UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUCAS TELES DE OLIVEIRA

UMA ANÁLISE DO CONSUMO DE RECURSOS COMPUTACIONAIS DE UM IDS BASEADO EM NFV USANDO CONTÊINERES

CURITIBA

2023

LUCAS TELES DE OLIVEIRA

UMA ANÁLISE DO CONSUMO DE RECURSOS COMPUTACIONAIS DE UM IDS BASEADO EM NFV USANDO CONTÊINERES

An analysis of computational resource consumption of an NFV-based IDS using containers

Dissertação apresentado como requisito para obtenção do título de Mestrado em Computação Aplicada do Programa de Pós-graduação em Computação Aplicada, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Daniel Fernando Pigatto

Coorientador: Prof. Dra. Juliana de Santi

CURITIBA

2023

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

LUCAS TELES DE OLIVEIRA

UMA ANÁLISE DO CONSUMO DE RECURSOS COMPUTACIONAIS DE UM IDS BASEADO EM NFV USANDO CONTÊINERES

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Computação Aplicada da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Sistemas Computacionais.

Data de aprovação: 17 de Agosto de 2023

Dr. Daniel Fernando Pigatto, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Andre Ricardo Abed Gregio, Doutorado - Universidade Federal do Paraná (Ufpr)

Dra. Juliana De Santi, Doutorado - Universidade Tecnológica Federal do Paraná

Natassya Barlate Floro Da Silva, - Universidade Tecnológica Federal do Paraná

Dedico este trabalho de mestrado à minha família, que sempre esteve ao meu lado me incentivando. Também dedico este trabalho aos meus orientadores que me guiaram com sabedoria e paciência durante esta jornada.

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas que contribuíram para a realização deste trabalho.

Primeiramente, gostaria de agradecer a meus orientadores, por todo o suporte, orientação e incentivo ao longo deste processo. Seus valiosos conselhos e sugestões foram fundamentais para o desenvolvimento deste estudo.

Agradeço a minha família, que sempre me apoiou e me encorajou em todos os momentos. Obrigado por estarem sempre presentes e por serem uma fonte constante de amor e inspiração.

A NFV nasceu para quebrar paradigmas e remover a dependência de hardwares proprietários, possibilitando a escalabilidade com custos menores.

RESUMO

Com o aumento de usuários conectados às redes de telecomunicações e de computadores e a inerente busca pela redução de custos, surge a necessidade de melhorar o desempenho de funções de rede, tais como sistemas de detecção de intrusão, *firewalls* e balanceadores de carga. Em diversos casos, um caminho natural consiste no investimento em novos *hardwares*, que normalmente apresentam custos elevados. A Virtualização de Funções de Rede (*Network Functions Virtualization* – NFV) busca quebrar um paradigma nas redes de telecomunicações, possibilitando o desenvolvimento de novos modelos de negócios, além de redução nos custos com aquisição de novos elementos de *hardware*. Ao executar uma função de rede virtualizada em um contêiner utilizando a tecnologia Docker é possível ter um alto grau de portabilidade nos ambientes operacionais, permitindo a movimentação de uma função de rede virtualizada que está em desenvolvimento ao teste na produção sem a necessidade de fazer alterações ao longo do caminho. A Virtualização de Funções de Rede permitiu abrir novos caminhos para atacantes cibernéticos, tornando necessário aderir a mecanismos de proteção, como os sistemas de detecção de intrusão (*Intrusion Detection System* – IDS). Estes sistemas, quando baseados em funções de rede virtualizadas, podem tornar possível a mobilidade e escalabilidade conforme a demanda, além de reduzir investimentos em bens de capital (CAPEX) e permitir a disponibilização de novos serviços sem necessidade de trocar equipamentos. Nesse sentido, este trabalho propõe: (1) uma revisão sobre funções de rede virtualizadas e sistemas detecção de intrusão; (2) revisão sobre ferramentas de virtualização de funções de rede e levantamento de vantagens e desvantagens de cada uma; (3) execução de um sistema de detecção de intrusão *open source* usando virtualização de funções de rede; (4) elaboração e montagem de um ambiente virtualizado para os testes. Para fins de avaliação desta dissertação, desenvolveu-se duas análises das funções virtualizadas: (1) utilizando máquinas virtuais (simulando um sistema completo); (2) utilizando contêineres (compartilhando o *kernel* do sistema hospedeiro sem necessidade de emular *hardware* completo).

Palavras-chave: avaliação de desempenho; loic; redes de computadores; sistemas de detecção de intrusão; virtualização de funções de rede.

ABSTRACT

With the increase of users connected to telecommunications and computer networks and the inherent search for cost reduction, there is a need to improve the performance of network functions, such as intrusion detection systems, *firewalls* and security balancers. charge. In many cases, a natural path is to invest in new *hardware*, which usually has high costs. The Network Functions Virtualization (*Network Functions Virtualization* – NFV) seeks to break a paradigm in telecommunications networks, enabling the development of new business models, in addition to reducing costs with the acquisition of new *elements hardware*. When running a virtualized network function in a container using Docker technology, it is possible to have a high degree of portability across operating environments, allowing the movement of a virtualized network function that is in development to test in production without the need to make changes to the along the way. Network Functions Virtualization has opened new avenues for cyber attackers, making it necessary to adhere to protection mechanisms such as intrusion detection systems (*Intrusion Detection System* – IDS). These systems, when based on virtualized network functions, can make mobility and scalability possible according to demand, in addition to reducing investments in capital goods (CAPEX) and allowing the availability of new services without the need to change equipment. In this sense, this work proposes: (1) A review of virtualized network functions and intrusion detection systems; (2) Review of network functions virtualization tools and survey of the advantages and disadvantages of each one; (3) Execution of an *open source* intrusion detection system using network functions virtualization; (4) Elaboration and assembly of a virtualized environment for the tests. For the purposes of evaluating this dissertation, two analyzes of the virtualized functions were developed: (1) using virtual machines (simulating a complete system); (2) using containers (sharing the host system's *kernel* without needing to emulate full *hardware*).

Keywords: computer networks; intrusion detection system; loic; network functions virtualization; performance evaluatiom.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3GPP - 3rd Generation Partnership Project
Al-DDoS - Application Layer Distributed Denial of Service
API - Application Programming Interface
ARPANET - Advanced Research Projects Agency Network
CAPEX - Capital Expenditure
CPE - Customer Premises Equipment
CPU - Central Processing Unit
DDoS - Distributed Denial of Service
DMZ - Demilitarized Zone
DoS - Denial of Service
EPC - Evolved Packet Core
ETSI - European Telecommunications Standards Institute
FTP - File Transfer Protocol
GB - Gigabytes
GPRS - General Packet Radio Service
HIDS - Host Intrusion Detection System
Hosp - Hospedeiro
HOIC - High Orbit Ion Cannon
HTTP - Hypertext Transfer Protocol
ICMP - Internet Control Message Protocol
IDS - Intrusion Detection System
IDS - Intrusion Detection Systems
IETF - Internet Engineering Task Force
IP - Internet Protocol
IPS - Intrusion Prevention System
IRC - Internet Relay Chat
JSON - JavaScript Object Notation
KB - Kilobytes
LB - Load Balancer
LOIC - Low Orbit Ion Cannon
LTE - Long-Term Evolution
LXC - Linux Container
MB - Megabytes
MTU - Maximum Transmission Unit
NAT - Network Address Translation
NFV - Network function virtualization
NFVI - Network Function Virtualization Infrastructure

NIDS - Network Intrusion Detection System
NSV - Network Security Virtualization
OISF - Open Information Security Foundation
OPEX - Operational Expenditure
OSI - Open System Interconnection
QoE - Quality of Experience
S.O. - Sistema Operacional
SBC - Single Board Computer
SMB - Server Message Block
TCB - Transmission Control Block
TCP/IP - Transmission Control Protocol/Internet Protocol
UDP - User Datagram Protocol
vCPE - Virtual Customer Premises Equipment
VM - Virtual machine
VMM - Virtual Machine Managing
VPN - Virtual Private Network

SUMÁRIO

1 INTRODUÇÃO

O relatório anual da Cisco (CISCO, 2020) mostra uma previsão de que, até o final de 2023, aproximadamente 5,3 bilhões de usuários, ou seja, dois terços da população mundial, terá acesso à Internet. Esta expansão das redes de telecomunicações e o aumento no número de novos usuários e serviços, propicia, também, o crescimento das ameaças cibernéticas, tais como acessos não autorizados, roubos de informações, entre outros (JOFFER, 2019). Desta forma, torna-se necessário implementar novos meios de proteção, tais como os sistemas de detecção de intrusão (*Intrusion Detection Systems* - IDS). Segundo Tiwari *et al.* (2017), um IDS é uma função de rede utilizada para detectar acessos não autorizados e indesejados em uma rede de telecomunicação, permitindo que os gestores e administradores de rede possam tomar decisões para prover as medidas necessárias. Os sistemas IDS se subdividem em sistemas de detecção de intrusão baseados em rede (*Network Intrusion Detection System* - NID) e baseados em hospedeiro (*Host Intrusion Detection System* - HID). Os sistemas NID, normalmente, ficam posicionados em uma máquina em modo promíscuo, monitorando todo o tráfego de rede de entrada e saída. São exemplos de sistemas NID o Snort[1] e o Suricata[2]. Por outro lado, os sistemas HID são voltados ao usuário final, ou seja, protegem apenas uma máquina. Alternativamente, é possível utilizar de forma conjunta um NID e um HID (ASHOOR; GORE, 2012).

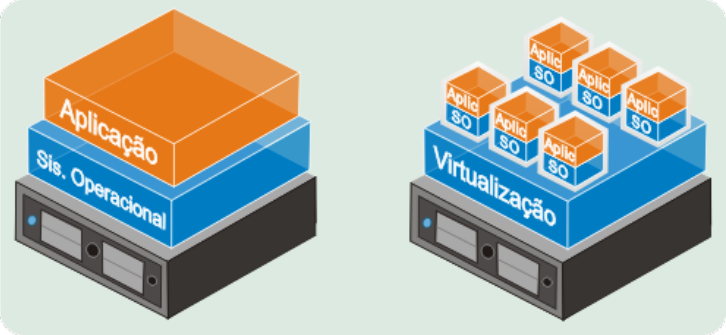
O Instituto Europeu de Padrões de Telecomunicações (ETSI, 2019) mostra que as empresas de telecomunicações precisam ser capazes de lançar novos serviços e atender de forma rápida os novos usuários. Com isso tem-se a necessidade de implementar novos mecanismos que sejam escaláveis e que tenham custo reduzido. Segundo Han *et al.* (2015), a virtualização de funções de rede (do inglês, *Network Functions Virtualization* - NFV) trata-se de uma tecnologia de rede que permite substituir dispositivos de *hardware* dedicados com custo elevado, como roteadores, *firewalls*, balanceadores de carga e *hardware* de detecção de intrusão, por funções de rede virtualizadas, que são executadas em máquinas virtuais tradicionais ou por contêineres. A Figura 1 mostra a diferença entre uma arquitetura tradicional e uma arquitetura virtualizada. No sistema tradicional o *hardware* só executa uma aplicação específica, enquanto no sistema virtualizado é possível executar diversas aplicações.

Atualmente, algumas empresas usam a virtualização de funções de rede, porém ainda existem incertezas quanto à estabilidade e escalabilidade que esta tecnologia proporciona (Rosa; Rothenberg; Szabo, 2015). A possibilidade de dimensionar dinamicamente as funções de rede de forma automatizada é uma das principais vantagens da NFV. Este processo permite a utilização mais adequada dos recursos, prevenindo-se, assim, o desperdício de recursos computacionais e investimentos em novos dispositivos de *hardware*.

A NFV permite a implantação ágil de serviços de rede, além de escalar de acordo com as demandas de novos usuários. Essa tecnologia também reduz a dependência dos clientes com

[1] https://www.snort.org/get-started
[2] https://suricata-ids.org/

Figura 1 – Comparação de arquitetura não virtualizada e arquitetura virtualizada.



Arquitetura Tradicional x Virtualização

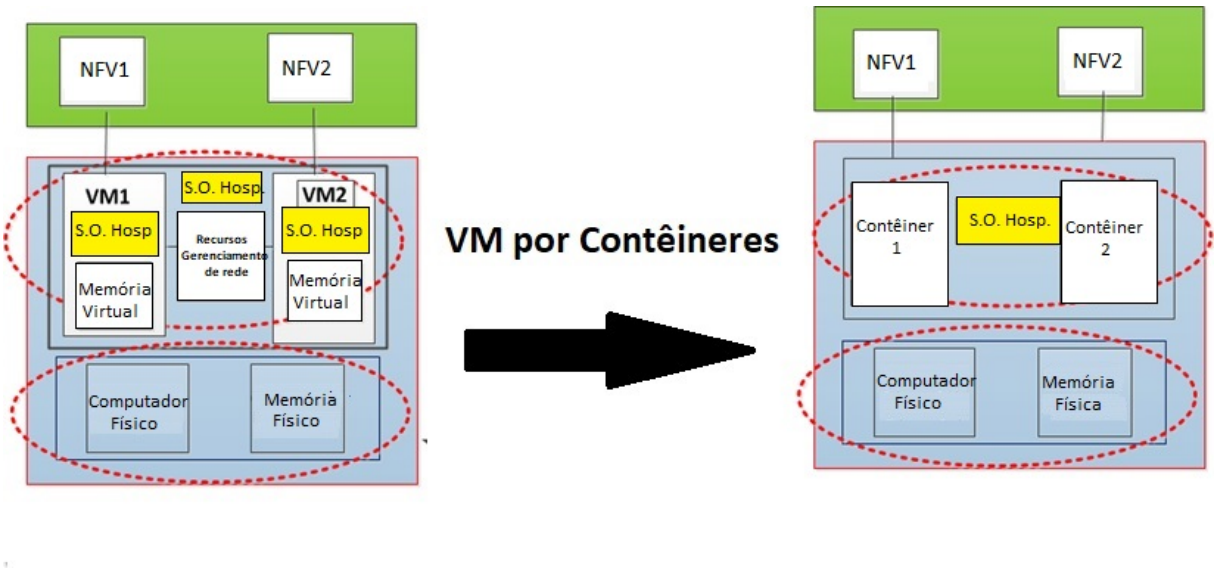
Fonte: DEVMEDIA (2019).

relação a fornecedores de *hardware*, uma vez que as NFV's são portáteis e podem executar em diferentes plataformas de *hardware*. As vantagens de utilizar a NFV incluem a capacidade de acelerar a entrega de serviços e simplificar as operações de rede, enquanto reduz as despesas com investimento de capital (*Capital Expenditure* - CAPEX) e operacional (*Operational Expenditure* - OPEX) (HAWILO *et al.*, 2014). Ao adotar esse novo conceito, os gestores de rede podem implantar e configurar prontamente serviços de rede para proporcionar soluções para diferentes perfis de clientes, oferecendo suporte aos requisitos de várias soluções. Apesar da virtualização de funções de rede estabelecer um novo paradigma (ROSA *et al.*, 2014), no processo de virtualização tradicional as instalações são demoradas, o que, em alguns casos, torna o processo de virtualização de funções de rede lento e dispendioso. Por outro lado, o Docker (DOCKER, 2013) permite criar, testar e implementar aplicações em um ambiente separado da máquina original, chamado de contêiner, o que facilita o processo de virtualização já que não há necessidade de criar e simular um *hardware* completo, uma vez que o contêiner compartilha o *kernel* do sistema hospedeiro. Uma NFV transformada em uma imagem Docker, pode ser instanciada como contêiner em qualquer ambiente desejado, ou seja, é possível utilizar a aplicação no computador do desenvolvedor da mesma forma que seria executada no servidor de produção.

Uma característica de um contêiner Docker é que ele compartilha o *hardware* e o *kernel* do sistema operacional do hospedeiro (HAT, 2020), não sendo necessário criar uma máquina virtual, um *hardware* completo, como na virtualização tradicional por *Virtual Machine* (VM). A Figura 2 ilustra a utilização de NFV através de máquinas VM e através de contêineres. Com a virtualização por VM (esquerda), a instalação de uma função de rede só pode ser realizada após a criação do *hardware* completo e da instalação do sistema operacional (SO). Por outro lado, com contêineres (direita), as funções de rede compartilham os recursos do S.O. hospedeiro (Hosp) dentro do contêiner em execução.

Assim, visando atender a grande demanda de serviços de Internet e proteger as redes de telecomunicações de ameaças cibernéticas, com sistemas escaláveis e de menor custo, faz-se necessário recorrer às novas tecnologias como a NFV, trocar *hardware* de custos elevados

Figura 2 – Utilização de NFV com Máquinas Virtuais versus Contêineres.



Fonte: Adaptado de FAISAL (2016).

com funções específicas e proprietárias por funções virtualizadas compartilhadas com custo reduzido.

1.1 Problema de Pesquisa

Porque os ataques nas redes de telecomunicação continuam crescendo em quantidade, frequência e sofisticação. Os dispositivos de *hardware* que executam funções para proteger as redes de comunicação, normalmente são proprietários e com custos elevados, demandando altos investimentos para ter a escalabilidade necessária.

1.2 Objetivo

O objetivo deste trabalho é implementar e avaliar o comportamento e o desempenho de um sistema de detecção de intrusão baseado em NFV (NFV-IDS) utilizando contêineres e ataques simulados. Faz parte do objetivo a realização de uma análise comparativa entre duas tecnologias de virtualização, VM e Contêineres, para determinar o impacto delas no consumo dos recursos computacionais, comparando processamento, memória, armazenamento e banda de rede.

Além disso, pretende-se:

1. Mostrar a viabilidade de uso do NFV-IDS avaliando a capacidade de escalabilidade e mobilidade;

2. Testar a NFV-IDS em conjunto com um servidor web Nginx[3] sob ataque.

A contribuição deste trabalho consiste em investigar como a NFV-IDS pode ajudar na competitividade para as empresas de telecomunicações e ajudar na escolha de qual tecnologia utilizar, seja virtualização utilizando VM's ou contêineres.

1.3 Justificativa

Segundo JOFFER (2019), só em 2019 houve um aumento de 180% nos ataques de negação de serviços distribuídos (*Distributed Denial of Service* — DDoS) e a previsão é que continue crescendo devido ao aumento de usuários conectados. Assim, há a necessidade de proteger-se daqueles cuja finalidade é causar danos, tanto para alimentar uma satisfação pessoal como uma tentativa de obter lucros provenientes da comercialização de informações roubadas. Um fator que ajudou a aumentar estes dados foi o crescimento do trabalho remoto *Home office* (FORTINET, 2020), que reacendeu o interesse dos cibercriminosos em ataques utilizando força bruta (JOFFER, 2019), tentativas repetidas e sistemáticas de adivinhar uma credencial enviando diferentes nomes de usuário e senhas para tentar acessar um sistema.

Neste contexto, um sistema de detecção baseado em NFV utilizando contêineres pode proporcionar uma redução nos investimentos. Segundo Lopez *et al.* (2019), a NFV veio para alavancar a tecnologia de virtualização, mesclar a utilização de dispositivos de *hardwares* distintos e diminuir os investimentos (LOPEZ *et al.*, 2019).

As funções de rede virtualizadas trouxeram a possibilidade de introduzir, de forma rápida, novos serviços direcionados e personalizados (HAN *et al.*, 2015), como um sistema de IDS, (por exemplo, Snort, Suricata). O Snort é um dos sistemas de detecção de intrusão mais conhecidos. Segundo Qayyum, Hamid e Shah (2018), um IDS opera normalmente posicionado entre a rede interna e o*firewall*, onde ele já filtra pacotes, aliviando a carga de processamento do IDS. Ainda é possível ser integrado ao *firewall*, sendo considerado uma parte importante dos sistemas de segurança de rede. O crescimento dos ataques DDoS, a necessidade de reduzir custos de investimento e as empresas de telecomunicação buscando formas de proteger-se, proporcionando motivação para analisar e comparar uma NFV implementada em um sistema de virtualização tradicional, a VM, e um sistema de virtualização por contêineres. Porém, ao lidar com essa complexidade, os operadores de rede podem ter dificuldade para solucionar falhas que venham ocorrer, mas a virtualização de funções de rede pode facilitar:

1. O aumento de desempenho conforme a demanda, evitando gargalos de um determinado serviço;

2. O incremento no desempenho da função sem necessidade de novo *hardware*, possibilitando a migração para outro *hardware* sem necessidade de refazer toda instalação;

[3] https://docs.docker.com/engine/swarm/services/

3. A análise de consumo e desempenho através de ferramentas de gerenciamento de NFV's.

Uma função de rede virtualizada pode executar diversas funções previamente gerenciada por *hardware* específico na rede, como, por exemplo: roteadores, *switches* e *firewalls*. Segundo GUIMARÃES *et al.* (2016), apesar dessas aplicações e benefícios potenciais, ainda faltam funções de rede no contexto de NFV. Apesar dos esforços da ETSI no desenvolvimento de novas funções, ainda há falta de estudos maduros, o que reduz a capacidade das operadoras de rede de tomarem decisões adequadas, podendo até dificultar a adoção de novas tecnologias (Mijumbi *et al.*, 2016).

1.4 Produção derivada desta dissertação

O seguinte artigo foi produzido a partir dos resultados desta dissertação e está em fase de avaliação:

- Oliveira, Lucas Teles de; Santi, Juliana de; Pigatto, Daniel Fernando. **An analysis of computational resource consumption of an NFV-based IDS using containers**. Computer Networks (submetido para revisão), 2023.

1.5 Estrutura do documento

Este trabalho está estruturado em cinco capítulos, sendo que neste capítulo foi apresentado a introdução e o problema de pesquisa. Ele contempla também a motivação da pesquisa e os objetivos a serem alcançados. No Capítulo 2 é apresentado a revisão literária e relatos de pesquisa referentes ao tema deste trabalho. No Capítulo 3 são apresentados a definição da proposta e os experimentos iniciais; cujos resultados irão nortear os demais experimentos no próximo capítulo. O Capítulo 4 apresenta os experimentos realizados e os resultados obtidos do direcionamento do Capítulo 3. Finalizando este documento, o Capítulo 5 apresenta algumas conclusões obtidas com esta pesquisa e sugestões para trabalhos futuros.

2 REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão da literatura sobre virtualização de funções de rede, sistemas de detecção de intrusão e ferramentas de detecção.

2.1 Início da NFV

As funções de rede tradicionais dependem de *hardware* proprietário e dispositivos de rede específicos para a função que devem desempenhar (LI; CHEN, 2015). Esta situação induz à estagnação na rede, o que limita a escalabilidade de serviços e atualizações. No final de 2012, várias das principais empresas de telecomunicações do mundo e as principais fábricas de *hardware* de telecomunicação elaboraram um relatório para ajudar na pesquisa e desenvolvimento de um trabalho colaborativo (ETSI, 2016). Em novembro de 2012, sete dessas operadoras de telecomunicação (*AT&T, BT, Deutsche Telekom, Orange, Telecom Italia, Telefonica e Verizon*) escolheram a ETSI (ETSI, 2012) para ser a casa do grupo de especificações da indústria da NFV. Somente em 2013 a virtualização de funções de rede foi apresentada oficialmente, com a publicação de cinco documentos para alinhar o entendimento do que seria NFV nas empresas de telecomunicações, além de determinar o caminho a ser seguido e uma visão geral desta nova tecnologia. Na ocasião, foram apresentados os seguintes itens:

1. Uma visão geral da infraestrutura;
2. Arquitetura atualizada de *framework*;
3. Descrições de computação, *hypervisor* e domínios de rede da infraestrutura;
4. Gestão e Orquestração;
5. Segurança e confiança;
6. Resiliência e métricas de qualidade do serviço.

