pelo *hipervisor*.



(a) CPU **(b) Memória**

Figura 6: Resultado da avaliação de desempenho das VNFs em contêineres

6. Conclusão

Nesse artigo foi proposta uma extensão da arquitetura NFV, para fornecer gerenciamento e orquestração de VNFs para mitigação de ataques. Um novo componente, o Intel-OCNF, foi projetado para integrar a arquitetura NFV, de modo a fornecer orquestração inteligente de funções de rede. Ao implementar o protótipo da nossa solução, demonstramos a viabilidade da abordagem, implementando técnicas de aprendizado supervisionado para detecção de anomalias, categorização de ataques e seleção de mitigadores.

Os resultados demonstram que com a utilização dos algoritmos selecionados (Naive Bayes, Random Forest, K Nearest Neighbours e QDA) é possível alcançar mais de 90% de sucesso na detecção das categorias de ataque e seleção da VNF de mitigação. E com a utilização de virtualização baseada em contêiner é possível provisionar uma VNF com *overhead* muito menor que o de uma VNF baseada em VM. Como trabalhos futuros, pretendemos fornecer uma especificação detalhada para integração da nossa solução ao NFVO. Além disso, pretendemos estender a avaliação da nossa proposta com outros *datasets*, a fim de avaliar a generalidade da solução.

Agradecimentos

Os autores agradecem o suporte do CNPq e FAPERGS para a condução dessa pesquisa. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Alawe, I. et al. An efficient and lightweight load forecasting for proactive scaling in 5g mobile networks. In: *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*. [S.l.: s.n.], 2018. p. 1–6.

Bhuyan, M. H.; Bhattacharyya, D. K.; Kalita, J. K. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 303–336, 2014.

Bondan, L. et al. Fende: Marketplace-based distribution, execution, and life cycle management of vnfs. *IEEE Communications Magazine*, v. 57, n. 1, p. 13–19, 2019.

Chatras, B. On the standardization of nfv management and orchestration apis. *IEEE Communications Standards Magazine*, v. 2, n. 4, p. 66–71, 2018.

Dominicini, C. K. et al. Virtphy: Fully programmable nfv orchestration architecture for edge data centers. *IEEE Transactions on Network and Service Management*, v. 14, n. 4, p. 817–830, 2017.

ETSI, G. Network functions virtualisation (nfv): Architectural framework. *ETsI Gs NFV*, v. 2, n. 2, p. V1, 2013.

KOSTAS, K. Anomaly detection in networks using machine learning. *Research Proposal*, v. 23, 2018.

KRAWCZYK, B. et al. Ensemble learning for data stream analysis: A survey. *Information Fusion*, v. 37, p. 132–156, 2017. ISSN 1566-2535. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1566253516302329⟩.

LI, Y.; LI, T.; LIU, H. Recent advances in feature selection and its applications. *Knowledge and Information Systems*, Springer, v. 53, n. 3, p. 551–577, 2017.

Mijumbi, R. et al. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, v. 18, n. 1, p. 236–262, Firstquarter 2016.

MODI, C. et al. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, v. 36, n. 1, p. 42 – 57, 2013. ISSN 1084-8045.

Pattaranantakul, M. et al. Secmano: Towards network functions virtualization (nfv) based security management and orchestration. In: *2016 IEEE Trustcom/BigDataSE/ISPA*. [S.l.: s.n.], 2016. p. 598–605.

PAUNOVIĆ, M. et al. Two-stage fuzzy logic model for cloud service supplier selection and evaluation. *Mathematical Problems in Engineering*, Hindawi, v. 2018, 2018.

Peuster, M.; Karl, H.; van Rossem, S. Medicine: Rapid prototyping of production-ready network services in multi-pop environments. In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. [S.l.: s.n.], 2016. p. 148–153.

Salman, T. et al. Machine learning for anomaly detection and categorization in multi-cloud environments. In: *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. [S.l.: s.n.], 2017. p. 97–103.

Saraiva de Sousa, N. F. et al. Network service orchestration: A survey. *Computer Communications*, v. 142-143, p. 69–94, 2019. ISSN 0140-3664. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0140366418309502⟩.

Saravanan, R.; Sujatha, P. A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification. In: *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. [S.l.: s.n.], 2018. p. 945–949.

Schardong, F.; Nunes, I.; Schaeffer-Filho, A. Providing cognitive components with a bidding heuristic for emergent nfv orchestration. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2018. p. 1–9. ISSN 2374-9709.

Schardong, F.; Nunes, I.; Schaeffer-Filho, A. NFV resource allocation: a systematic review and taxonomy of VNF forwarding graph embedding. *Computer Networks*, v. 185, p. 107726, 2021. ISSN 1389-1286.

SEGAL, M. R. Machine learning benchmarks and random forest regression. 2004.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: *ICISSP*. [S.l.: s.n.], 2018. p. 108–116.

Zhang, X. et al. Proactive vnf provisioning with multi-timescale cloud resources: Fusing online learning and online optimization. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. [S.l.: s.n.], 2017. p. 1–9.

Zhou, L.; Guo, H. Applying nfv/sdn in mitigating ddos attacks. In: *TENCON 2017 - 2017 IEEE Region 10 Conference*. [S.l.: s.n.], 2017. p. 2061–2066.