Em 2014 foram lançados outros documentos como *NFV Release 2 Description*, definindo especificações técnicas e padronização da NFV (ETSI, 2016). Outras publicações posteriores definiram um conjunto de recursos-chave para tornar implantável em escala (ROSA; BERTOLDO; ROTHENBERG, 2017), garantindo a interoperabilidade das soluções NFV e compatibilidade com sistemas atuais.

A ETSI teve o papel importante no direcionamento das empresas no processo de criação de suas próprias soluções e também definiu a arquitetura básica da tecnologia, sendo: a infraestrutura de virtualização de funções de rede (*Network Function Virtualization Infrastructure* – NFVI e *Hypervisor*), a função de rede virtualizada (*Network Function Virtual* – NFV) e

o gerenciador e orquestrador de funções de rede virtualizadas. Esta parte da arquitetura básica refere-se a uma ferramenta de gerenciamento de NFV's, voltada para ambientes de *data centers*. Soluções mais simples, com poucas funções virtualizadas, podem ser gerenciadas por soluções menos robustas.

Atualmente existe uma grande comunidade de especialistas dedicados ao desenvolvimento de novos padrões e ferramentas necessárias para a NFV. Com o tempo, mais empresas aderiram e hoje o ETSI conta com mais de 900 organizações ativas em todo o mundo, provenientes de 65 países e cinco continentes (ETSI, 2020). Os membros compreendem um grupo diversificado de grandes e pequenas empresas privadas, entidades de pesquisa, academia, governo e organizações públicas. O documento *Architectural Framework* é um dos mais importantes produtos do NFV, pois fornece a base para o futuro do ecossistema NFV em toda a indústria, conforme previsto pela comunidade. O documento apresenta diferentes pontos de referência, definindo explicitamente certos blocos de construção funcionais que podem ser fornecidos por diferentes participantes do setor, abrindo caminho para novas soluções NFV multipartidárias totalmente interoperáveis (ETSI, 2016).

2.2 Virtualização

A virtualização é uma camada de *software* que implementa uma arquitetura virtual (van Cleeff; Pieters; Wieringa, 2009), a qual pode ser útil para recursos computacionais, como discos rígidos, interfaces de rede, memórias e processadores. Fornece uma interface consistente, que permite desacoplar os sistemas de *software* do *hardware* em que estão sendo executados, proporcionando mobilidade entre hospedeiros com diferentes recursos computacionais. A virtualização tem algumas camadas, como a de abstração que atua como monitor de máquina virtual (*Virtual Machine Managing* – VMM), o qual gerencia e controla as máquinas virtuais executadas sobre ele. Os principais exemplos de monitores são o *Openstack*, o *Portainer* e *Kubernetes* [1] [2], ambos de código aberto (*open source*). Uma infraestrutura virtualizada (Krishnaswamy *et al.*, 2015) é formada por várias VMMs como é possível observar na Figura 3.

Quando o assunto é virtualização, é inevitável que seja associado à ideia de vários sistemas operacionais (convidados) rodando em um mesmo computador (hospedeiro), porém este é um dos diversos tipos de virtualização e o mais comum (VMWARE, 2019). A definição de virtualização foi dada por uma das grandes empresas do mercado, uma das pioneiras quando o assunto é virtualização, conhecida como VMware[3], da seguinte maneira: "Virtualização é o processo de criar uma representação baseada em *software*, ou virtual, de alguma coisa, como aplicativos, servidores, armazenamento e redes" (VMWARE, 2019). Com o passar do tempo

[1] https://stackshare.io/stackups/kubernetes-vs-portainer
[2] https://stackshare.io/stackups/openstack-vs-portainer
[3] https://www.makeuseof.com/tag/virtualbox-vs-vmware-vs-hyper-v/

Figura 3 – Utilização de Virtualização.

Fonte: Docker Hub (2017).

outras empresas passaram a concorrer no mercado da virtualização, tais como a VirtualBox e o Hyper-V.

A virtualização de funções de rede foi recentemente proposta para melhorar a flexibilidade das redes de prestação de serviços e redução do tempo para desenvolvimento e implementação de novos serviços para o mercado. Como uma tecnologia emergente, a NFV traz diversos desafios paras as operadoras de rede tais como melhorar o desempenho atual, tornar ágil a implantação de novas funções de rede e flexibilizar a escalabilidade. A NFV traz consigo condições para mudar o paradigma que tem-se hoje na concepção e operação de redes de telecomunicação e também apresenta funções de código aberto com intuito de desacoplar a implementação de *software* de funções de rede a partir do *hardware* subjacente e fechado (HAN *et al*., 2015). O seu propósito é separar instâncias de *software* da plataforma de *hardware*, gerando desacoplamento das funcionalidades e tornando o provisionamento de serviços de rede rápido e eficiente (ETSI, 2019).

Ainda existe outro tipo de virtualização que não tem necessidade de recriar, de forma virtualizada, todo o sistema complexo, desenvolvida antes do Docker e que virtualiza somente a aplicação. Segundo Lin *et al.* (2018), o LXC (*Linux Container*) permite a criação de múltiplas instâncias isoladas de um determinado sistema operacional, dentro de um único hospedeiro, ou seja, é uma maneira de virtualizar aplicações dentro de um servidor GNU/Linux compartilhando o mesmo núcleo do *kernel* do sistema operacional.

Atualmente, existe o Docker que é uma ferramenta que tornou-se popular por ser desenvolvida utilizando a linguagem de programação Go, da Google. A Go é uma linguagem que nasceu em 2007 com três necessidades básicas: desempenho, escalabilidade e facilidade de manutenção. O Docker roda uma camada de rede acima do LXC, porém também trabalha com o mesmo conceito de contêineres. Para facilitar o entendimento desta tecnologia, precisa ser traçado um paralelo entre o que é virtual e o que é realidade. Quando segue-se esta linha de raciocínio, entende-se que algo real precisa ter características físicas, concretas, ou seja, algo palpável. Por outro lado o virtual está vinculado a algo simulado, abstrato.

2.2.1 Virtualização Tradicional

É a forma mais importante entre os tipos de virtualização e a mais utilizada, sendo conhecida como *hypervisor*. Tem como líder do mercado a solução ESX (*VSphere*) da VMware, porém existem outros grandes concorrentes como *XenServer* (*Citrix System*) e *Hyper-V*. O *hypervisor* é um sistema operacional funcional, ou seja, ele substitui integralmente o padrão de instalação de um sistema operacional, como Windows Server, Linux, Novell e etc. Segundo ETSI (2016), normalmente a utilização dos recursos físicos do *hardware* de um servidor em uso normal chega no máximo a 20%, tornando, assim, seu investimento caro e sem uso completo. Quando se utiliza uma solução de *hypervisor*, o aproveitamento dos recursos do *hardware* sai de 20% e sobe para 80% ou 90%. Isso acontece devido ao *hypervisor* compartilhar este *hardware* e prover todos os recursos para gerenciamento e funcionamento, provendo também outras soluções, como quebra da incompatibilidade de recursos no mesmo servidor, pois é possível criar "N" servidores virtuais em um único *hardware*.

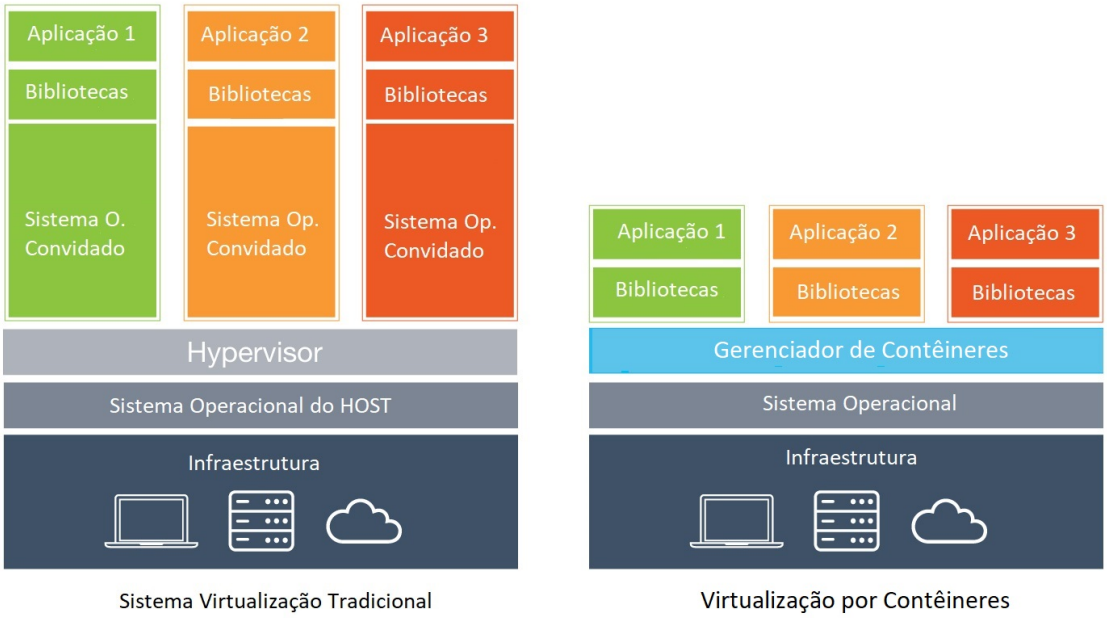
2.2.2 Virtualização por Contêineres

Este conceito nasceu ainda nos anos 90 com um sistema desenvolvido pela Centrix, porém, somente em dias atuais, esta tecnologia passou a ser utilizada em larga escala por desenvolvedores. Segundo a CANONICAL (2019), o LXC (*Linux Container*) é uma interface de espaço que compartilha o *kernel* do Linux. Por meio de uma interface de programação de aplicação (*Application Programming Interface – API*) poderosa, permite que os usuários do Linux criem e gerenciem facilmente contêineres de aplicativos ou sistemas. Basicamente, o LXC é uma ferramenta de criação de contêineres no núcleo do Linux compartilhando o mesmo *kernel*. A função do LXC é criar um ambiente o mais próximo possível de uma distribuição Linux padrão sem precisar separar *kernel* do sistema operacional hospedeiro (CANONICAL, 2019). A forma como ele fica organizado pode ser visto na Figura 4, onde mostra como está organizado um sistema de virtualização tradicional e como está organizado um sistema de virtualização por contêineres, sendo possível notar que os contêineres estão associados ao sistema operacional hospedeiro.

Já o Docker introduz um mecanismo de camadas subjacentes, ou seja, junto com uma API funcional que permite facilmente criar, gerenciar e remover as aplicações (DOCKER, 2013). Ele utiliza conceitos parecidos com o LXC, trazendo características de ser leve e gerar uma pequena sobrecarga no sistema que o hospeda, sendo possível executar diversos contêineres mesmo em dispositivos com recursos computacionais limitados, como plataformas de computador de placa única (*Single Board Computer* – SBC).

O Docker é considerado uma das formas mais atuais de virtualização, sendo desenvolvida em uma plataforma de código aberto. Segundo a Golang (2016) ele garante maior facilidade na criação e administração de ambientes isolados, ambientes de emulação ou virtualização.

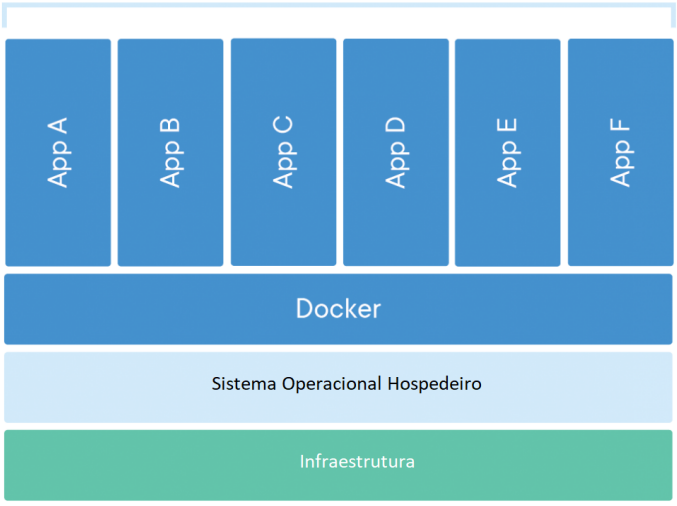
Figura 4 – Virtualização de contêineres no Linux.



Fonte: Adaptado de Docker (2013)

Desta forma, a disponibilização de novos serviços ao usuário final é ágil. A Figura 5 mostra como fica organizada a estrutura do Docker, sendo possível observar que cada função é um contêiner, compartilhando o mesmo sistema operacional.

Figura 5 – Contêineres.



Fonte: Adaptado de Docker (2013).

Uma das maiores vantagens no uso do Docker é o desempenho superior frente às máquinas virtuais com *hypervisor* (DOCKER, 2013). Enquanto nas máquinas virtuais comuns é necessária a instalação de um sistema operacional completo, com suas rotinas de bibliotecas e parte do *kernel* simuladas sobre um *hardware* artificial, o Docker compartilha essas estruturas,

quando possível, com o sistema operacional hospedeiro, resultando em um grande ganho no desempenho da virtualização e acarretando em contêineres mais leves. Enquanto as máquinas virtuais simulam o *hardware*, os contêineres simulam o sistema operacional sem necessidade de emular um *hardware* completo.

2.3 Virtualização de Funções de Rede

O modelo de referência OSI (*Open System Interconnection*), assim como o próprio modelo TCP/IP (*Transmission Control Protocol/Internet Protocol*), tem como objetivo a criação de camadas de abstração da infraestrutura de rede necessária à comunicação (Zhou *et al.*, 2010). Essa abstração mostrou-se eficaz e promoveu o crescimento das redes de computadores. No cenário atual, é comum que a informação, entre origem e destino, atravesse um grande número de elementos de rede. Além dos enlaces, estão envolvidos nesta tarefa os chamados *middleboxes*, entre os quais se destacam Roteadores, NAT (*Network Address Translation*), VPN (*Virtual Private Network*), *Proxy*, *firewall*, LB (*Load Balancer*), IDS, IPS (*Intrusion Detection System*), entre outros.

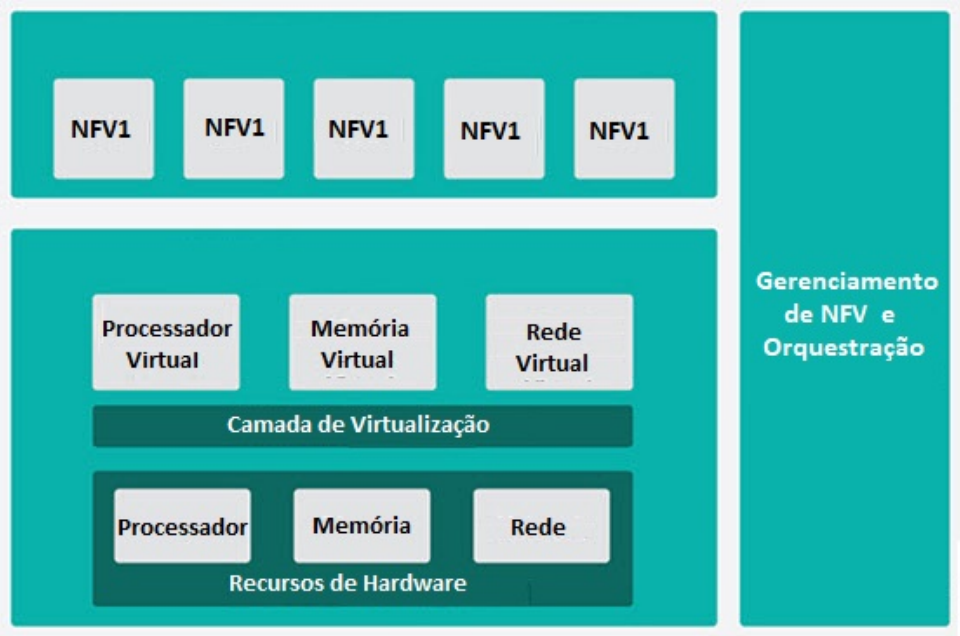
Segundo Sousa *et al.* (2007), a virtualização de redes consiste na abstração destes recursos, simplificando a tarefa de alocação dos mesmos, considerando localização física, podendo prover a separação entre recursos físicos e lógicos. O caminho percorrido pela informação em uma rede atual envolve diversos elementos além dos enlaces. Estes *middleboxes*, sob o ponto de vista da virtualização de redes, são denominados funções de rede. Essas funções de rede são implementadas por equipamentos dedicados, com *hardware* e *software* proprietários, conhecidos como *appliances*. De acordo com o ETSI (2016), além do alto custo de aquisição destes equipamentos, as tecnologias utilizadas impedem a evolução e/ou modificação para implementação de novas ideias e tecnologias experimentais, ou seja, o crescimento ou escalabilidade fica comprometida, sendo limitada pelo equipamento. Do ponto de vista do gerenciamento das redes, a tarefa ou ajuste é, na maioria dos casos, realizada somente no local físico, normalmente exigindo modificações nas conexões físicas em certas circunstâncias. Finalmente, o dimensionamento destes é realizado considerando a demanda máxima, gerando desperdícios na aquisição dos equipamentos e no consumo de energia. Quando esta demanda máxima é superada, novos equipamentos devem ser adquiridos, configurados e instalados no local, gerando um tempo de solução para o problema da demanda, que pode compreender algo entre horas ou até dias.

A NFV traz consigo condições e nova forma de trabalhar que criou um novo paradigma na concepção e operação de redes de telecomunicação, além de tornar algumas funções de código aberto para que possam ser compartilhadas e utilizadas em *hardware* de diferentes marcas. A principal tarefa deste sistema é eliminar as dependências entre o *hardware* e o *software* reduzindo a relação de dependência que os recursos de computação exercem entre si, possibilitando, por exemplo, a dissociação entre um aplicativo e o sistema operacional que ele

utiliza, como o *Office* rodando no Linux sem utilizar nenhuma ferramenta de adaptação ou emulação. Segundo Li e Chen (2015), diversos proprietários de dispositivos fixos tornam o serviço de implementação e testes cada vez mais difíceis. A NFV foi proposta como uma tecnologia essencial para beneficiar a virtualização, separando as funções da rede dos dispositivos de *hardware* subjacentes, transferindo funções de rede do *hardware* dedicado para *software* geral executando em equipamentos comerciais prontos para uso e genéricos.

Para fortalecer os estudos, sete das principais operadoras de telecomunicações do mundo se reuniram para criar grupo de estudos com objetivo de criar um padrão, aprimorar, desenvolver e buscar soluções para os problemas enfrentados pela virtualização destas funções, pois desde o início da tecnologia pesquisadores já previam sua aplicação e seus problemas com a segurança das redes (ETSI, 2020). Com a tecnologia de virtualização, novos termos nasceram e novos conceitos foram implementados como o *Network Security Virtualization* – NSV, que é voltado à segurança da virtualização de rede. Dentre os diversos tópicos de estudo da ETSI (2019), segurança é um dos que estão em destaque pelo fato de que um único servidor pode suportar duas ou mais funções, ou seja, caso a função “A” sofra algum ataque, pode acabar comprometendo o desempenho do servidor e das máquinas virtuais, além da função “B”, que está sendo virtualizada.

Figura 6 – Camadas NFV - Infraestrutura.



Fonte: Adaptado de Cao *et al.* (2015).

A infraestrutura necessária para executar uma NFV é composta por algumas camadas. Na Figura 6 é possível compreender essa organização, onde é mostrado o *hardware* hospedeiro e nesta camada ficam os recursos disponíveis como processador, memória RAM, interface de rede e memória de armazenamento. Já na camada de virtualização é onde as funções de rede são virtualizadas e instanciadas como máquinas virtuais ou contêineres. Ainda temos a

camada de orquestração coordenando o provisionamento, a escalabilidade, o monitoramento e o gerenciamento das funções de rede, as NFV's.

2.3.1 Aplicação da NFV

As aplicações práticas da NFV são inúmeras. Segundo ETSI (2019), a possibilidade de disponibilização de novos serviços com escalabilidade e a promessa de uma redução no consumo de recursos computacionais, tornam a aplicação da NFV mais atrativa. Diversos especialistas consideram esta tecnologia como uma maneira eficaz de reduzir custos com investimentos (CAPEX) e também aumentar eficiência, agilidade e escalabilidade das funções de rede (HAN *et al.*, 2015). Os campos de uso da NFV tendem a aumentar, conforme os estudos vão se aperfeiçoando (CHIOSI *et al.*, 2012). Atualmente podem ser citadas algumas aplicações, como:

- Equipamentos de comutação de rede, ou seja, servidor de acesso remoto de banda larga e roteadores, entre outros;
- Dispositivos de rede móvel, serviços de assinante doméstico, serviço de GPRS (*General Packet Radio Service*), entidades de gerenciamento móvel, gerenciamento de pacotes de dados;
- Equipamentos de análise de tráfego, ou seja, de inspeção profunda de pacotes para medição de qualidade de experiência (*Quality of Experience* - QoE);
- Dispositivos de segurança de rede, como *firewalls*, sistemas de detecção de intrusão (IDS), antivírus, proteção contra *spam*.

A NFV ainda tem alguns desafios para conseguir competir com equipamentos proprietários (HAN *et al.*, 2015), mas já é considerado um catalisador para grandes mudanças na área de redes de telecomunicações, mudando amplamente a organização das indústrias.

2.3.2 Desafios e problemas da NFV

A NFV é uma inovação que tem enorme importância para empresas prestadoras de serviço. No entanto, também enfrenta vários desafios e problemas, principalmente relacionados à estabilidade e ao desempenho para que o cliente não sinta a diferença em comparação com serviços tradicionais (LI; CHEN, 2015). A maioria das empresas de telecomunicações está acostumada a fazer mudanças durante semanas ou meses com seus dispositivos de rede tradicionais, o que pode ser feito com mais agilidade a partir do uso de NFV.

Uma NFV precisa ter o desempenho similar ao de uma função proprietária dedicada o suficiente para atender a demanda (LI; CHEN, 2015). Os *hypervisors* e as máquinas virtuais não

foram desenvolvidos para o processamento de alto desempenho e para altas velocidades de entrada e saída de informações. Ainda que um servidor possa executar uma grande quantidade de funcionalidades, sua plataforma de *hardware* e *software* deve oferecer suporte à multi-locação. As NFVs co-localizadas devem possibilitar o isolamento não apenas de segurança, mas também de recursos computacionais. Um exemplo de sistema operacional de virtualização de funções de rede com alto desempenho, segundo Li e Chen (2015), é o *ClickOS*, que considera um sistema simples, porém rápido e com suporte acima de cem máquinas virtuais. Para alcançar alto desempenho, o *ClickOS* conta com uma ampla revisão do subsistema de entrada e saída de dados do *Xen ClickOS*, consistindo em uma prova de que apenas as soluções de *software* são suficientes para aplicar o conceito da NFV.

Ainda existem desafios para a NFV ligados à nuvem. As infraestruturas de nuvem fornecem métodos para aprimorar a disponibilidade e o uso de recursos por meio da orquestração e mecanismos de gerenciamento, com aplicações e instanciação automática de dispositivos virtuais, seja para o gerenciamento de recursos ou para a reinicialização em caso de falhas, atribuindo recursos virtuais como memória, processadores e interfaces de rede. Outro desafio prático do NFV é que ele se propõe a substituir a infraestrutura existente. Segundo Han *et al.* (2015), a utilização para novos serviços ou segmentos é viável e recomendada, porém, apesar dessa proposta, a tendência é que as funções de rede atuais estarão presente por um longo período. O uso de dispositivos para gerar interfaces de rede virtuais será necessário, como por exemplo o *vswitch*, que é utilizado para conectar tráfego entre máquinas virtuais e interfaces físicas. O desafio está em como aplicar esta tecnologia e o desaclopamento do *hardware* proprietário de preço elevado.

2.3.3 Benefícios do uso do NFV

A aplicação de virtualização de funções de rede traz consigo benefícios, mas também desafios para as operadoras de rede de telecomunicações, pois com a NFV o ciclo de inovação das operadoras muda, contribuindo para mudanças neste setor. Alguns benefícios do uso de NFV, segundo a (ETSI, 2019), são:

- Custos de equipamentos e consumo de energia reduzidos através da padronização dos equipamentos;
- O NFV permite abstrair o *hardware* subjacente e permite elasticidade, escalabilidade e automação. Caso a operadora queira colocar um novo serviço, não terá necessidade de investimentos altos;
- Melhora a flexibilidade do provisionamento de serviços de rede e reduz o tempo para implantar novos serviços;

- Os investimentos necessários para cobrir as novas funcionalidades baseadas em *hardware* não são mais aplicáveis ao desenvolvimento baseado em *software*;
- Através da virtualização será possível que os operadores de rede reduzam significativamente o ciclo de maturação (tempo necessário para uma tecnologia tornar-se viável, estável e segura);
- Possibilidade de executar instalações de produção, teste e referência na mesma infraestrutura, permitindo teste e integração muito mais eficientes, reduzindo custos de desenvolvimento e tempo de colocação no mercado;
- Os serviços podem ser escalados rapidamente conforme a necessidade. Além disso, a velocidade da implantação do serviço pode ser aprimorada;
- O provisionamento de *software* pode ser feito de maneira remota sem a necessidade de visitas ao local para instalar o novo *hardware*;
- Permite uma ampla variedade de ecossistemas e incentiva o mercado de dispositivos virtuais para participantes de *software* puro e pequenos desenvolvedores, impulsionando a inovação;
- Redução no consumo de energia utilizando os recursos de gerenciamento de energia em servidores e armazenamento padrão, bem como definindo a carga de trabalho e otimização.

Existem diversos benefícios, tais como a redução na necessidade de investimentos (CAPEX), sendo este responsável pelo impulsionamento da tecnologia, tornando-a mais atrativa para empresas. Outro benefício é a possibilidade de concentrar a carga de trabalho em um número menor de servidores durante horários com baixo consumo, como à noite, permitindo que os servidores sejam desligados ou entrem e modo de economia de energia.

2.4 Sistemas de detecção de intrusão

Um IDS monitora silenciosamente a entrada e saída de dados, tendo como principal função a detecção, em tempo real, de comportamentos não autorizados dentro de uma rede ou dentro de um mesmo sistema (TIWARI *et al.*, 2017). Para chegar ao objetivo de detectar intrusão em tempo real, são necessários a coleta e o processamento de todo o tráfego dos dados que transitam pela rede, comparando-os com assinaturas ou comportamentos conhecidos, a fim de encontrar atividades maliciosas e acessos não autorizados aos recursos da rede, através de um serviço de monitoramento.

O conceito de sistemas de detecção de intrusão começou a ser notado em meados dos anos 80, em resposta aos crescentes números de acesso não autorizados nos sistemas conec-

tados as redes[4] e devido estudos importantes neste seguimento. No contexto dos sistemas de informação, o termo intrusão se referencia a indivíduos e sistemas que utilizam recursos computacionais sem autorização, assim como indivíduos legítimos, ou seja, autenticados, que abusam de seus privilégios. Os mecanismos e técnicas de ataque cibernético evoluem na mesma proporção que o desenvolvimento de novas tecnologias para a defesa desses ataques. Simon e Mitnick (2003) apontam em seu trabalho que, para que a política de segurança de uma organização esteja sempre em dia, é essencial serem estabelecidos procedimentos rotineiros que visem identificar novas ameaças e, dessa forma, combatê-las com procedimentos e ferramentas adequados. Inseridos em tal cenário, mostra-se necessário o uso de ferramentas como IDS como um meio de proteção para os ativos computacionais de indivíduos e organizações.

Um IDS é um sistema que monitora o tráfego de rede em busca de atividades suspeitas, sejam acessos não autorizados, tentativas de acessos e atividades de ataques, emitindo alertas quando essa atividade é descoberta. Embora a detecção e os relatórios de anomalias sejam a função principal, alguns sistemas de detecção de intrusões são capazes de executar ações quando atividades maliciosas ou tráfego anômalo são detectados, incluindo o bloqueio de tráfego enviado de endereços IP suspeitos. O termo IDS se refere a um mecanismo que, sigilosamente, ouve o tráfego na rede para detectar atividades anormais ou suspeitas e, deste modo, reduz os riscos de intrusão. Ao implantar uma função virtual é necessário verificar se os recursos de segurança de sua rede não serão afetados.

Segundo Han *et al.* (2015), o NFV traz consigo novos desafios para a segurança dos serviços e das máquinas. As aplicações virtualizadas podem funcionar em *data centers* que não são de propriedade direta das empresas, de maneira terceirizada, representando um risco à empresa quando da ocorrência de falhas na infraestrutura do operador. Portanto, é importante implantar novas ferramentas para ajudar no combate de ameaças, o que vai ao encontro da proposta dos sistemas de detecção de intrusão. Estes sistemas podem ser divididos em dois modelos: IDS (detecção) e IPS (prevenção) (TIWARI *et al.*, 2017).

2.4.1 IDS – *Intrusion Detection System*

Um sistema de detecção de intrusão, que pode ser baseado rede, monitora o tráfego de rede observando atividades suspeitas e alerta o sistema ou administrador da rede sobre eventuais descobertas. Em alguns casos, o IDS também pode responder ao tráfego anômalo ou malicioso, tomando medidas como o bloqueio de acesso à rede. As características básicas são:

- Atua como uma câmera ou um alarme contra as intrusões;
- Pode detectar e alertar os administradores quanto a possíveis ataques;

[4] https://www.gta.ufrj.br/grad/11$_1/sdi/historia.html$

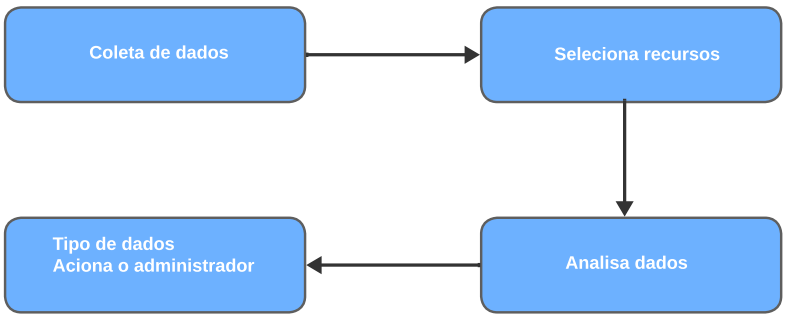
- Capaz de detectar ataques a DMZ (*Demilitarized zone*);
- O *firewall* libera conexões e o IDS detecta, notifica e responde a tráfegos suspeitos.

2.4.1.1 Sistema de Detecção de Intrusão baseado em Hospedeiro (HIDS)

Quando é utilizado um sistema de detecção de intrusão baseado em hospedeiro (ou *host*) (HIDS), normalmente instala-se em um computador ou servidor específico e monitora-se a atividade apenas deste computador. Este sistema pode ser dividido em duas partes (TIWARI *et al.*, 2017), uma baseada em assinatura e a outra baseada em anomalias. Ainda existe a possibilidade de monitorar o status do sistema operacional, monitorar os arquivos acessados e detectar um invasor quando ele cria, modifica ou exclui um arquivo. Quando ocorre uma alteração não autorizada, o HIDS ainda pode disparar um alerta, avisando o administrador responsável pela rede. Existem no mercado diversas ferramentas gratuitas *open source*, como OSSEC (*Open Source Security Event Correlator*), Wazuh, Samhain, Snort, Suricata, entre outros.

A principal diferença entre o HIDS e o NIDS, é que o HIDS pode monitorar somente o que acontece no computador local, enquanto o NIDS permite monitorar o que acontece em uma rede inteira (KLEIN, 2020). O HIDS é apenas uma das muitas ferramentas de segurança importantes que podem ajudar empresas e equipes de trabalho a melhorar a postura de segurança de suas empresas.

Figura 7 – Funcionamento IDS.



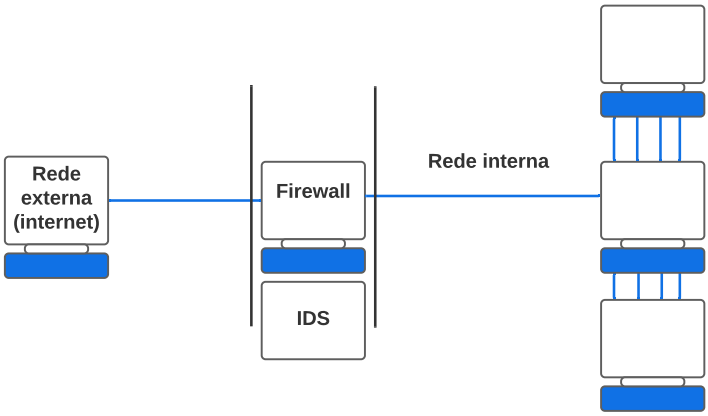
Fonte: Autor (2023).

Embora os sistemas de detecção de intrusão monitorem as redes quanto a atividades potencialmente maliciosos ou atividades não autorizados, eles também podem gerar alarmes falsos (falsos positivos). Consequentemente, é necessário que o sistema de detecção de intrusão de rede seja configurando com cuidado, na Figura 7 é apresentado como funciona IDS e o fluxo de coleta de dados até o acionamento do administrador. A necessidade de configurar adequadamente um sistema de detecção de intrusão para reconhecer a aparência do tráfego normal na rede em comparação com atividades potencialmente maliciososs requer grande atenção. Caso seja feita uma configuração adequada, esses alarmes falsos raramente ocorrerão.

2.4.1.2 Sistema de Detecção de Intrusão baseado em Rede (NIDS)

Um sistema de detecção de intrusão, quando utilizado em rede, seja ele presente em um computador ou dispositivo conectado, monitora todo o tráfego de dados que passa. Na Figura 8 é possível visualizar o IDS posicionado entre a rede interna e a rede de externa, continuamente procurando ataques e acessos indevidos da rede interna ou externa. Atualmente é comum encontrarmos o NIDS integrado aos *firewalls*, entretanto é possível ter a função de rede de forma independente. Para proteger a integridade de arquivos em uma rede, é necessário usar mecanismos de proteção como um NIDS. Segundo Li e Chen (2015), o IDS responde enviando notificações para os administradores da rede, avisando que está ocorrendo um acesso não autorizado. O NIDS procura padrões de ataque, além de utilizar coleções de itens relacionados para ajudar na definição de um padrão. Além disso, consegue especificar qual o tipo de ataque que está acontecendo naquele momento, como um ataque de negação de serviço (DDoS). Para ter acesso à rede e, principalmente, ao tráfego dela, o sistema de detecção de intrusão NIDSs é instalado próximo à saída do roteador ou *firewall*, ficando posicionado entre a rede interna e o *firewall*. Existem casos em que o IDS fica instalado junto ao *firewall* e monitora tanto a rede interna quanto a rede externa.

Figura 8 – Organização do NIDS entre uma rede externa e interna.



Fonte: Autor (2023).

Além do IDS, os sistemas de prevenção de ataques são frequentemente utilizados e podem ser associados aos IDS. A próxima seção aborda este assunto.

2.4.2 IPS – *Intrusion Prevention System*

O IPS pode ser implementado em conjunto com o IDS entre a rede externa e a rede interna (PETTERS, 2020). A diferença é que o IPS pode atuar bloqueando o atacante e eli-

minando eventuais pacotes de dados originados por ele. O IPS trabalha fornecendo políticas e normas para o tráfego de rede, juntamente com um sistema de detecção de intrusão para alertar os administradores de sistema ou de rede, quando houver tráfego suspeito.

As características básicas do IPS são:

- O IPS posiciona-se na rede podendo tomar decisões e atuar com auxílio de um *Firewall*;
- Tem a possibilidade de bloquear imediatamente a intrusão, independentemente do tipo de protocolo de transporte utilizado;
- O IPS é constituído como uma técnica de filtragem de pacotes e meios de bloqueio.

2.5 Ferramentas de detecção e prevenção de intrusão

A detecção e prevenção de intrusão (IDS/IPS) desempenham um papel crucial na segurança cibernética, ajudando a identificar atividades maliciosas e proteger sistemas e redes contra ameaças. Ferramentas populares, como Snort e Suricata, são amplamente reconhecidas por suas capacidades de monitoramento e detecção em tempo real. Existem outras ferramentas ainda que podem ser citadas de detecção e prevenção de intrusão como o Bro(Zeek), OSSEC, Samhain Labs e OpenDLP. Essas soluções utilizam regras e análises de tráfego para identificar comportamentos suspeitos e tomar medidas para proteger sistemas e redes contra intrusões. À medida que as ameaças cibernéticas evoluem, o uso de ferramentas de IDS/IPS torna-se cada vez mais essencial para manter a segurança digital.

2.5.1 Snort

O Snort (SNORT, 2019) é um sistema que atua monitorando pacotes com base em bibliotecas do tipo *libpcap*, fator que o torna um sistema de detecção de intrusão muito leve. Foi escrito em 1998 na linguagem de programação ‘C’, mas é constantemente revisado. Por ser de código aberto tornou-se conhecido mundialmente como uma das principais ferramentas de detecção e prevenção de intrusão em redes de computadores (ALBIN; Rowe, 2012). Sua principal característica consiste em analisar detalhadamente o fluxo de dados e os protocolos em tempo real. Pode ser baixado da Internet e executado em quase todas as plataformas de *hardware* e sistemas operacionais, o Snort pode ser configurado para funcionar em três modos:

1. Modo *sniffer*, que lê os pacotes da rede e os exibe para o usuário em um fluxo contínuo no *console* (tela);
2. Modo *Packet Logger*, que registra os pacotes no disco;

3. Modo NIDS, que realiza detecção e análise do tráfego da rede. Este é o modo mais complexo e configurável.

O Snort utiliza um registro baseado em regras que podem ser customizadas e alteradas conforme a necessidade. Ele é capaz de executar pesquisas, verificar correspondências de conteúdo e detectar uma variedade de outros ataques e análises, tais como sobrecarga de *buffer*, verificações de portas furtivas, ataques em sistemas de páginas web dinâmicas, análises de blocos de mensagem de servidores (*Server Message Block* – SMB) e verificação de diversas outras anomalias.

Por ser um *software* de código aberto, o Snort pode ser facilmente integrado a outros sistemas de segurança, como *firewalls*, tornando-se um sistema mais completo. O Snort possui também um sistema de alerta em tempo real, por meio do qual os alertas são enviados para um sistema de *logs* e armazenados posteriormente. Ele utiliza o padrão de *logs* criado pela IETF, o modelo *Syslogs*.

O fluxo de trabalho do Snort é composto por seis partes:

1. Captura do pacote de dados;
2. Análise do código dos dados capturados;
3. Pré-processamento do pacote;
4. Análise das regras;
5. Detecção da estrutura do pacote;
6. Registro de informações.

Para os experimentos apresentados neste trabalho a função de rede IDS (snort) estará analisando o tráfego de rede e monitorando todos os pacotes que passam pela rede.

2.5.2 Suricata

O Suricata[5] é um mecanismo de detecção de ameaças à rede gratuito e aberto, maduro, rápido e robusto. O mecanismo Suricata efetua detecção de intrusão em tempo real (IDS), prevenção de intrusão em linha (IPS), monitoramento de segurança de rede e processamento de *pcap*[6] *offline*. O Suricata inspeciona o tráfego da rede usando regras poderosas e extensas e linguagem de assinatura, além de possuir um poderoso suporte de *script* para detecção de ameaças complexas e compatibilidade com os formatos padrões de entrada e saída, como JSON (*JavaScript Object Notation*), e ferramentas como *Splunk*, *Logstash*, *Kibana* e outros bancos de

[5] https://suricata-ids.org/docs/

[6] https://www.reviversoft.com/pt/file-extensions/pcap

dados. O rápido desenvolvimento da comunidade do Suricata contribui na sua usabilidade e eficiência. O projeto e o código do Suricata pertencem à fundação de segurança de informação (*Open Information Security Foundation* – OISF)[7], uma fundação sem fins lucrativos, comprometida em garantir o desenvolvimento do Suricata e seu sucesso sustentado como um projeto de código aberto.

2.5.3 Zeek

Zeek, conhecido antigamente como Bro-IDS, é um pouco diferente do Snort e do Suricata. De certa forma, o Zeek é um IDS baseado em assinaturas e anomalias, seu mecanismo de análise converte o tráfego capturado em uma série de eventos. Um evento pode ser um login de usuário no FTP, uma conexão com um site(ZEEK, 2021). O que vêm depois do mecanismo de eventos é o *Policy Script Interpreter*. Esse mecanismo de política tem sua própria linguagem (Zeek-*Script*) e pode realizar algumas tarefas muito poderosas e versáteis. Apesar de ser uma ferramenta que possibilita diversas automações, e por ser desenvolvida inicialmente como projeto de pesquisa, dificultou sua utilização.

2.5.4 OSSEC

O OSSEC pode ser executado em quase todos os principais sistemas operacionais e inclui gerenciamento baseado em cliente/servidor com arquitetura de registros, o que é muito importante em um sistema de detecção de intrusão do tipo HIDS (KLEIN, 2020). Como o tipo HIDS pode ser comprometido, é muito importante que as informações forenses e de segurança saiam do hospedeiro e sejam armazenadas em outro lugar o mais rápido possível para evitar qualquer adulteração ou ofuscação que impeça a detecção.

2.5.5 Samhain

Samhain trabalha de forma similar ao OSSEC, tendo a mesma arquitetura cliente/servidor, entretanto não requer isso como o OSSEC. O próprio agente tem uma variedade de métodos de saída de registros logs que auxiliam no gerenciamento. Existe até a opção de usar o Samhain como um aplicativo independente em um único hospedeiro. Segundo Klein (2020) a diferença importante é onde a análise ocorre, ao contrário do OSSEC, o processamento ocorre no próprio cliente.

[7] https://oisf.net/about-us/

2.6 Técnicas de detecção e prevenção de intrusão

Uma característica das redes de computadores é que o tráfego de dados é constituído, na maioria das vezes, por pacotes IP contendo informações importantes do remetente e do destinatário. Um NIDS pode capturar pacotes de dados enquanto eles circulam na rede (LIAO *et al.*, 2013). Existem diferentes abordagens do NIDS, porém todas elas consistem em:

- Parametrização: nesta etapa, as instâncias do sistema alvo são representadas de forma pré-estabelecida.
- Estágio de treinamento: determina as características de comportamento normal e anormal, criando um modelo para ser usado como base. Esta base pode ser feita de diversas formas, pode ser manual ou automática.
- Estágio de detecção: após a disponibilidade do modelo base, este é utilizado para comparação com o tráfego de dados. Caso seja encontrado algum desvio que excede um determinado limite, um alarme será acionado conforme a categoria de processamento relacionado ao modelo de comportamento do sistema de destino.

O IDS de rede monitora o tráfego de rede de forma ininterrupta. Para que seja possível ter acesso a esse tráfego é necessário estar posicionado entre a saída e a entrada de dados (Nikolov; Kordev; Stefanova, 2018). Um sistema de detecção de intrusão pode ser classificado conforme a técnica utilizada para detectar o ataque (ASHOOR; GORE, 2012). Os sistemas de detecção de intrusão são classificados em três categorias: baseados em assinatura, em anomalia e em especificações.

2.6.1 Detecção baseada em assinatura

A utilização de um IDS baseado em assinatura ou em conhecimento utiliza assinaturas pré-determinadas para detectar o acesso não autorizado. Uma assinatura é um padrão ou uma *string* que corresponde a um ataque conhecido, geralmente armazenada em um banco de dados. Caso as características de uma detecção correspondam com alguma assinatura presente no banco de dados, um alerta é disparado conforme a configuração associada (PARK; AHN, 2017). Uma das vantagens dos sistemas baseados em assinaturas é a baixa taxa de falsos positivos, como ocorre no Snort e Suricata. A detecção baseada em assinatura mostra resultados promissores para detectar ameaças conhecidas, mas é ineficaz na detecção de novos tipos de ameaças ou intrusões, devido à dependência de assinaturas já conhecidas. Como o intuito deste trabalho é analisar o consumo dos recursos computacionais, este modelo de detecção baseado e assinatura por não gerar grandes números de falso-positivo tornou-se a opção viável para aplicação durante os experimentos.

2.6.2 Detecção baseada em anomalia

A detecção baseada em anomalia é uma técnica utilizada para detectar intrusões de rede e computador, monitorando as atividades do sistema e classificando-as em normal ou anômala (LIAO *et al.*, 2013). A classificação é baseada em regras ao invés de padrões ou assinaturas e tenta detectar qualquer acesso que sai da normalidade. A vantagem desta técnica é a possibilidade de reconhecer anomalias sem entrar em contato com suas características. Pode, então, ser considerado um modelo com várias formas de trabalho, porém é necessário achar a que melhor se aplica à necessidade.

2.6.3 Detecção baseada em especificações

A detecção baseada em especificações é um modelo mais complexo quando comparado aos demais, pois sua análise pode ser realizada em camadas, inclusive operando abaixo da camada de aplicação da pilha de protocolos da Internet (TCP/IP), no nível de controle do sistema operacional. Este tipo de detecção é responsável por monitorar os processos e combinar os dados reais com o programa. No caso de qualquer comportamento anormal será emitido um alerta avisando o responsável pela rede (ASHOOR; GORE, 2012). Uma característica importante desse modelo de detecção é a possibilidade de descobrir novos ataques.

2.7 Ataque de negação de serviço

Os ataques de negação de serviço (*Denial of Service* – DoS) têm o objetivo de prejudicar o fornecimento de um determinado serviço. Originalmente, o ataque DoS é realizado por apenas um atacante. Com o passar do tempo surgiu o ataque do tipo DDoS (*Distributed Denial of Service*), realizado por dois ou mais atacantes. Segundo Sieklik, Macfarlane e Buchanan (2016), os ataques de negação de serviço usam uma grande variedade de abordagens a fim de tornar o serviço de destino indisponível. Segundo Lin *et al.* (2019) os ataques de negação de serviço distribuído na camada de aplicativo (AL-DDoS) estão se tornando ameaças críticas para sites porque a furtividade dos ataques AL-DDoS torna muitos sistemas de prevenção de intrusão ineficazes.Ainda existem os ataques não volumétricos que exploram recursos específicos do protocolo de comunicação utilizado pela aplicação, causando esgotamento de alguns recursos da vítima por meio de um baixo volume de tráfego, como exemplo o HTTP Flood (*HyperText Transfer Protocol*) e *Slow Head*.

Um ataque DDoS sobrecarrega uma rede de modo a torná-la lenta ou inacessível, afetando a estabilidade de serviços, como páginas web e e-mails. Um ataque DDoS pode ser feito de diferentes formas: direto ou sem disfarce, podendo ser interrompido adicionando um blo-

queio no *firewall*; por meio do mascaramento de endereços IP, aproveitando o fato de que não há verificação do remetente nos comutadores da rede.

Os tipos de ataque mais conhecidos são: TCP Syn Flood, UDP (*User Datagram Protocol*) Flooding, ICMP (*Internet Control Message Protocol*) Flooding, Smurf, HTTP Flood e Slow Head.

2.7.1 TCP Syn Flood

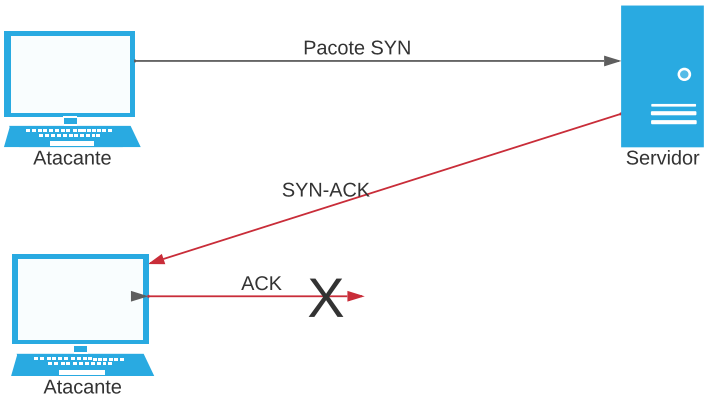
Os alvos deste tipo de ataque são, na maioria das vezes, dispositivos que utilizam o protocolo de rede TCP, um dos protocolos de rede que fica na camada de transporte. Esse ataque explora o processo de *handshake* de três vias, que é quando duas ou mais máquinas afirmam que reconheceram umas às outras e estão prontas para iniciar a comunicação (MOHAMMADI; JAVIDAN; CONTI, 2017). Neste tipo de ataque TCP SYN Flood, o atacante envia uma grande quantidade de pacotes SYN falsificados para o destino, sem realmente ter a intenção de concluir o estabelecimento da conexão. Neste modo o atacante não envia o segmento ACK para completar o *handshake*, deixando as conexões em estado de espera no servidor. Como resultado, o servidor fica sobrecarregado com um grande número de requisições pendentes e com recursos alocados. Atualmente não é fácil fazer um ataque do tipo SYN Flood, pois o SYN COOKIES um mecanismo de defesa criado para mitigá-lo já vêm habilitado no kernel do Linux. Entretanto, é importante reconhecer que essas medidas de segurança podem não ser infalíveis. Em algumas situações, os atacantes podem encontrar maneiras de contornar essas defesas ou explorar vulnerabilidades que ainda não foram abordadas. Nesses casos, a detecção por meio de um sistema de Detecção de Intrusão (IDS) pode desempenhar um papel complementar e extremamente importante

Durante o processo de estabelecimento de uma conexão é necessário efetuar uma requisição, ação explorada pelo ataque TCP SYN Flood. O cliente envia um pacote do tipo SYN para o servidor, solicitando uma conexão. Ao receber esse pacote, o servidor responde com o SYN-ACK, uma confirmação de recebimento. O servidor reserva um espaço na memória através da estrutura de dados *Transmission Control Block* – (TCB) com o estado SYN-RECEIVED desta conexão, indicando se a conexão está aberta. Quando o cliente recebe o SYN-ACK, ele envia o pacote ACK para o servidor, que, recebendo-o, abre a conexão e muda o TCB para o estado de conexão estabelecida (*estabilished*). Este processo é ilustrado na Figura 9.

2.7.2 UDP Flood

O UDP é um protocolo que opera sem estabelecimento de conexão antes da transmissão de dados entre o remetente e destinatário, ou seja, ele não valida a conexão e nem confirma envio ou recebimento de pacote de dados. O UDP não detecta perda de pacotes du-

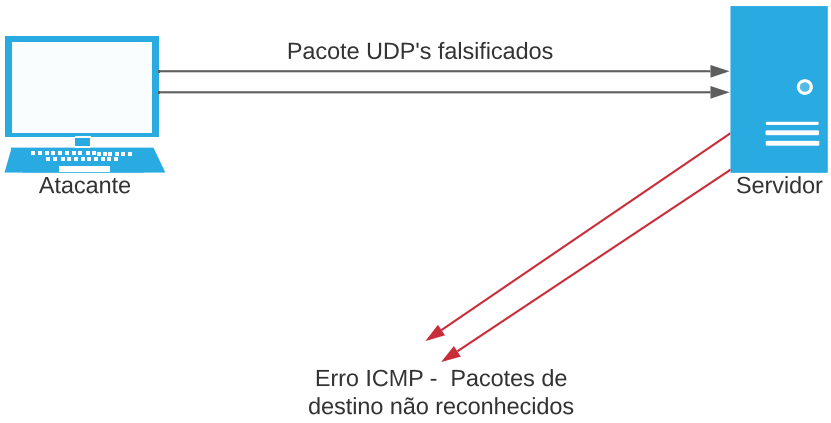
Figura 9 – TCP Syn Flood.



Fonte: Autor (2020).

rante a transmissão de dados e não envia qualquer mensagem de erro (YUSOF; ALI; DARUS, 2017). Por essas características, o UDP é um protocolo rápido, principalmente se comparado ao TCP. Entretanto, pacotes UDP podem ser explorados por invasores para lançar ataques de inundação UDP. Na Figura 10, o atacante envia um determinado número de pacotes UDP-Flood nas portas de destino do alvo, porém o alvo não reconhece a origem do ataque.

Figura 10 – UDP Flood.



Fonte: Autor (2020).

2.7.3 ICMP Flood

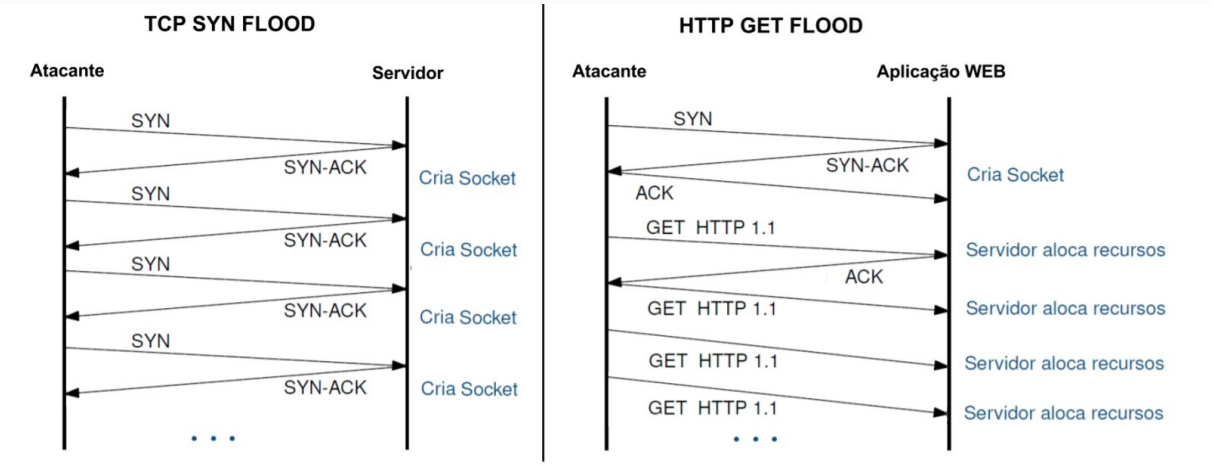
O tamanho máximo de um pacote IP é 65.535 bytes, incluindo os cabeçalhos (YUSOF; ALI; DARUS, 2017). Os sistemas de computadores não lidam com um pacote de *ping* maior que o tamanho máximo de um pacote. No ataque Ping da Morte, os atacantes enviam pacotes quebrados em fragmentos, os quais serão remontados pelo sistema de destino. Como os pacotes

estão superdimensionados, o sistema de destino terá problemas com uso de memória, ocasionando um *overflow* e, com isso, gerando problemas ao sistema alvo, tais como travamento ou lentidão. O ICMP *Flooding* pode ser considerado um ataque eficaz, devido à capacidade de falsificar os dados do atacante. Além disso, o invasor não precisará de detalhes do computador da vítima, exceto seu endereço IP.

2.7.4 HTTP Flood

Um ataque deste tipo explora as requisições do HTTP (*Hypertext Transfer Protocol*), principalmente as requisições GET e POST para atacar um servidor web. Através das requisições HTTP Get é possível obter conteúdos estáticos, como textos e imagens, do servidor que sofre o ataque, enquanto que as requisições HTTP Post são usadas para obter recursos gerados dinamicamente (UFRJ, 2015). O ataque do tipo HTTP GET junta vários computadores, que enviam várias requisições de arquivos ou dados importantes dentro de um alvo, sempre de forma coordenada. Quando o alvo é inundado com solicitações e respostas recebidas, ocorre a negação de serviço para solicitações adicionais de fontes de tráfego legítimas conforme ilustrados na Figura 12, a qual compara com o fluxo de requisições do tipo TCP Flood e HTTP Flood. Normalmente são mais simples e mais escaláveis quando distribuídos. O objetivo de um ataque do tipo HTTP é enviar várias solicitações de processamento de conteúdo para o servidor, de forma a esgotar os recursos computacionais do alvo. Deste modo, as requisições HTTP Post tendem a ser mais eficientes, visto que exigem processamento mais complexo por parte do servidor. Esse ataque utiliza a disparidade no consumo relativo de recursos, enviando muitas solicitações post diretamente para um servidor visado até que sua capacidade esteja saturada e ocorra a negação de serviço (CLOUDFLARE, 2021).

Figura 11 – Comparação entre TCP Flood e HTTP Flood.



Fonte: Dados Adaptados de Wang e Ng (2010).

2.7.5 Smurf

Um ataque do tipo Smurf acontece quando uma grande quantidade de pacotes ICMP é enviada para o computador alvo, sendo inundado com mensagens de *ping* falsas. Um ataque Smurf bem sucedido requer conhecimento e organização (YUSOF; ALI; DARUS, 2017).Um ataque Smurf segue algumas etapas, a primeira é quando um malware cria um pacote de rede anexado a um endereço IP falso, técnica conhecida como "spoofing", ou falsificação. Dentro deste pacote, há uma mensagem de ping por ICMP que por sua vez solicita aos nós de rede que receberam o pacote enviem uma resposta de volta. Essas respostas são retornadas a endereços IP da rede, entrando em um loop infinito. Quando combinado com uma transmissão de IP, que envia o pacote malicioso para todos os endereços IP na rede, o ataque Smurf é capaz de causar rapidamente uma negação de serviço completa. O impacto de um ataque Smurf de sucesso pode durar horas ou dias, ocasionando perda de receita, frustração do alvo e roubo de arquivos. Muitos ataques Smurf podem estar agrupados com *rootkit*, que permite aos invasores criarem uma *backdoor* para facilitar o acesso ao sistema, podendo derrubar um servidor ou site de uma empresa.

2.7.6 Ferramentas que simulam ataques DDoS

Os ataques de negação de serviço por inundação visam consumir o recurso de processamento, memória, banda de rede passante, sendo enviados uma enorme quantidade de pacotes por segundo para consumir estes recursos ou até mesmo deixar suas respostas lentas. Outra característica de algumas ferramentas como o LOIC é que possibilitam não usar a falsificação de IP, porque os ataques à camada de aplicação precisam primeiro estabelecer uma conexão TCP e a falsificação do IP de origem não conseguiria estabelecer uma conexão. Atualmente existem diversas ferramentas para realização de ataques DDoS.

2.7.6.1 Hping3

A ferramenta hping3 é uma ferramenta amplamente utilizada no meio acadêmico para a realização de experimentos, pois envia pacotes de dados manipulados (SU *et al.*, 2016). Esta ferramenta permite controlar o tamanho, a quantidade e a fragmentação dos pacotes para sobrecarregar o alvo. O Hping3 pode ser utilizado para fins de segurança ou para testes de capacidade, como por exemplo, para testar a eficácia de *firewall* ou um sistema de proteção como o IDS.

2.7.6.2 Low Ion Orbital Cannon (LOIC)

O LOIC[8] foi uma das principais ferramentas utilizadas pelo grupo de ativistas Anonymous nos ataques de DDoS que são realizados desde 2010. O Anonymous é um grupo de ativistas que coordena ataque formado por milhares de pessoas voluntárias no mundo. Deste modo o LOIC é utilizado por atacantes voluntários e, portanto, não possui nenhum mecanismo para invadir e contaminar máquinas para fazer um exército de *bots* e também não precisa esconder seu funcionamento. O ataque é formado então por um exército de voluntários que executam o código de ataque a vítima, coordenados por um canal IRC. É uma ferramenta que permite realizar inundação de HTTP, enviando uma abundante quantidade de requisições HTTP, UDP e TCP (LOIC, 2017). A característica fundamental do LOIC é que foi projetado para ter um uso muito simples, e ainda possui uma interface gráfica. Uma característica desta ferramenta é que utiliza pouca banda de rede.

2.7.6.3 Slowloris

O objetivo desta ferramenta também é derrubar um serviço web através da manutenção de conexões ativas ao enviar diversas requisições. Slowloris é um programa de ataque de negação de serviço que permite que um invasor sobrecarregue um servidor alvo, abrindo e mantendo muitas conexões HTTP simultâneas entre o invasor e o alvo (CLOUDFLARE, 2020). Este procedimento mantém os sockets abertos no servidor web e depois de um tempo de ataque, todos os sockets do servidor ficam ocupados com as requisições de Slowris, o que impede que ele atenda as requisições de usuários legítimos. Diferentemente dos ataques por inundação de pacotes IP, esse tipo de ataque usa uma baixa quantidade de largura de banda e, em vez disso, visa usar os recursos do servidor com solicitações que parecem mais lentas que o normal, mas que imitam o tráfego normal. Uma característica do Slowris é que durante o ataque, o arquivo de logs do servidor web não criará entradas correspondentes a erros, porque nenhuma sessão HTTP é concluída, mas ao finalizar o ataque, muitas entradas de erros 400 serão criadas no log.

[8] https://www.imperva.com/learn/ddos/low-orbit-ion-cannon/

3 TRABALHOS RELACIONADOS

O artigo de Cao *et al.* (2015) destaca que as características da NFV são essenciais para a otimização de um ambiente de rede. Ele propõe a utilização de uma biblioteca chamada de NFV-Vital para gerenciar e implementar uma função de rede virtualizada, permitindo, ainda, analisar os recursos computacionais utilizando duas funções de rede para a detecção de intrusão: o Snort e o Suricata. Durante a realização dos testes (CAO *et al.*, 2015), o NFV-VITAL permitiu determinar automaticamente as configurações ideais de recursos computacionais para cada um dos sistemas de detecção de intrusão com diferentes cargas de trabalho. Entretanto, este trabalho limitou-se em analisar e gerenciar os recursos computacionais disponíveis na VM e comparar o Snort e o Suricata, que teve um consumo um pouco maior de recursos computacional.

Os autores Qayyum, Hamid e Shah (2018) apresentaram uma solução para reduzir a perda de pacotes de dados quando o tráfego da rede é muito grande. As organizações gastam milhões de dólares todos os anos, com o intuito de proteger suas redes de computadores e manter seus dados seguros (Qayyum; Hamid; Shah, 2018). Existem empresas que fornecem o Snort integrado a um *firewall* que, por sua vez, está integrado a um *hardware*, entretanto o seu custo é elevado e com limitação de tráfego de dados. Desta forma, para aumentar o tráfego de dados é necessário adquirir um novo equipamento ou uma nova licença. Para resolver este problema, Qayyum, Hamid e Shah (2018) propuseram aplicar o Snort baseado em NFV sem integração com um *firewall* para atender esta demanda e reduzir o custo do investimento em *hardware*. Os resultados mostraram que a perda de pacotes pode ser controlada.

Durante o estudo de Brumen e Legvart (2016) foi realizada uma comparação entre dois IDS's: o Snort e o Suricata. Diversos experimentos foram realizados permitindo avaliar o consumo dos recursos computacionais e a capacidade de detecção de cada IDS. Os testes foram realizados tanto no sistema operacional Linux quanto no Windows, onde foram simulados diversos ataques utilizando os dois sistemas operacionais. Os resultados mostraram que o Suricata teve um consumo de processamento e de memória RAM[1] maior que o consumo do Snort, independentemente do sistema operacional. Entretanto, a porcentagem de pacotes perdidos foi menor durante cinco de seis ataques simulados. O resultado mostrou que a solução baseada em Linux consome mais recursos computacionais, porém a solução sendo executada no Windows teve uma taxa maior de perda de pacotes. A conclusão dos autores é que os sistemas operacionais utilizados nos testes não são os ideais para grandes volumes de tráfego quando executados em um único servidor. Com base nestes resultados, torna-se essencial a análise do comportamento de NFV sendo executado no mesmo servidor Web como também mostra quais caminhos não seguir, quais sistemas operacionais utilizar e o que analisar durante a realização dos experimentos que serão realizados para este trabalho.

[1] https://www.processtec.com.br/artigos/o-que-e-memoria-ram-qual-a-funcao

O estudo de Fadhlillah, Karna e Irawan (2021), realiza testes de ataque com as ferramentas LOIC, Torshammer e Xerxes, em um cenário de teste com IDS e sem IDS. Além disto foram implementados um servidor Web e FTP para a realização dos experimentos. A partir dos resultados dos experimentos realizados, o IDS detectou com sucesso os ataques recebidos. É importante salientar que o objetivo era testar a eficiência de uma ferramenta IDS e analisar o consumo dos recursos disponíveis, como processamento, memória e a quantidade de pacotes detectados pelos IDS. Este estudo auxiliou na escolha dos recursos computacionais mensurados, como ainda possibilitou nortear alguns experimentos, como o caso do servidor Web.

3.1 Casos de uso do NFV

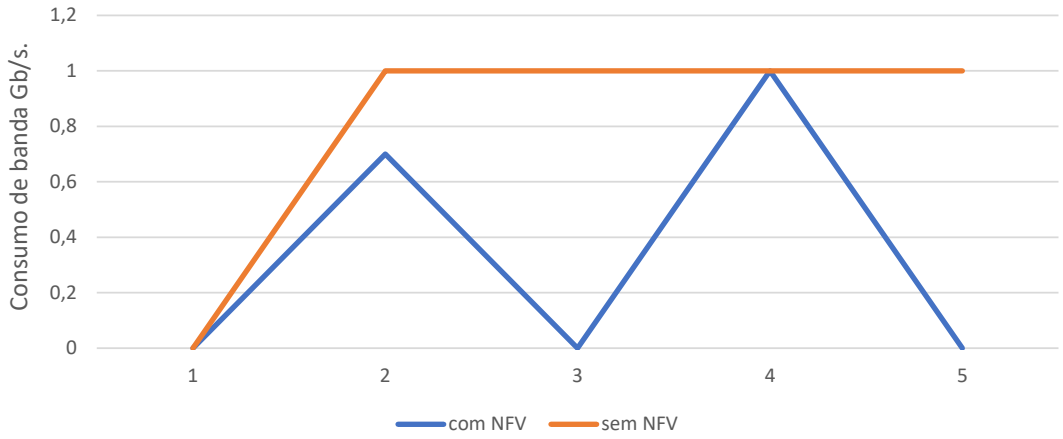
Em Basta *et al.* (2014), um caso de uso foi realizado com a função de rede virtualizada *CORE*, que são roteadores ou *switches* especialistas em uma função específica de controlar o encaminhamento de dados, determinando o caminho que o pacote deverá seguir. O NFV mostrou-se estável e apresentou flexibilidade e mobilidade, além de reduzir o consumo de energia.

Outro caso de uso está relacionado a uma função que serve como estrutura para fornecer voz e dados convergidos em uma rede 4G (*Long-Term Evolution* – LTE), o *Evolved Packet Core* (EPC). Caso o EPC apresente problemas e não esteja disponível, a operadora normalmente faria a substituição do equipamento existente. Segundo Jain *et al.* (2016), a maioria dos componentes do EPC tem toda a lógica do plano de controle e dados programados no *hardware*. Quaisquer modificações na funcionalidade para adicionar novos serviços requerem a substituição de todo o aparelho. Nesse caso de uso, a implementação do EPC baseada em NFV se torna interessante para academia e indústria. Com NFV, o EPC teve potencial para resolver problemas de escalabilidade de forma rápida.

Wang e Ng (2010) executaram testes e efetuaram medições de desempenho em uma rede de ponta a ponta utilizando o serviço de nuvem da Amazon EC2. Instabilidades foram detectadas durante o teste. Na Figura 12, é possível ver a comparação do processamento compartilhado. A transferência de dados foi instável, com taxa de transmissão de dados variando de zero a 1 Gb/s.

Ainda pode ser mencionado o caso de uso desenvolvido pela Intel em colaboração com a *British Telecom*, onde foi proposta uma solução para substituir o equipamento que fica dentro das instalações do cliente *Customer Premises Equipment* (CPE) por uma solução virtualizada. Este é considerado um dos principais casos de uso, pois o CPE é um equipamento utilizados em larga escala, que fica localizado na ponta do acesso do usuário de um serviço de telecomunicação (INTEL, 2016). Em 2016, a Intel e a British Telecom concluíram esse estudo mostrando os benefícios de utilizar o *virtual Customer Premises Equipment* (vCPE) em clientes corporativos. A experiência moveu as funções do dispositivo para a nuvem da operadora ou para servidores localizados no cliente.

Figura 12 – Comparação entre NFV e sistema tradicional.



Fonte: Dados Adaptados de Wang e Ng (2010).

Segundo a Intel (2016), os números apresentados revelaram grandes vantagens para o modelo virtualizado, atingindo redução de 32% a 39% no custo de instalação dos vCPEs e até 90% de redução do custo de aquisição de *hardware* para grandes clientes. A economia potencial projetada para um período de cinco anos, é de aproximadamente 259 milhões de dólares. Para pequenos e médios clientes corporativos, a economia foi de 33% quando usando hospedagem na nuvem, e um pouco menor quando instalado em servidores do cliente.

4 DESENVOLVIMENTO E EXPERIMENTOS INICIAIS

Este capítulo apresenta o desenvolvimento do trabalho e os experimentos preliminares. Inicialmente, foi realizada uma revisão da literatura considerando estudos de caso com NFV seção 3.1, estudos práticos de utilização da NFV associada a IDS e estudos que analisam e comparam sistemas de detecção de intrusão Capítulo 3, como também o comportamento dentro de um ambiente de infraestrutura de rede.

O modelo de NFV-IDS proposto nesta dissertação de mestrado é baseado em NFV's implementadas em contêineres no Docker. Tal modelo será comparado inicialmente com o modelo de IDS baseado em VM tradicional, considerando as seguintes métricas relacionadas ao consumo de recursos computacionais: processamento, memória, largura de banda de rede e armazenamento de dados. Através da análise das medidas realizadas, busca-se verificar a eficiência no consumo dos recursos computacionais, destacando as características do modelo de virtualização proposto que geram impactos nestas métricas e, assim, apontando suas vantagens e desvantagens, e direcionando os experimentos finais deste trabalho. Além de analisar o consumo dos recursos computacionais, será analisada também a escalabilidade da função de rede.

Para analisar o desempenho da função de rede IDS baseada em NFV, utilizou-se contêineres em Docker para implementação de dois cenários de testes preliminares: um cenário local (seção 4.3) e um cenário distribuído (seção 4.4). Estes experimentos tinham por objetivo uma análise prévia de fatores relevantes a serem considerados no trabalho. A composição dos dispositivos de *hardware* usados nos testes é descrita na Tabela 1.

Tabela 1 – Detalhes de hardware.

Identificação do nó	Versão do SO	Processador	Armazenamento
Computador com Docker	Ubuntu 18.04.5 LTS	I5 8250u	SSD
Atacante 1	Windows 10	i5 7500	HD
Atacante 2	Windows 10	i5 7500	HD
Atacante 3	Windows 10	i5 7500	HD

Fonte: Autor (2020).

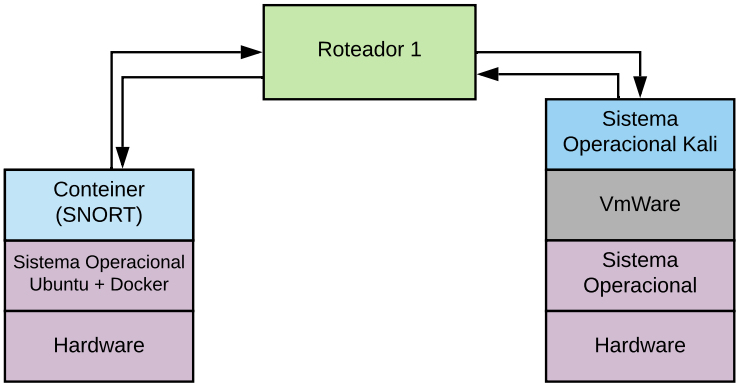
Para todos os cenários desenvolvidos, a instalação do Docker foi realizada no Linux utilizando a distribuição Ubuntu 18.04.5 LTS[1]. Para auxiliar no gerenciamento e orquestração dos contêineres (imagens, interfaces de rede e consumo de recursos computacionais), foi utilizado o *Portainer*[2]. Para realização dos ataques DDoS iniciais foi utilizada a ferramenta Hping3[3], posteriormente substituída pela ferramenta LOIC [4] devido à possibilidade de realizar ataques do tipo HTTP.

[1] https://ubuntu.com/
[2] https://www.portainer.io/
[3] https://tools.kali.org/information-gathering/hping3
[4] https://www.imperva.com/learn/ddos/low-orbit-ion-cannon/

4.1 Cenário de ajustes

Nesses experimentos o intuito foi identificar as dificuldades e os problemas relacionados ao funcionamento da NFV-IDS executada em um ambiente utilizando contêineres e, assim, nortear o andamento dos demais testes. Neste cenário utilizou-se uma conexão de rede sem fio e foi implementada uma rede local conforme mostrado na Figura 13. No lado esquerdo da figura é apresentado o *hardware*, o sistema operacional Linux com Docker e o contêiner executando SNORT. No lado direito da figura pode ser visto o *hardware* e o sistema operacional utilizado para a realização dos ataques.

Figura 13 – Topologia de rede usada no primeiro teste.



Fonte: Autor (2019).

O Snort foi escolhido para os testes por ser um sistema de detecção de intrusão devido a sua história e maturidade, onde conta com uma grande comunidade de usuários e suporte comercial sólido. Para não ser necessário criar todas as regras manualmente no NFV-IDS, foi utilizada a opção disponível gratuitamente, comunitária, com uma regra genérica para possibilitar o teste. A regra utilizada no experimento foi: **alert icmp any any -> any any (msg>"Teste Ping..."; sid:1000004;)**. Esta regra permitiu monitorar o protocolo ICMP, utilizado para os testes de comunicação, seja para entrada ou saída de informações.

4.2 Procedimento atacante

Para simular o atacante na rede nestes experimentos, foi instalado o sistema operacional Kali Linux 2020.1 em uma máquina virtual VMware (que simula um *hardware* completo). O ataque DDoS foi realizado através do usuário *root* e da ferramenta Hping3, que permite enviar pacotes com tamanhos variados. A Tabela 2 apresenta os argumentos do Hping3 e suas respectivas funções.

O Hping3 é uma ferramenta de rede versátil que oferece uma variedade de recursos para a geração e manipulação de pacotes de rede. Com o Hping3 foi possível criar cenários de

Tabela 2 – Tabela de argumentos do Hping3.

Argumentos	Resultado
Sudo	Fornece os privilégios de root
Hping3	Chama o programa hping3
-S:	Especifica pacotes SYN
-Flood	Respostas serão ignoradas e desempenho melhorado
-V	Verbosidade (expressão do ataque)
-c	Quantidade de pacotes
-d	Tamanho dos dados
-p 80:	Porta que será atacada
192.168.5.100	Endereço de IP que irá receber o ataque

Fonte: Autor (2020).

teste personalizados. Por ser uma ferramenta que já vem junto ao Kali Linux e ter a capacidade de montar e modificar pacotes de forma eficiente e simples foi valioso para os fins de teste e diagnóstico de rede(PATIL; DUDEJA; MODI, 2019). O Hping3 facilitou os experimentos iniciais e, entretanto, ao longo da pesquisa, verificou-se a necessidade de utilizar ferramentas mais atuais como o LOIC e HOIC [5]. Para realizar os ataques utilizando o Hping3, foram utilizados argumentos com tamanhos de pacotes limitados pela MTU (*Maximum Transmission Unit*), ou seja, 1500 bytes.

4.2.1 Resultado preliminar

Para analisar preliminar o ataque DDoS sobre NFV-IDS, foram consideradas as seguintes métricas: o consumo de processamento, o consumo de banda de rede, da memória principal e da memória de armazenamento. Algumas características descritas por Mijumbi *et al.* (2016) foram identificadas, como a mobilidade do contêiner rodando no Docker, a possibilidade de atualização do contêiner, a configuração a qualquer momento mesmo estando em execução e a criação de novas imagens com diferentes recursos ocupando pouco espaço da memória de armazenamento. Outra característica notada foi a capacidade de escalar em função da demanda.

Durante as execuções, não observou-se gargalo ou sobrecarga dos recursos computacionais devido à baixa carga do ataque. Para testar ataques DDoS foi necessário utilizar amplificação, ou seja, multiplicar o tamanho do ataque, o que é feito por técnicas tais como o uso de computadores zumbi. Segundo Colella e Colombini (2014), os ataques de amplificação podem usar máquinas legítimas e não infectadas, que sofrem de falhas de segurança. Como os ataques realizados neste cenário foram elaborados em um ambiente controlado (KUHRER *et al.*, 2014), foi necessário amplificar a quantidade de requisições, no HPing3 foi possível através de parâmetros, como o de quantidade de requisições. Somente com esta amplificação torna-se

[5] https://www.wallarm.com/what/what-is-high-orbit-ion-cannon-hoic

possível analisar o consumo dos recursos computacionais e determinar o tamanho dos pacotes de dados a ser utilizado nos experimentos com maior quantidade de amostras[6].

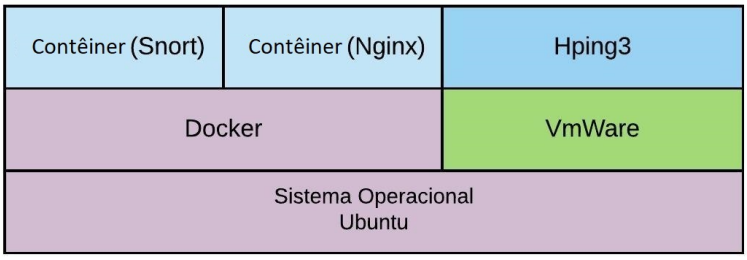
4.3 Cenário local

Neste cenário, além do atacante, dois contêineres foram implementados e executados no mesmo hospedeiro, como mostra a Figura 14. Cada contêiner executou uma função específica:

- Primeiro Contêiner: Snort (Função virtualizada de detecção de intrusão);
- Segundo Contêiner: Nginx (Servidor Web direcionado na porta 80).

É importante notar que, mesmo ocorrendo o compartilhamento do hospedeiro, as análises de consumo de processamento de memória são independentes, sendo possível devido à transparência proporcionada pela implementação através de contêineres. O contêiner do Nginx (servidor web) foi implementado para realizar a função de servidor *web* que recebeu o ataque. Por sua vez, o contêiner Snort atuou no monitoramento da porta de rede de entrada do servidor Nginx. Outra característica que deve ser observada é que neste cenário não foi possível ter informações do consumo de banda de rede da porta *ethernet* física, visto que neste cenário ele utilizou somente as interfaces de rede virtualizadas.

Figura 14 – Infraestrutura utilizada para cenário local.



Fonte: Autor (2020).

Os parâmetros utilizados pelo Hping3 (Figura 15) descritos no cenário de ajustes (seção 4.1).

4.4 Cenário distribuído

Neste cenário, as características dos primeiros testes são replicadas. A mudança ocorreu nos atacantes, executados em *hardwares* separados (Linux com Hping3), e os contêineres com as NFV's (Snort e Nginx) foram implementados e executados no mesmo hospedeiro, como mostra a Figura 16.

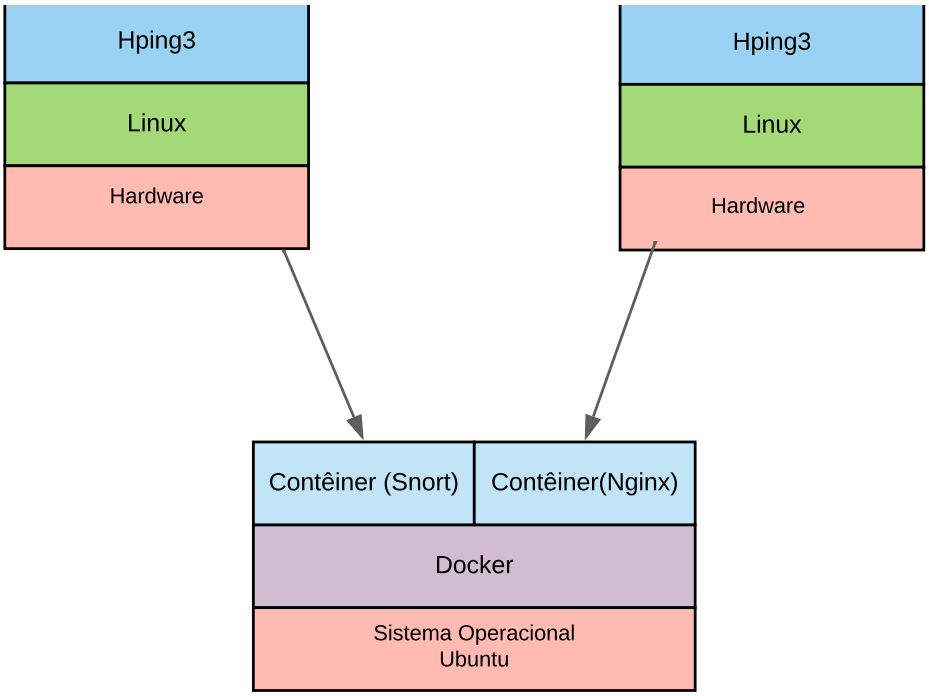
[6] https://cbltech.com.br/blog/o-que-e-ataque-ddos.html

Figura 15 – Regras de ataque hping3.

```
root@kali:/# sudo hping3 -V -c 1000000 -d 1200 -S --flood --rand-source 192
.168.5.110
using eth0, addr: 192.168.5.109, MTU: 1500
HPING 192.168.5.110 (eth0 192.168.5.110): S set, 40 headers + 1200 data byt
es
hping in flood mode, no replies will be shown
```

Fonte: Autor (2019).

Figura 16 – Infraestrutura utilizada no cenário distribuído.



Fonte: Autor (2019).

Para analisar o comportamento dos recursos computacionais durante um ataque distribuído, os ataques foram executados 5 vezes utilizando o ataque do tipo TCP *Syn Flood* com duração de 5 minutos cada experimento. Também levou-se em consideração o conceito de amplificação de um ataque DDoS (CERT, 2012). O tamanho dos pacotes utilizados durante o ataque distribuído foi o mesmo definido no cenário local (Seção 4.3). O objetivo desse teste foi identificar qual tamanho de pacote de dados gerava gargalo durante o ataque distribuído.

De acordo com a Tabela 3, o consumo de processamento já chega próximo dos 100% durante o ataque com tamanho de 1500KB. Os dados de consumo dos recursos computacionais foram extraídos através do sistema operacional, o consumo de Banda de rede refere-se ao total durante a execução do experimento. No ataque, utilizando o pacote de dados com tamanho de 1500KB, o consumo do recurso de processamento ficou em 31%. Esse consumo ocorreu devido ao tamanho de pacote de dados com amplificação ser de 10 (dez) vezes, considerado um valor baixo quando comparado a um ataque real, que em diversos casos é amplificado milhares de vezes (CERT, 2012).

Tabela 3 – Consumo de recursos de *hardware* para diferentes tamanhos de pacote em um ataque distribuído.

Testes	15KB	150KB	1500KB	3000KB	6000KB
Consumo de Processamento (%)	31%	53%	98%	100%	100%
Consumo de memória (MB)	290,1	351,1	351,5	351,6	351,8
Consumo de Armazenamento (GB)	0,68	0,68	0,68	0,68	0,68
Consumo de Banda (GB)	4,1	13,2	35,4	114	141
Tempo (min)	5	5	5	5	5

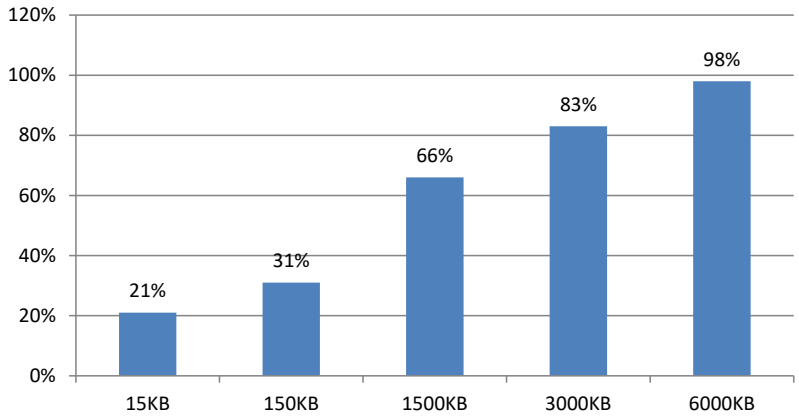
Fonte: Autor (2020).

No Hospedeiro, onde estavam as funções de rede virtualizadas (Snort e Nginx), o consumo de processamento durante o ataque DDoS com tamanho do pacote de 1500KB foi de 98%. Nos testes com pacotes de tamanho 3000KB e 6000KB, o consumo ficou em 100% e 101%, respectivamente, o que indica a existência de gargalo de processamento. Isto deve-se ao ataque do tipo distribuído e ao tamanho do pacote de dados utilizado.

4.5 Funcionamento do Snort durante uma execução preliminar

O funcionamento do Snort foi monitorado e gerenciado por uma ferramenta executada em um contêiner, o *Portainer*. Assim, foi possível monitorar o consumo de processamento, memória e banda de rede dos contêineres que estavam em execução. O Snort foi executado no Docker para identificar o comportamento da função de rede virtualizada e, com isso, identificar se houve gargalo de processamento, durante os ataques. A Figura 17 mostra o consumo de processamento do primeiro ataque, onde o Snort detectou o ataque, deste modo permitiu a análise do consumo do recurso computacional.

Figura 17 – Consumo de processamento do SNORT no primeiro ataque.



Fonte: Autor (2020).

Para estes experimentos, levou-se em consideração 3 (três) fatores: o tipo de ataque DoS (UDP Flood e SYN Flood), o tamanho do pacote (150KB e 1500KB) e o tipo de ambiente (local ou distribuído). Cada combinação foi executada 15 vezes com duração de 5 minutos cada. A Tabela 4 apresenta as combinações de fatores que foram levadas em consideração nos experimentos desta seção. As métricas usadas foram: o consumo de processamento, o consumo de memória e o consumo de banda de rede. É importante salientar que os experimentos foram executados 15 vezes cada, buscando atingir maior precisão estatística nos resultados.

Para prosseguir com os demais experimentos, definiu-se o tamanho dos pacotes como 150 KB e 1500 KB. O tamanho do pacote de 1500 KB utilizado nos experimentos foi determinado com base no consumo do recurso de processamento observado nos experimentos realizados no cenário local (Seção 4.3) e no cenário distribuído (Seção 4.4), observando o consumo até chegar no gargalo do recurso de processamento. O tamanho do pacote de 150KB utilizado nos experimentos foi escolhido pela possibilidade de identificar o consumo do recurso de processamento durante os experimentos do cenário local na Seção 4.3 e do cenário distribuído na Seção4.4.

Tabela 4 – Tabela de combinações.

Combinações	Tipo de Ataque	Tamanho do Pacote	Ambiente
Combinação 1	UDP Flood	1500KB	Local
Combinação 2	UDP Flood	1500KB	Distribuído
Combinação 3	UDP Flood	150KB	Local
Combinação 4	UDP Flood	150KB	Distribuído
Combinação 5	TCP SYN Flood	1500KB	Local
Combinação 6	TCP SYN Flood	1500KB	Distribuído
Combinação 7	TCP SYN Flood	150KB	Local
Combinação 8	TCP SYN Flood	150KB	Distribuído

Fonte: Autor (2020).

4.5.1 Análise de consumo de processamento para o uso de contêiner

A Tabela 5 apresenta o consumo de processamento e os respectivos intervalos de confiança[7] para as combinações executadas. A combinação com menor consumo de processamento utilizou 42,27% de CPU, enquanto a maior utilizou 80,87%. Uma diferença de 38,6% entre elas. As combinações TCP SYN Flood/Distribuído/1500KB e TCP SYN Flood/Distribuído/150KB foram influenciadas pelo tamanho do pacote com 78,42% e 80,87% respectivamente com intervalo de confiança de 10,93% apenas na combinação TCP SYN Flood/Distribuído/1500KB, lembrando que o nível de confiança utilizado para o cálculo do intervalo de confiança foi de 95%.

Diante dos experimentos realizados utilizando o protocolo UDP, conforme apresentado na Tabela 5, o consumo de processamento manteve-se abaixo de 50% e o intervalo de confiança

[7] http://www.portalaction.com.br/inferencia/intervalo-de-confianca

Tabela 5 – Tabela de consumo de processamento para o uso de contêiner.

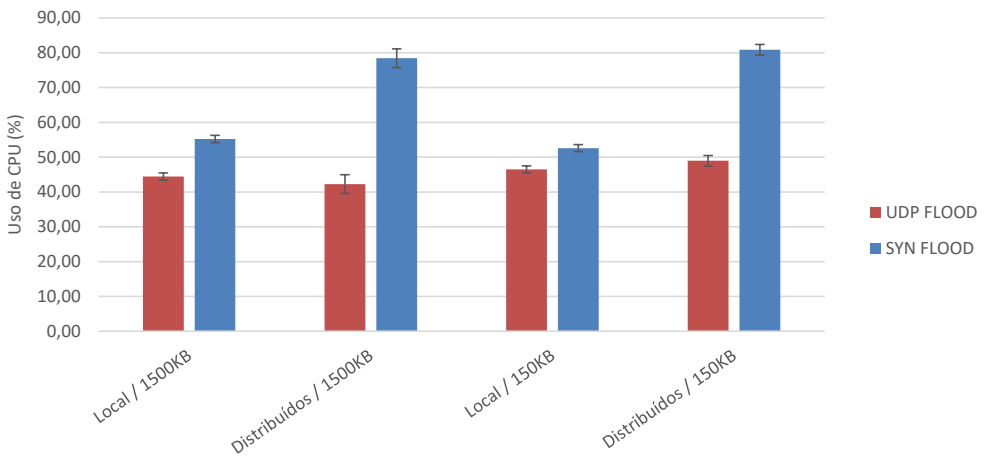
	Processamento (%)	Intervalo de confiança (%)
UDP Flood		
Local / 1500KB	44,43	1,05
Distribuído/1500KB	42,27	2,69
Local/150KB	46,50	1,00
Distribuído/150KB	48,95	1,52
TCP SYN Flood		
Local / 1500KB	55,23	0,76
Distribuído/1500KB	78,42	10,93
Local/150KB	52,61	0,68
Distribuído/150KB	80,87	1,16

Fonte: Autor (2020).

teve variação abaixo de 3%. Nos experimentos utilizando o protocolo TCP (SYN Flood), o uso de recursos de processamento ficou próximo de 80%. Entretanto, na combinação TCP SYN Flood/Distribuído/1500KB, o intervalo de confiança ficou com variação de 10,93%, um valor alto quando comparado às demais combinações mostradas na Tabela 5. Os resultados na Tabela 5 permitem melhor análise do consumo e intervalo de confiança. Além disso, a fim de mostrar o comportamento do consumo de processamento, são apresentados também os resultados da Figura 18.

A diferença do consumo de processamento é que no caso do TCP SYN Flood é explorada a reserva de recursos e armazenamento da tabela para gerenciar conexões e no caso do UDP Flood é explorada a produção de uma resposta ICMP. Outro fator que influenciou no consumo de processamento do TCP foi a garantia de entrega de pacotes de dados. O UDP, por outro lado, não confirma a entrega dos pacotes de dados (TANENBAUM, 2003), o que o ajuda a obter um desempenho superior.

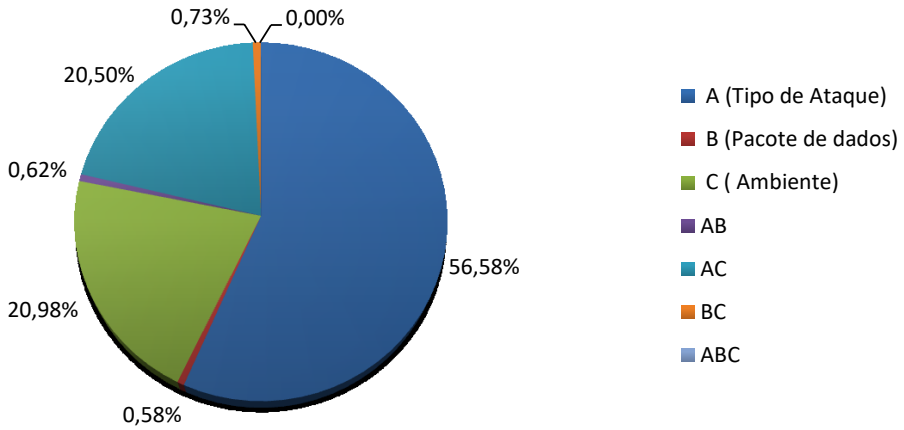
Figura 18 – Consumo do recurso de processamento durante os testes com contêineres.



Fonte: Autor (2020).

Com os resultados obtidos durante a realização dos experimentos, foi possível identificar os fatores que tiveram maior influência no consumo de processamento. De acordo com a Figura 19, o fator que teve maior influência foi a opção “A”, ou seja, o tipo de ataque, com 56,58%. Já o segundo fator com maior influência foi o “C”, que representa o ambiente, atingindo 20,98%. Entretanto, ainda deve ser considerada a associação dos fatores, como é o caso da opção “AC” (categoria de ataque e ambiente), que obteve 20,50% de influência no consumo do recurso de processamento.

Figura 19 – Fatores de influência no consumo de processamento(contêineres).



Fonte: Autor (2020).

4.5.2 Análise de consumo de memória para o uso de contêiner

A Tabela 6 apresenta o consumo de memória gerado para as combinações utilizadas nos experimentos. Ao iniciar os experimentos, o consumo de memória subiu rapidamente e, em aproximadamente 5 segundos, estabilizou. As combinações TCP SYN Flood/Local/1500KB e TCP SYN Flood/Local/150KB foram as que tiveram maior variação, com 7,21% e 3,73%, respectivamente. A Figura 20 ilustra os resultados.

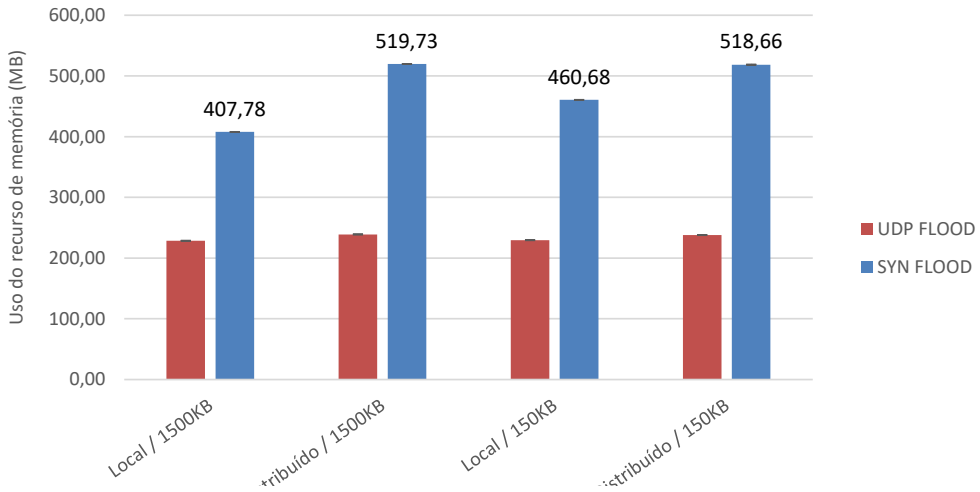
Para o UDP Flood, o consumo de memória varia entre 228,64MB (para cenário local com pacote de 1500KB) e 238,99 (para cenário distribuído com pacote de 1500KB). Por sua vez, o SYN Flood gera consumo de memória entre 407,78 (para o cenário local com pacote de 1500KB) e 519,73 (para o cenário distribuído com pacote de 1500KB). A diferença de memória entre o ataque TCP SYN Flood e UDP Flood deve-se ao fato que o protocolo UDP não realiza confirmação de entrega, enquanto o TCP sempre gera confirmações, o que demanda maior quantidade de memória.

Tabela 6 – Tabela de consumo de memória com o uso de contêiner.

	Uso de memória (MB)	Intervalo de confiança
UDP Flood		
Local / 1500KB	228,64	0,18
Distribuído/1500KB	238,99	0,04
Local/150KB	229,44	0,12
Distribuído/150KB	237,98	0,22
SYN Flood		
Local / 1500KB	407,78	7,21
Distribuído/1500KB	519,73	0,50
Local/150KB	460,68	3,73
Distribuído/150KB	518,66	0,05

Fonte: Autor (2020).

Figura 20 – Consumo do recurso de memória durante a realização do teste com contêiner.



Fonte: Autor (2020).

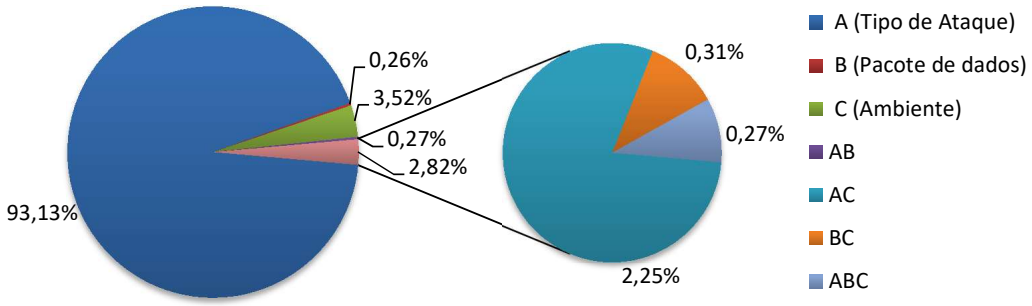
Com os resultados obtidos durante a realização dos experimentos, foi possível identificar o fator que teve a maior influência no consumo de memória. Analisando a Figura 21, o fator que teve maior influência foi a opção “A” (tipo de ataque). Os demais fatores tiveram baixa influência no consumo de memória.

4.5.3 Análise de consumo de banda de rede

A largura de banda de rede está relacionada à capacidade de troca de informações entre dois pontos em uma rede[8]. A quantidade de dados que trafegou na rede durante os ataques foi medido em Gigabytes (GB), usando os tamanhos de pacotes de 150KB e 1500KB e com influência do tipo de cenário. O consumo de banda passante, conforme a Figura 22, foi menor nos cenários locais (entre 103,5GB para a combinação SYN Flood/Local/150KB e 155,21GB para a combinação UDP Flood/Local/1500KB) e maiores para cenários distribuídos (entre 214,61GB

[8] https://www.hostmidia.com.br/blog/trafego-transferencia-e-largura-de-banda/

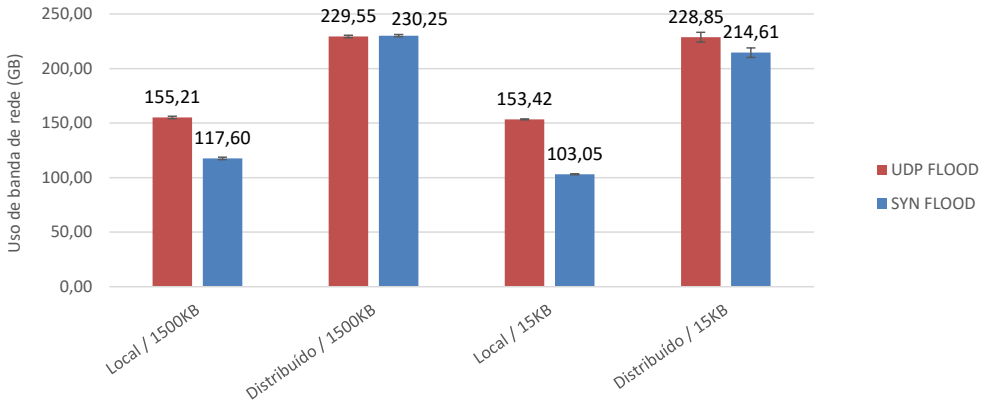
Figura 21 – Fator de maior influência durante o teste com contêineres.



Fonte: Autor (2020).

para a combinação SYN Flood/Distribuído/150KB e 230,25GB para a combinação SYN Flood/-Distribuído/1500KB). Esta diferença de banda deu-se devido ao tipo de cenário em que o ataque ocorreu, sendo menor nos cenários locais por ter apenas um atacante.

Figura 22 – Consumo de banda na execução do teste com contêineres.



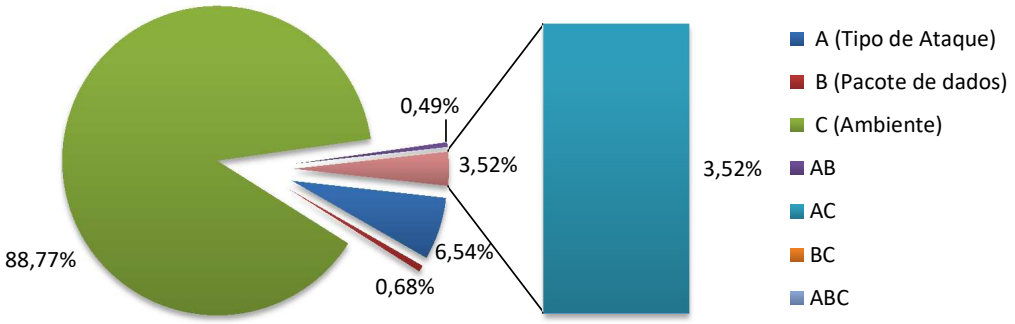
Fonte: Autor (2021).

Em relação ao consumo de banda de rede (Figura 23), o fator que teve a maior influência (Figura 23) foi o fator ambiente (opção “C”) com 88,77%. Os demais fatores tiveram baixa influência no consumo de banda de rede.

4.6 Experimentos utilizando máquina virtual tradicional

Para realizar os experimentos observou-se os mesmos fatores dos experimentos com contêineres, apresentados na Seção 5.2, tais como o tipo de ataque DDoS, o tamanho do pa-

Figura 23 – Fator de maior influência no consumo de banda de rede uso de contêiner



Fonte: Autor (2021).

cote e o tipo de ambiente (local ou distribuído). As combinações dos testes também foram seguidas conforme a Tabela 4. O consumo dos recursos de processamento e de memória RAM foram monitorados de dentro da VM, como também a execução das funções de rede foram nestas VM's. A ferramenta utilizada para a virtualização foi o Virtual Box com 1 núcleo de processamento disponível e 2 GB de memória RAM disponível.

4.6.1 Análise de consumo de processamento

A Tabela 7 apresenta o consumo de processamento e o intervalo de confiança durante a realização dos experimentos. Durante os experimentos, utilizando o tipo de ataque UDP Flood, o consumo de processamento foi maior nas combinações Distribuído/1500KB e Distribuído/150KB, onde houve um consumo de processamento de 46,27% e 57,27%, respectivamente mostrados na Figura 7. Nos ataques do tipo SYN Flood, o consumo de processamento chegou a 87,66% e 91,01% nas combinações Distribuído/1500KB e Distribuído/150KB, respectivamente. Esse consumo maior nos ataques distribuídos deve-se à característica do ataque DDoS, tendo dois atacantes simultâneos.

Nos experimentos realizados utilizando o protocolo UDP, conforme observado na Tabela 7, o intervalo de confiança ficou abaixo de 2%. Nos experimentos utilizando o protocolo TCP (SYN Flood), o consumo de processamento ficou entre 75,24% e 91,01%, e o intervalo de confiança ficou em 1,87% e 0,51%, respectivamente.

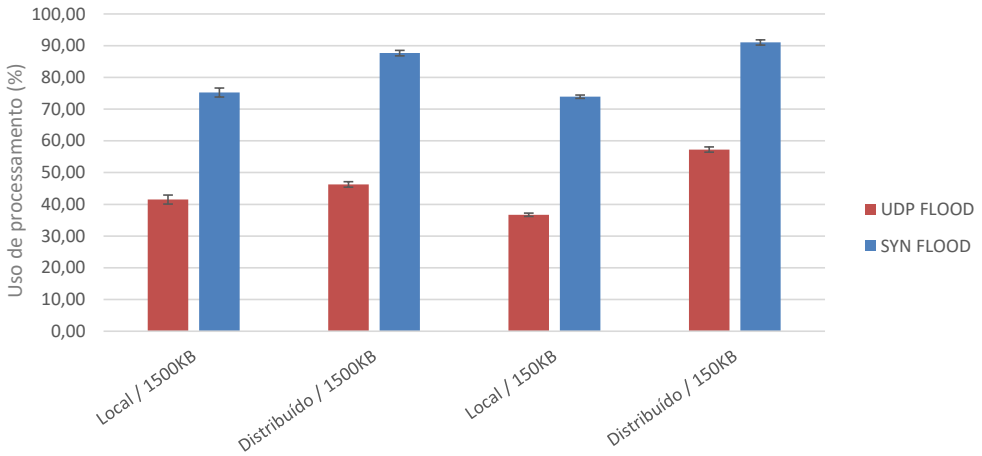
Com os resultados obtidos dos experimentos realizados com a VM, calculou-se a influência dos fatores no consumo de processamento, como pode ser visto na Figura 25. O fator com maior influência foi o tipo de ataque com 85,39% (Fator "A"), seguido do ambiente com 12,03% (Fator "C"). Entretanto, ainda deve ser considerada a associação dos fatores, como o

Tabela 7 – Tabela de consumo de processamento com a NFV-IDS.

	Processamento (%)	Intervalo de Confiança
UDP Flood		
Local / 1500KB	41,50	1,42
Distribuído/1500KB	46,27	0,86
Local/150KB	36,71	0,53
Distribuído/150KB	57,27	0,83
SYN Flood		
Local / 1500KB	75,24	1,87
Distribuído/1500KB	87,66	1,19
Local/150KB	73,92	1,10
Distribuído/150KB	91,01	0,51

Fonte: Autor (2020).

Figura 24 – Consumo de processamento durante a execução dos testes com a VMs.



Fonte: Autor (2020).

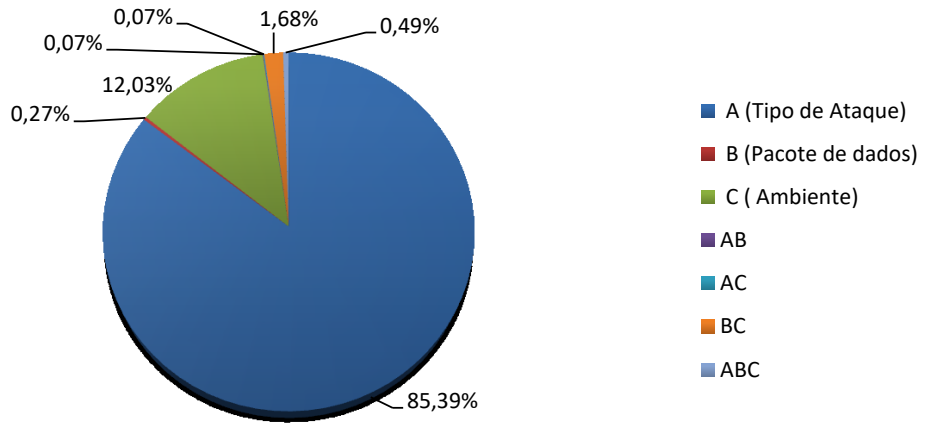
caso “BC” (pacote de dados e ambiente), que representou 1,68% de influência no consumo do recurso de processamento.

4.6.2 Análise do consumo de memória (VMs)

Na Tabela 8 o intervalo de confiança na combinação UDP Flood/Local/1500KB teve a maior margem de incerteza, sendo 7,85MB. No entanto, nos demais testes, a variação ficou abaixo de 4MB. Já na combinação SYN Flood/Distribuído/150KB, a maior variação foi de 4,47MB, enquanto a menor foi na combinação SYN Flood/Distribuído/1500KB com valor de 1,66MB.

O consumo de memória durante a realização dos experimentos podem ser vistos na Figura 26, com números entre 1482,42MB e 1523,23MB para os experimentos do tipo UDP Flood, e entre 1752,53MB e 1824,36MB para os experimentos do tipo SYN Flood. Esse consumo de memória acima de 1400MB deve-se à necessidade de virtualizar um *hardware* completo, como um computador real.

Figura 25 – Fatores de influências no consumo de processamento.



Fonte: Autor (2020).

Tabela 8 – Tabela de consumo de memória da VM com a NFV-IDS.

	Uso Memória/MB	**Intervalo de Confiança MB**
UDP Flood		
Local / 1500KB	1482,42	7,85
Distribuído/1500KB	1507,33	3,72
Local/150KB	1470,70	1,94
Distribuído/150KB	1523,23	2,08
SYN Flood		
Local / 1500KB	1772,85	2,65
Distribuído/1500KB	1824,36	1,66
Local/150KB	1752,53	4,47
Distribuído/150KB	1818,84	3,23

Fonte: Autor (2020).

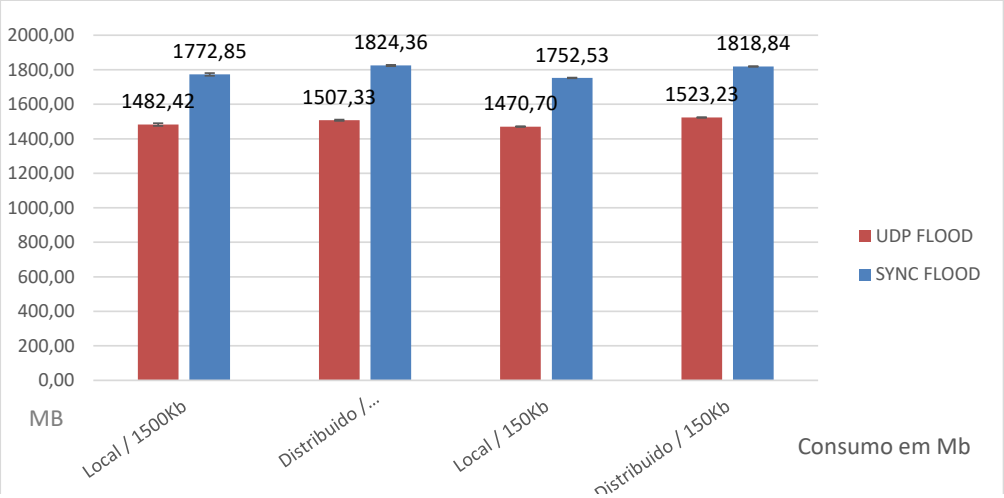
Durante a execução dos experimentos, o fator que teve maior porcentagem de influência no consumo do recurso de memória foi o tipo de ataque, como visto na Figura 27. Os demais fatores tiveram baixa influência no consumo de memória, como o fator ambiente com somente 2,63%.

4.6.3 Análise de consumo de banda de rede

Em relação ao consumo de banda de rede utilizando VMs. A quantidade de dados que trafegaram na interface de rede foi grande devido ao tamanho dos pacotes utilizados nos ataques, o que fez com que a interface de rede trabalhasse próximo de seu desempenho máximo. O consumo de banda de rede, em alguns casos, ultrapassou os 200 (duzentos) gigabytes, conforme visto na Figura 28.

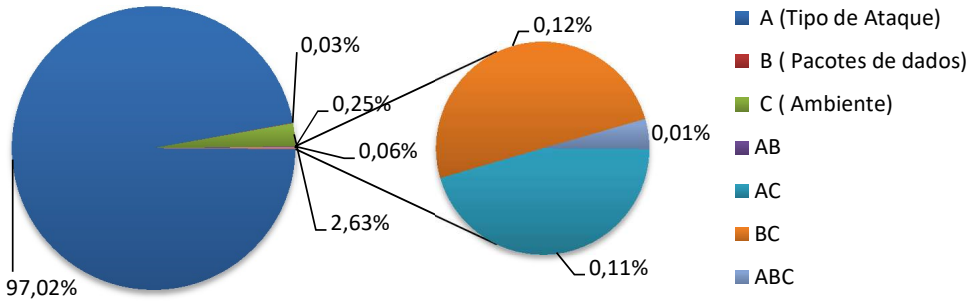
Conforme a Figura 23, o fator que teve maior influência foi o ambiente (Fator “C”) com 93,77%. Os demais fatores tiveram baixa influência no consumo de banda de rede, sendo o pacote de dados (Fator “B”) o segundo, com 4,75% de influência.

Figura 26 – Consumo do recurso de memória durante testes com a VM.



Fonte: Autor (2020).

Figura 27 – Fator de maior influência na execução dos experimentos com a VM.

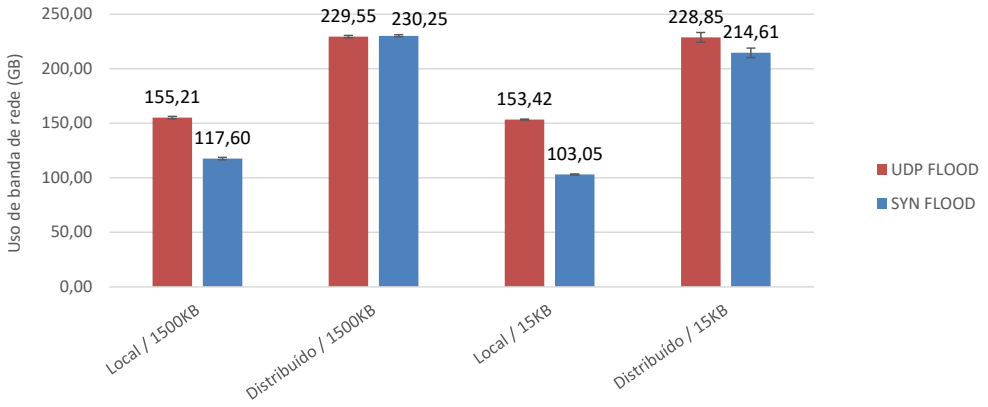


Fonte: Autor (2020).

4.7 Comparação entre as formas de virtualização de um NFV

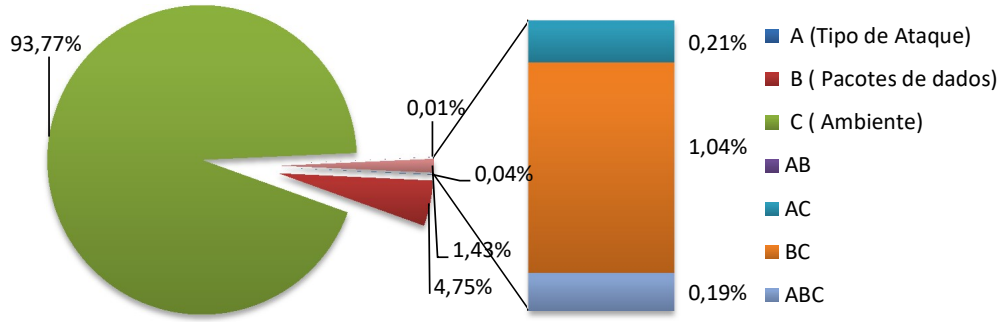
Um ambiente virtual pode ser uma opção flexível, escalável e com custo menor para as empresas de telecomunicações. Um fator importante é a decisão sobre qual forma de virtualização vale mais a pena adotar, reduzindo o consumo de recursos computacionais sempre que possível. Esta seção vai analisar o consumo de recursos computacionais para cada modelo de virtualização, além dos fatores mais influentes. Na Figura 30 pode ser visualizada uma comparação simples e direta, onde fica clara uma grande vantagem a favor da NFV-IDS utilizando contêineres (neste caso, Docker), onde o consumo de memória de armazenamento é de aproximadamente 9,5% quando comparado com a virtualização tradicional por VM.

Figura 28 – Consumo de banda na execução dos testes com VM.



Fonte: Autor (2021).

Figura 29 – Fator de maior influência durante os testes com VM.

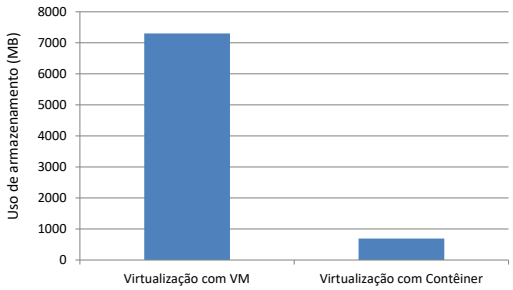


Fonte: Autor (2021).

4.7.1 Comparação de recurso de processamento (Contêiner X VM)

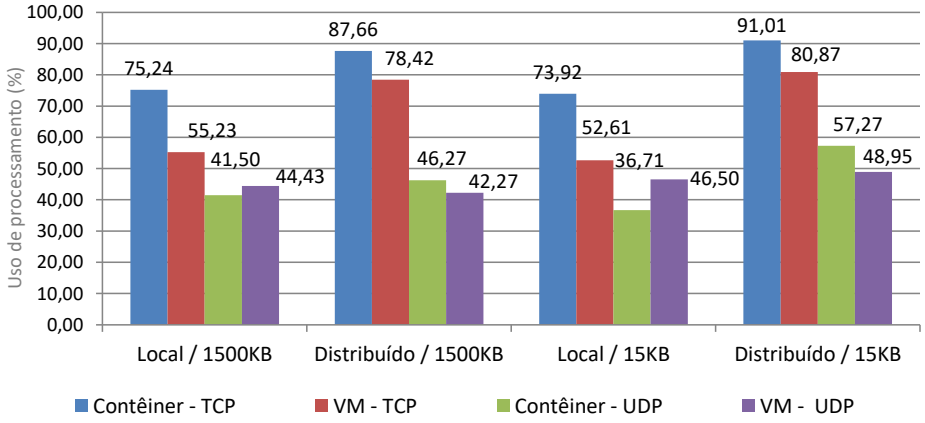
O consumo de processamento sofreu com alguns fatores, como: o tipo de ataque, o ambiente que ocorreu o ataque e o tamanho do pacote de dados. Durante os ataques do tipo SYN Flood, o consumo de processamento da NFV-IDS, em execução no contêiner, foi maior quando comparado com o ataque da mesma natureza realizada na VM tradicional. Na Figura 31 é possível verificar essa diferença de consumo. Entretanto, durante o ataque do tipo UDP Flood, o recurso de processamento teve menor consumo em três dos quatro ambientes de teste. É importante salientar que o maior consumo dos recursos de processamento ocorreu devido à necessidade de resposta durante um ataque do tipo SYN Flood. Na combinação Contêiner utilizando TCP, os casos Distribuído/1500KB e Distribuído/150KB tiveram consumos de 87,66% e 91,01%, respectivamente, sendo os consumos mais altos durante a realização dos experimentos.

Figura 30 – Comparação do consumo de espaço em Disco (Conteiner X Vm).



Fonte: Autor (2020).

Figura 31 – Comparação de processamento ataque entre VM e contêiner.



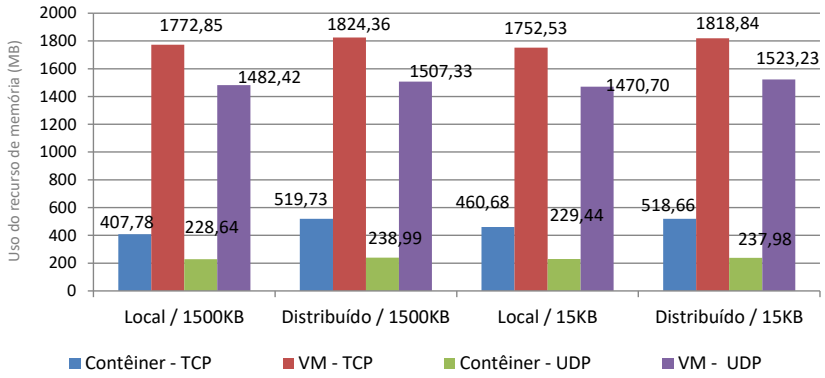
Fonte: Autor (2020).

Ainda observando os dados da Figura 31, durante a realização do ataque do tipo UDP Flood, o menor consumo do recurso de processamento utilizado pelo contêiner foi de 36,71% na combinação UDP/Local/150KB, inferior ao consumo da VM tradicional que foi de 45,27% na combinação UDP/Distribuído/1500KB. Isto deve-se ao fato de que um ataque UDP nunca espera resposta, deste modo gerando tráfego de rede sem necessidade de resposta.

4.7.2 Comparação de recurso de memória RAM (Contêiner X VM)

Ao executar a NFV-IDS utilizando o contêiner no Docker, o consumo do recurso de memória foi menor quando comparado com a VM tradicional. Esta diferença pode ser observada na Figura 32, onde o consumo de memória do contêiner não passou dos 600MB durante a execução do ataque SYN Flood, e durante a execução do ataque UDP Flood ficou abaixo dos 300MB.

Figura 32 – Comparação do consumo do recurso de memória entre VM e contêiner.

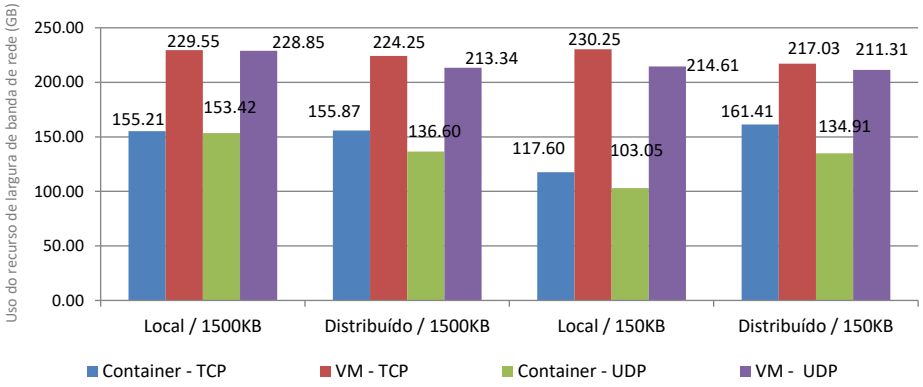


Fonte: Autor (2020).

4.7.3 Comparação de banda de rede (Contêiner X VM)

A largura de banda de rede está relacionada à capacidade de troca de informações entre dois pontos em uma rede[9]. A quantidade de dados que trafegou na rede durante os ataques foi medido em Gigabytes (GB), usando os tamanhos de pacotes de 150KB e 1500KB e com influência do tipo de cenário. O consumo de banda passante, conforme a Figura 33, foi menor nas combinações do que utilizaram a virtualização por contêineres, e maior nas combinações com virtualização tradicional. Esta diferença de banda deu-se devido ao tipo as VMs serem criadas em cima de um hipervisor que emula hardware real. Isso inclui a emulação de dispositivos de rede virtuais. O processo de emulação adiciona latência e processamento extra, resultando em um consumo mais elevado de recursos de rede. Em contraste, contêineres compartilham o kernel do sistema operacional do host.

Figura 33 – Comparação do consumo de banda de rede entre contêineres e VM's.



Fonte: Autor (2020).

[9] https://www.hostmidia.com.br/blog/trafego-transferencia-e-largura-de-banda/

A capacidade de transmissão de pacotes sofreu variação nos experimentos realizados. Entretanto, comparando o ataque do tipo DDoS (distribuído), o resultado do consumo de banda de rede manteve-se com diferença pequena, pois a limitação na capacidade de transmissão da rede deu-se pelo tamanho do pacote de dados e da interface de rede utilizada. Ainda é possível observar que o consumo de banda de rede utilizando VMs foi similar aos testes utilizando contêineres. A quantidade de dados que trafegaram na interface de rede foi grande devido à quantidade de pacotes enviados nos ataques, o que fez com que a interface de rede trabalhasse próximo ao seu limite. O consumo de banda de rede, em alguns casos, ultrapassou os 200 (duzentos) gigabytes, conforme visto na Figura 33.

5 RESULTADOS E DISCUSSÕES

5.1 Considerações Iniciais e direcionamento

Este capítulo apresenta os experimentos realizados e os resultados obtidos do direcionamento do trabalho através do capítulo 4. Após a realização dos testes comparativos entre os sistemas de virtualização por VM e contêineres, foi possível determinar o foco na virtualização por contêineres, devido à economia de recursos computacionais e à facilidade que o Docker permite. Os experimentos iniciais permitiram direcionar o andamento deste trabalho, analisar os fatores que mais influenciam nos resultados, determinar o melhor sistema de virtualização e, assim, realizar os testes finais. Os fatores foram revistos e identificou-se que o tipo de ambiente local não é relevante, visto que o ataque DDoS não concorre com os recursos da mesma máquina. Ao observar a ferramenta LOIC compreendeu-se esta necessidade. O fator tipo de ataque (HTTP, UDP ou TCP), a velocidade que ocorrem as requisições (*slow* ou *fast*), a quantidade de requisições e a quantidade de *threads*.

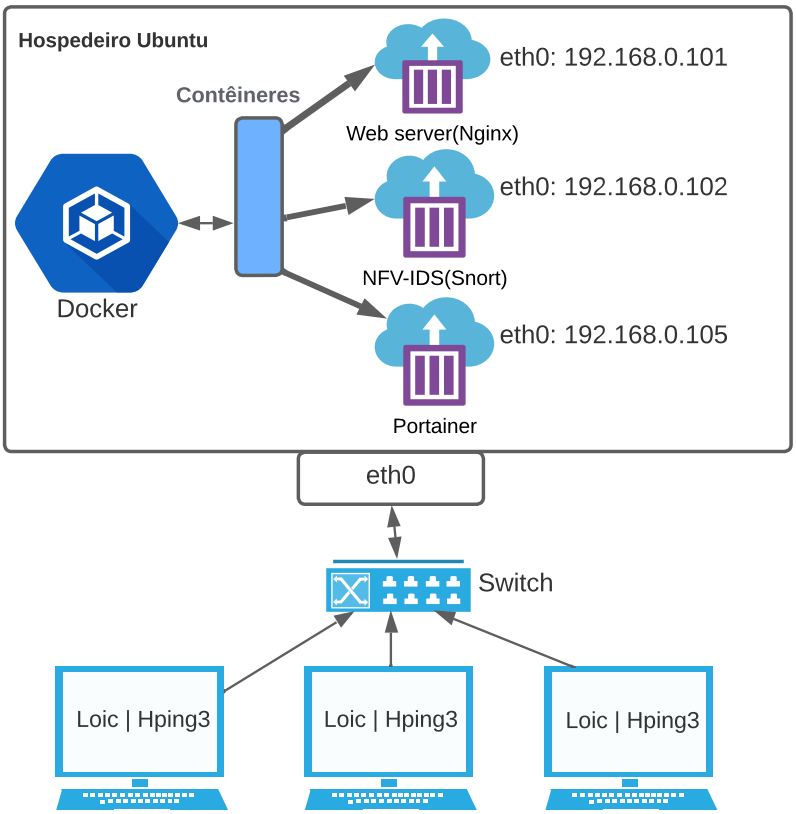
Para estes experimentos o intuito foi analisar o comportamento do hospedeiro compartilhando recursos entre a NFV-IDS e o servidor Web, e, assim, mostrar os resultados obtidos. Neste cenário utilizou-se uma conexão de rede cabeada, implementada em uma rede controlada conforme pode ser visto na Figura 34. Na figura é apresentado o *hardware* hospedeiro e como ficou a organização do contêiner executando a NFV-IDS e servidor Web.

5.2 Experimentos finais utilizando contêineres

Por meio de experimentos iniciais, apresentados na Seção 4.4, foi possível compreender o comportamento do IDS baseado em NFV e definir qual ferramenta utilizar. Devido ao fato de o Hping3 não permitir ataque do tipo HTTP, foi necessário recorrer a outra ferramenta, o LOIC. Através destes experimentos iniciais foi possível determinar o direcionamento de qual sistema de virtualização analisar nos testes finais e também o comportamento de uma NFV em execução no Docker.

Para os experimentos finais, foi necessário revisar os três fatores utilizados nos experimentos iniciais: o tipo de ataque DDoS (HTTP, UDP ou TCP), a velocidade das requisições e a quantidade de threads. Essa revisão foi necessária pois a ferramenta LOIC já vem com configurações pré-definidas, o que significa que não é possível alterar o tamanho do pacote de dados, mas é possível modificar a velocidade do ataque, a quantidade de threads utilizadas e o tipo de ataque (TCP, UDP ou HTTP). Com base nessas opções, foram determinadas 12 combinações diferentes de experimentos, cada uma executada 15 vezes, com duração de 5 minutos por execução. A Tabela 9 apresenta as combinações de fatores considerados nos experimentos desta seção. As métricas analisadas foram as mesmas dos experimentos iniciais: consumo de

Figura 34 – Topologia do cenário



Fonte: Autor (2021).

processamento, consumo de memória e consumo de banda de rede. É importante salientar que os experimentos foram executados 15 vezes para alcançar maior confiabilidade estatística nos resultados.

Uma característica observada ao utilizar o LOIC para a realização dos experimentos como atacante é que o consumo de banda de rede é baixo devido ao tamanho dos pacotes, porém a quantidade de requisições é alta, com objetivo de estressar o processamento e consumo de memória RAM. Conforme observado na Tabela 9, o fator “Velocidade do ataque” está relacionado diretamente à quantidade de requisições executadas durante os ataques.

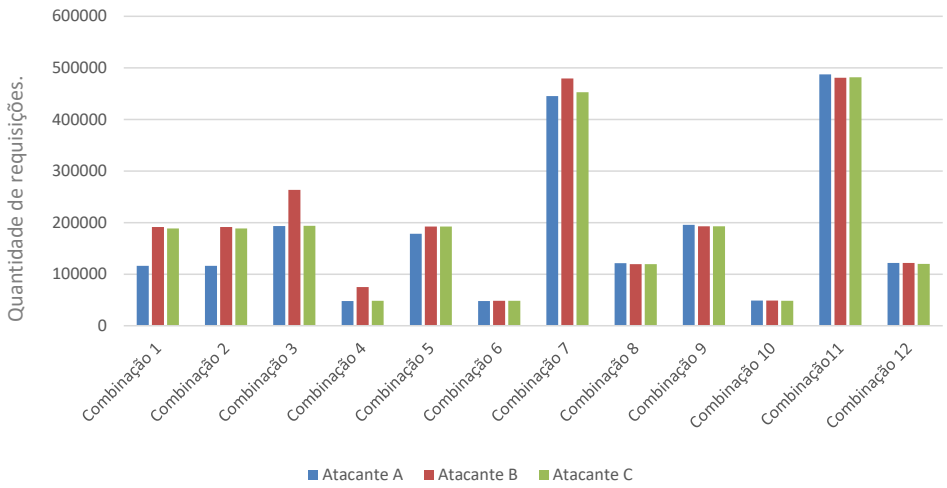
Durante a definição dos parâmetros, foi observado que a partir de 25 threads ocorria perda de pacotes por parte dos atacantes. Por outro lado, abaixo de 10 threads, o consumo de processamento era próximo a zero. Assim, foram estabelecidos limites para o número de threads: o máximo seria 25 e o mínimo, 10. Com base nisso, foram realizadas 12 combinações diferentes de experimentos, que contemplaram métricas relacionadas ao processamento, memória e consumo de banda. Na Figura 35 é possível verificar a média de requisições de cada combinação por atacante. Nas combinações 1 e 2, o atacante ’A’ apresentou quantidade de requisições inferior aos atacantes ’B’ e ’C’. Já nas combinações 3, 4 e 7, o atacante ’B’ se destacou em relação aos demais. É importante salientar que a ordem em que os experimentos foram realizados pode ter influenciado essas variações.

Tabela 9 – Combinações.

Combinações	Tipo de Ataque	Velocidade do ataque	Threads
Combinação 2	HTTP	Fast	10
Combinação 3	HTTP	Slow	10
Combinação 1	HTTP	Fast	25
Combinação 4	HTTP	Slow	25
Combinação 5	UDP	Fast	10
Combinação 6	UDP	Slow	10
Combinação 7	UDP	Fast	25
Combinação 8	UDP	Slow	25
Combinação 9	TCP	Fast	10
Combinação 10	TCP	Slow	10
Combinação 11	TCP	Fast	25
Combinação 12	TCP	Slow	25

Fonte: Autor (2020).

Figura 35 – Requisições por atacante.



Fonte: Autor(2022).

5.2.1 Análise de consumo de processamento NFV-IDS

A Tabela 10 apresenta o consumo de processamento e os respectivos intervalos de confiança para as combinações executadas. A combinação com menor consumo de processamento utilizou 0,14% de CPU, enquanto a maior utilizou 36,45%. Essa diferença de 36,31% entre elas foi influenciada pelo tipo de ataque, onde durante o ataque do tipo TCP a função de rede NFV-IDS recebe uma carga maior, devido ao tipo de requisição.

Conforme a Tabela 10, o consumo de processamento da NFV-IDS para o TCP manteve-se abaixo de 40% e o intervalo de confiança teve variação abaixo de 1%. Nas combinações UDP o consumo de processamento ficou baixo, devido às características do protocolo de comunicação. Como não há necessidade de respostas, este ataque é utilizado normalmente para comprometer a interface de comunicação. No ataque DDoS do tipo HTTP, o consumo de proces-

Tabela 10 – Consumo de processamento NFV-IDS.

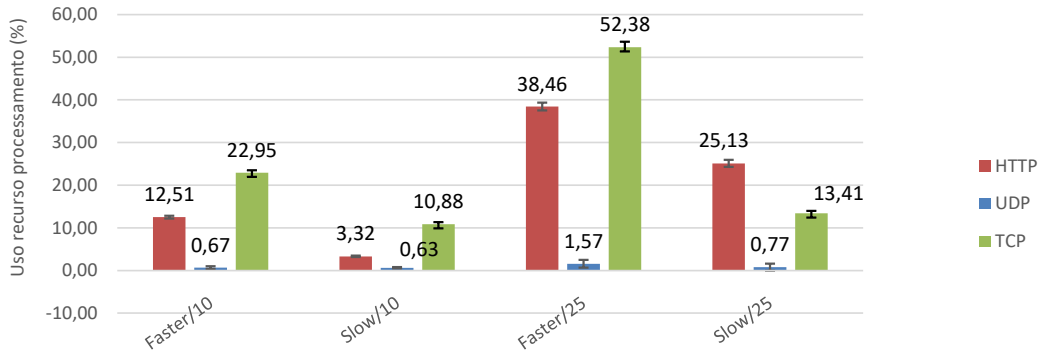
	Processamento (%)	Intervalo de confiança
HTTP		
Faster/10	12,51	0,33
Slow/10	0,14	0,01
Faster/25	17,01	0,28
Slow/25	7,27	0,13
UDP		
Faster/10	0,67	0,02
Slow/10	0,63	0,02
Faster/25	1,57	0,07
Slow/25	0,77	0,03
TCP		
Faster/10	12,47	0,25
Slow/10	10,88	0,46
Faster/25	36,45	0,45
Slow/25	5,91	0,10

Fonte: Autor (2021).

samento manteve-se abaixo de 20%, devido à função de rede ser apenas de detecção, desta forma ela apenas interceptava os pacotes mostrando através do console de monitoramento.

Outro fator que influenciou no consumo de processamento do TCP foi a necessidade do protocolo de rede garantir a entrega dos pacotes de dados. O UDP, por outro lado, não confirma a entrega dos pacotes de dados[1], desta forma reduz o consumo de processamento, visto que ele apenas monitora os pacotes recebidos e não realiza tratativas.

Figura 36 – Consumo do recurso de processamento durante os testes com NFV-IDS.



Fonte: Autor (2021).

Com os resultados dos experimentos, foi possível identificar os fatores que tiveram maior influência no consumo de processamento da NFV-IDS, a Figura 37 mostra que a combinação "TCP/Faster/25" obteve o maior consumo do recurso de processamento, chegando a 52,38%, enquanto a combinação "UDP/Slow/10" teve o menor consumo com apenas 0,63%. Este con-

[1] https://www.geeksforgeeks.org/differences-between-tcp-and-udp/

sumo de processamento está ligado diretamente ao ataque realizado no servidor Web, pois o NFV-IDS está analisando todos os pacotes que passam pela interface de rede do hospedeiro.

5.2.2 Análise de consumo de processamento WEB

O consumo do recurso de processamento do servidor web NGINX é mostrado na tabela 11, bem como os respectivos intervalos de confiança. É possível observar que o fator tipo de ataque “HTTP” teve o maior consumo de processamento, se comparado ao UDP e TCP. A combinação com menor consumo de processamento foi do tipo UDP que utilizou apenas 0,02% de CPU, enquanto a maior do tipo HTTP utilizou 100,13%. Essa diferença de aproximadamente 99% entre elas é resultado do tipo de ataque, onde durante o ataque do tipo HTTP o servidor web recebe uma carga maior devido ao tipo de requisição ser HTTP.

Tabela 11 – Consumo de processamento (WEB).

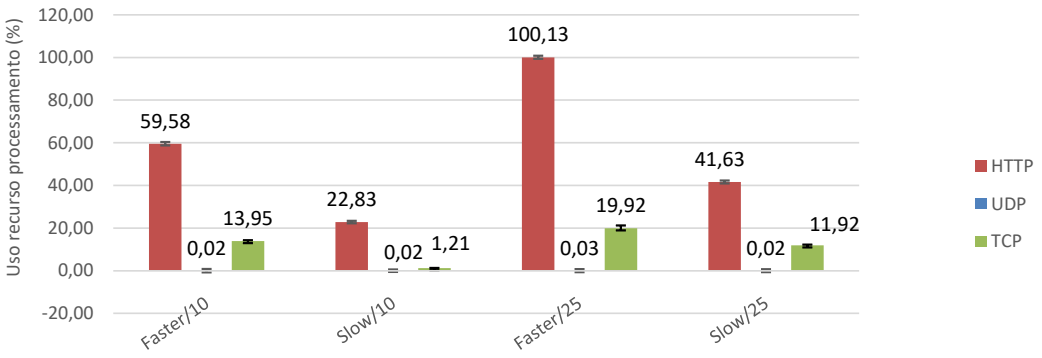
	Processamento (%)	**Intervalo de confiança (%)**
HTTP		
Faster/10	59,58	0,80
Slow/10	22,83	0,57
Faster/25	100,13	0,72
Slow/25	41,63	0,69
UDP		
Faster/10	0,02	0,00
Slow/10	0,02	0,00
Faster/25	0,03	0,01
Slow/25	0,02	0,00
TCP		
Faster/10	13,95	0,36
Slow/10	1,22	0,09
Faster/25	19,92	1,30
Slow/25	11,92	0,32

Fonte: Autor (2021).

Conforme a Tabela 11, o consumo de processamento do servidor web nos ataques do tipo UDP manteve-se abaixo de 1% e o intervalo de confiança teve variação abaixo de 0,02%. O motivo destes valores serem tão baixos está relacionado à forma como ocorre um ataque UDP, onde o servidor web não precisa responder nenhuma requisição, por estar processando apenas pacotes do tipo HTTP. Esta característica do protocolo de comunicação mostra que o consumo de processamento não é afetado, entretanto, a interface de comunicação recebe diversos pacotes, podendo comprometer o recebimento de requisições do servidor web. No ataque do tipo TCP o consumo de processamento manteve-se abaixo de 20%, um consumo baixo, entretanto já preocupante, pois um ataque deste tipo acaba afetando o servidor web.

Os resultados dos experimentos possibilitaram identificar os fatores que tiveram maior influência no consumo de processamento da NFV-IDS. A Figura 37 mostra que a combinação

Figura 37 – Consumo do recurso de processamento durante os testes com Servidor Web



Fonte: Autor (2021).

"HTTP/Faster/25" obteve o maior consumo do recurso de processamento, chegando a 100,13%, enquanto a combinação "UDP/Slow/10" teve o menor consumo com apenas 0,02% .

5.2.3 Análise de consumo de memória NFV-IDS

A Tabela 12 apresenta o consumo de memória gerado nos experimentos através das combinações utilizadas. Ao iniciar os experimentos das combinações UDP, TCP, o consumo de memória ficou estável, tendo pequenas variações conforme mostrado através do intervalo de confiança. As combinações do tipo HTTP tiveram maior variação, entretanto podem ser consideradas baixas, pois o intervalo ficou abaixo de 4MB.

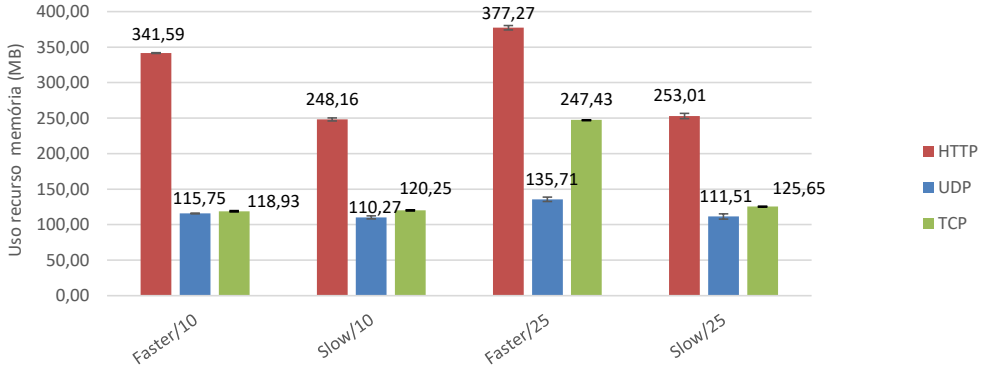
Tabela 12 – Consumo de memória do contêiner com a NFV-IDS.

	Uso de memória (MB)	**Intervalo de confiança (MB)**
HTTP		
Faster/10	341,59	0,55
Slow/10	248,16	2,14
Faster/25	377,2	3,08
Slow/25	253,01	3,7
UDP		
Faster/10	115,75	0,41
Slow/10	110,27	1,21
Faster/25	129,07	0,36
Slow/25	111,51	1,96
TCP		
Faster/10	118,93	0,86
Slow/10	120,25	0,67
Faster/25	247,43	0,20
Slow/25	125,65	0,38

Fonte: Autor (2021).

O consumo do recurso de memória nas combinações do tipo UDP, ficou abaixo de 130MB e com intervalo de confiança abaixo de 2MB, conforme mostrado na Tabela 12. Ao observar a Figura 38, a combinação do tipo UDP obteve um consumo de recurso de memória abaixo de 136MB. Nas combinações do tipo TCP, apenas na combinação TCP/Faster/25 o consumo foi de 247,43MB, consumo alto quando comparado as demais combinações do tipo TCP, esse consumo deveu-se a forma como ocorreu as requisições. Já nas requisições do tipo HTTP o consumo do recurso de memória foi superior as demais combinações.

Figura 38 – Consumo do recurso de memória da NFV-IDS.



Fonte: Autor (2022).

5.2.4 Análise de consumo de memória Web

Ao iniciar os experimentos das combinações UDP, TCP, o consumo médio de memória apresentou pequenas variações conforme mostrado através do intervalo de confiança na Tabela 13. As combinações do tipo HTTP tiveram maior variação e apresentaram características diferentes das demais combinações.

O consumo de memória nas combinações do tipo UDP e TCP, mantiveram-se com a média abaixo de 10MB e com intervalo de confiança abaixo de 1MB, conforme mostrado na Tabela 13. Na Figura 39 é possível observar que o consumo de memória com quantidade de requisições da combinação UDP mantiveram-se próximos, deste modo, o servidor WEB foi pouco afetado no consumo do recurso de memória. Ainda é possível observar que as combinações do tipo TCP/Faster/10 e TCP/Faster/25 mantiveram-se com valores próximos. Já nas requisições do tipo Slow o consumo manteve-se abaixo de 5,6MB, considerado um consumo baixo, o que mostra que a memória é pouco afetada durante os ataques.

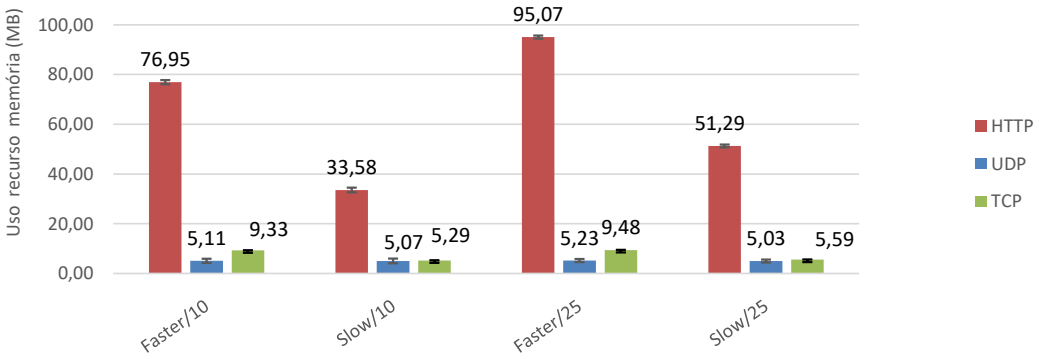
Durante os experimentos, observou-se que as combinações do tipo HTTP apresentaram características atípicas, grande consumo de memória quando comparado com os tipos UDP e TCP. Na Figura 40 é possível observar os resultados obtidos durante a realização dos experimentos em que o consumo de memória RAM é sempre crescente e contínuo, aumentando conforme o tempo vai passando. Este caso foi descoberto ao acaso, visto que nos testes pre-

Tabela 13 – Consumo médio de memória WEB.

	Uso de memória (MB)	Intervalo de confiança (MB)
HTTP		
Faster/10	76,95	0,83
Slow/10	33,58	0,95
Faster/25	95,07	0,62
Slow/25	51,29	0,58
UDP		
Faster/10	5,11	0,05
Slow/10	5,07	0,07
Faster/25	5,23	0,06
Slow/25	5,03	0,17
TCP		
Faster/10	9,33	0,04
Slow/10	5,29	0,05
Faster/25	9,48	0,05
Slow/25	5,59	0,06

Fonte: Autor (2022).

Figura 39 – Consumo do recurso de memória do contêiner.



Fonte: Autor (2022).

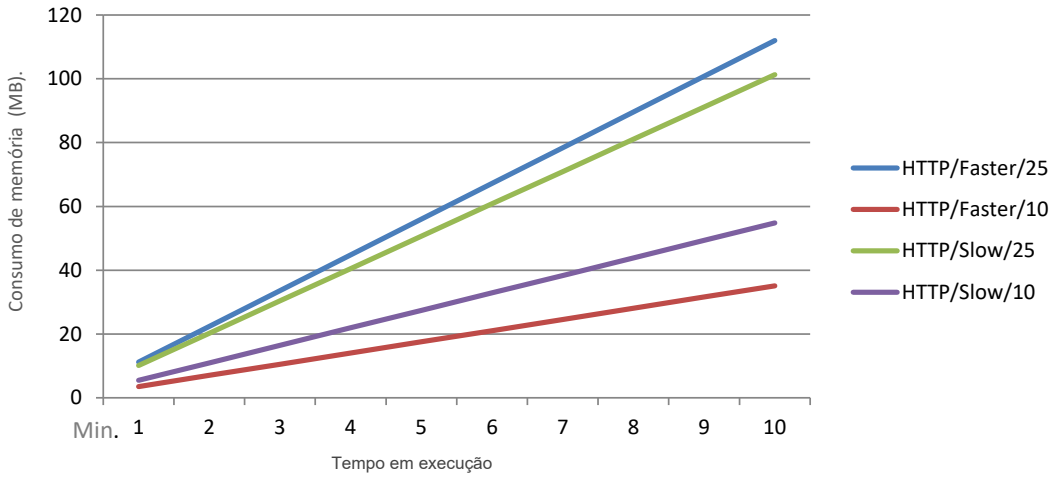
liminares foram executados somente nos protocolos TCP e UDP com diferentes fatores, não sendo possível identificar esta característica.

Nas combinações HTTP/Faster/25 e com 3 atacantes, o consumo do recurso de memória do servidor WEB cresceu cerca de 11,2MB a cada minuto durante os experimentos. Caso um ataque ocorra seguindo estas características, será difícil de identificá-lo, podendo levar ao gargalo da memória RAM disponível em poucas horas.

5.2.5 Análise de consumo de banda de rede.

A quantidade de dados que trafegou na rede durante os ataques foi medido em Gigabytes (GB), onde foi analisado a quantidade de dados recebidos com influência do tipo de ataque, quantidade de requisições e *threads* disponíveis. A interface de rede do hospedeiro foi compartilhada entre a NFV-IDS e o servidor Web. Na Tabela 14, o consumo de banda de rede foi maior

Figura 40 – Consumo contínuo do recurso de memória.



Fonte: Autor (2022).

durante os ataques do tipo HTTP, chegando nos 2,5GB na combinação HTTP/Faster/25. Já o menor consumo foi nos ataques do tipo UDP, onde o menor ataque foi na combinação UDP/Slow/10. Estes dados foram coletados da interface de rede compartilhada entre a NFV-IDS e o servidor web. Ao analisarmos o consumo de banda de rede com testes iniciais podemos ver nitidamente a diferença das ferramentas, o Hping3 permite manipular o tamanho do pacote enviado na requisição, utilizando pacotes de dados com tamanhos grandes é possível gerar um alto consumo do recurso de rede. Entretanto, por utilizar pacote de dados grande pode ser facilmente detectado pelo NFV-IDS.

Tabela 14 – Consumo médio de banda passante (GB).

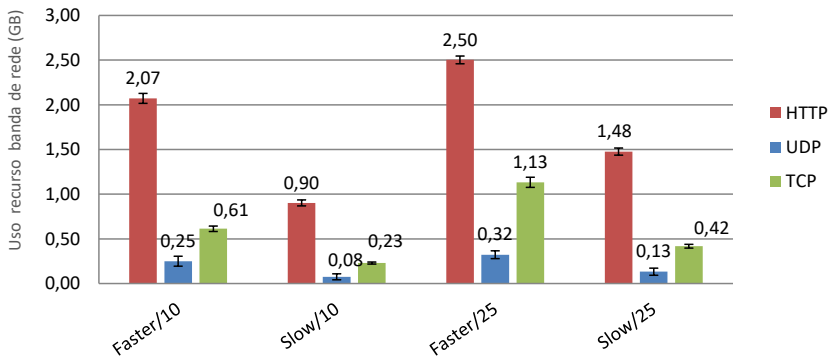
	Uso de banda de rede (GB)	**Intervalo de confiança**
HTTP		
Faster/10	2,07	0,06
Slow/10	0,90	0,03
Faster/25	2,50	0,04
Slow/25	1,48	0,04
UDP		
Faster/10	0,25	0,05
Slow/10	0,08	0,00
Faster/25	0,32	0,00
Slow/25	0,13	0,00
TCP		
Faster/10	0,61	0,05
Slow/10	0,23	0,00
Faster/25	1,13	0,03
Slow/25	0,42	0,01

Fonte: Autor (2022).

O consumo de banda passante, conforme a Figura 41, foi menor nos ataques do tipo UDP. Este consumo menor do recurso de banda é devido ao tipo de ataque que a ferramenta

LOIC executa, onde o intuito é simular um ataque real. Deste modo ela cria requisições como se fossem usuários reais. Durante os testes de consumo de banda passante é possível compreender que esta ferramenta trabalhou melhor com ataques do tipo HTTP, apesar de ter tido bom resultado com as demais combinações.

Figura 41 – Consumo de banda de rede.



Fonte: Autor (2022).

A capacidade de transmissão de pacotes sofreu variação durante os experimentos, entretanto, comparando com os resultados obtidos nos testes preliminares mostrados na subseção 4.7.3, o consumo de banda de rede passante foi baixo, devido ao funcionamento do LOIC. Na ferramenta Hping3 é possível determinar o tamanho e quantidade de cada pacote de forma manual, enquanto no LOIC eles são predeterminados com intuito de simular um ataque real.

6 CONCLUSÕES

Este trabalho motivou-se pela necessidade de empresas de telecomunicações e de tecnologia da informação em buscar soluções de custo reduzido, escaláveis, de fácil manutenção e facilidade de gerenciamento das funções de rede e na ampliação de segurança das informações utilizando sistemas de detecção de intrusão.

A associação de IDS e NFV mostra-se um caminho promissor para reduzir investimentos e ajudar na segurança de dados. O modelo de rede atual baseada em *hardware* proprietário apresenta baixa escalabilidade, resultando em alto custo de implantação e manutenção para ambientes de rede, principalmente devido a limitações impostas por fabricantes. A NFV almeja solucionar alguns destes problemas, uma vez que ela permite eliminar essa dependência de *hardware* proprietário. A proposta de utilizar um IDS baseado em NFV, compará-la e analisá-la posteriormente permitiu a obtenção de informações para auxiliar na tomada de decisão em empresas de telecomunicações. Possibilitou, também, a compreensão de aspectos e características do funcionamento do sistema de virtualização por contêineres através do Docker.

Esta pesquisa permitiu identificar ser possível utilizar tecnologias de código aberto, sem necessidade de adquirir *software* proprietário para viabilizar a aplicação desta tecnologia. Os resultados obtidos mostraram que a NFV-IDS, em execução através de contêineres, é vantajosa no consumo de recursos computacionais, tanto no consumo de espaço de armazenamento em disco, quanto no consumo de memória. Ao observar o recurso de processamento durante o ataque TCP SYN Flood usando o Hping3, o consumo utilizando o contêiner foi superior ao da VM. Entretanto, no cenário UDP Flood, o consumo do recurso de processamento manteve-se similar. Já ao observar o recurso de processamento durante um ataque do tipo TCP utilizando o LOIC, observou-se que não gera um consumo de processamento alto, visto que o tamanho dos pacotes são similares a um usuário real, mesmo tendo enorme quantidade de requisições. Enquanto isso nos ataques do tipo UDP o consumo foi menor ainda, somente nos ataques do tipo HTTP e TCP o consumo de processamento foi alto.

A partir dos pontos analisados e dos resultados encontrados, através do comportamento da NFV-IDS juntamente com o servidor Web, possibilitou validar que uma NFV-IDS pode compartilhar os recursos do hospedeiro, entretanto é necessário limitar o consumo dos recursos disponíveis, caso a função receba algum ataque ela não aferará diretamente os demais serviços compartilhados no hospedeiro. Com os resultados obtidos da análise do consumo do recurso de memória, observou-se que um ataque do tipo HTTP que não seja detectado pode gerar gargalo. De maneira geral, os resultados indicam que apesar do NFV-IDS ter um longo caminho a percorrer até atingir maturidade, ele apresenta vantagens e pode otimizar o uso de recursos computacionais.

Esta pesquisa abre novos caminhos a serem seguidos acerca da NFV. Há um leque de possibilidades para novos estudos e experimentos. Alguns exemplos de caminhos a serem seguidos são a execução de testes em um ambiente corporativo, a comparação entre dois ou

mais sistemas de detecção de intrusão e a realização de testes utilizando IPS baseado em NFV. Entretanto, antes da execução de novos experimentos, será necessário aprimorar o levantamento de requisitos e adaptar a infraestrutura de rede usada para a aplicação da NFV-IDS. Com a implementação em um ambiente real ou com características próximas à realidade, será possível identificar problemas frequentes, dificuldades de aplicação e necessidade de recursos computacionais para uma execução adequada.

Outro caminho consiste na conversão da NFV-IDS em uma função de prevenção de intrusão (NFV-IPS). O *software* Snort também opera como IPS, o que viabilizaria o processo. Com isso, a função de rede não irá apenas detectar intrusões, mas poderá executar medidas prévias para bloqueios de ataques. Assim como este estudo trouxe informações que podem auxiliar na tomada de decisão, estes experimentos também poderiam ajudar a caracterizar melhor os benefícios deste trabalho para um cenário real. Algo a ser explorado seria a alocação dinâmica de recurso para o NFV-IDS como também adicionar uma carga maior de trabalho ao NFV-IDS, para se aproximar de um cenário real, com tráfego sendo adicionado para a análise.

REFERÊNCIAS

ALBIN, E.; Rowe, N. C. A realistic experimental comparison of the suricata and snort intrusion-detection systems. *In*: **2012 26th International Conference on Advanced Information Networking and Applications Workshops**. [*S.l.*: *s.n.*], 2012. p. 122–127.

ASHOOR, A. S.; GORE, S. Intrusion detection system (IDS) & intrusion prevention system (IPS): Case study. **Internatioanl Journal of Scientific & Engineering Research**, v. 2, 2012.

BASTA, A. *et al.* SDN and NFV dynamic operation of LTE EPC gateways for time-varying traffic patterns. *In*: SPRINGER. **International Conference on Mobile Networks and Management**. [*S.l.*], 2014. p. 63–76.

BRUMEN, B.; LEGVART, J. Performance analysis of two open source intrusion detection systems. *In*: IEEE. **2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [*S.l.*], 2016. p. 1387–1392.

CANONICAL. **Linux Containers**. Linux Containers, 2019. Disponível em: https://linuxcontainers.org/pt-br/lxc/introduction/. Acesso em: 10 de Agosto de 2019.

CAO, L. *et al.* NFV-VITAL: A framework for characterizing the performance of virtual network functions. *In*: IEEE. **IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)**. [*S.l.*], 2015. p. 93–99.

CERT. **Cartilha de Segurança para Internet**. São Paulo: Comitê Gestor da Internet no Brasil, 2012. Disponível em: https://cartilha.cert.br/ataques/. Acesso em: 05 Maio de 2019.

CHIOSI, M. *et al.* **Network Functions Virtualization (NFV), An Introduction, Beneits, Enablers, Challenges & Call for Action. SDN and OpenFlow World Congress, Darmstadt, Germany.(2012)**. 2012.

CISCO. **Cisco Annual Internet Report (2018–2023) White Paper**. Cisco, 2020. Disponível em: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Acesso em: 12 outubro de 2020.

CLOUDFLARE. **What is a Slowloris DDoS attack.** CLOUDFLARE, 2020. Disponível em: https://www.cloudflare.com/pt-br/learning/ddos/ddos-attack-tools/slowloris/. Acesso em: 02 de Agosto de 2021.

CLOUDFLARE. **Ataque de inundação de HTTP**. CLOUDFLARE, 2021. Disponível em: https://www.cloudflare.com/pt-br/learning/ddos/http-flood-ddos-attack/. Acesso em: 02 de Outubro de 2021.

COLELLA, A.; COLOMBINI, C. M. Amplification DDoS attacks: Emerging threats and defense strategies. *In*: SPRINGER. **International Conference on Availability, Reliability, and Security**. [*S.l.*], 2014. p. 298–310.

DEVMEDIA. **Virtualização de servidores.** 2019. Disponível em: https://www.devmedia.com.br/virtualizacao-de-servidores/30820. Acesso em: 29 Julho de 2020.

DOCKER, I. **Get Docker**. ESTADOS UNIDOS: Docker docs, 2013. Disponível em: https://docs.docker.com/get-docker/. Acesso em: 10 outubro de 2020.

ETSI. **Network Functions Virtualisation (NFV)**. Sophia-Antipolis, França.: ETSI ORG, 2019. Disponível em: https://www.etsi.org/technologies/nfv. Acesso em: 05 Novembro 2019.

ETSI, N. W. European telecommunications standards institute, industry specification groups (ISG) - NFV. 2012. Acesso em: 05 Novembro 2019.

ETSI, N. W. Etsi members around the world. 2020. Disponível em: https://www.etsi.org/events/16-membership. Acesso em: 05 Novembro 2019.

ETSI, N. W. p. . Network operator perspectives on industry progress. 2016. Disponível em: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf.

FADHLILLAH, A.; KARNA, N.; IRAWAN, A. Ids performance analysis using anomaly-based detection method for dos attack. *In*: **2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)**. [*S.l.*: *s.n.*], 2021. p. 18–22.

FAISAL, K. **A Beginner's Guide to Docker Container in NFV**. Telcocloud Bridge, 2016. Disponível em: https://telcocloudbridge.com/blog/beginners-guide-docker-container-nfv/. Acesso em: 10 outubro de 2020.

FORTINET. **A tecnologia em casa é usada como uma porta de entrada para a empresa**. 2020. Disponível em: https://www.fortinetthreatinsiderlat.com/pt/Q3-2020/CL/html/trends. Acesso em: 18 Outubro 2020.

GOLANG. **The Go programming language is an open source project to make programmers more productive.** Golang Wiki, 2016. Disponível em: https://github.com/golang/go/wiki. Acesso em: 10 outubro de 2020.

GUIMARÃES, V. T. *et al.* A survey on information visualization for network and service management. **IEEE Communications Surveys Tutorials**, v. 18, n. 1, p. 285–323, 2016.

HAN, B. *et al.* Network function virtualization: Challenges and opportunities for innovations. **IEEE Communications Magazine**, IEEE, v. 53, n. 2, p. 90–97, 2015.

HAT, I. R. **CONTAINERS: O que é Docker?** ESTADOS UNIDOS: Red Hat, 2020. Disponível em: https://www.redhat.com/pt-br/topics/containers/what-is-docker. Acesso em: 10 outubro de 2020.

HAWILO, H. *et al.* Nfv: state of the art, challenges, and implementation in next generation mobile networks (vEPC). **IEEE Network**, IEEE, v. 28, n. 6, p. 18–26, 2014.

INTEL. **The Benefits of Virtual CPE – Business User**. Solution brief, 2016. Disponível em: https://builders.intel.com/docs/networkbuilders/The-benefits-of-virtual-CPE-business-user.pdf. Acesso em: 06 Maio de 2019.

JAIN, A. *et al.* A comparison of SDN and NFV for re-designing the LTE packet core. *In*: IEEE. **2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)**. [*S.l.*], 2016. p. 74–80.

JOFFER, R. **2019: The Year in Review, Cyber Threats and Trends Report**. Neustar SOC, 2019. Disponível em: https://www.home.neustar/resources/whitepapers/2019-cyber-threats-and-trends-report. Acesso em: 15 de Outubro de 2020.

KLEIN, E. **5 open-source HIDS systems**. logz.io, 2020. Disponível em: https://logz.io/blog/open-source-hids/. Acesso em: 02 de Outubro de 2021.

Krishnaswamy, D. *et al.* An open NFV and cloud architectural framework for managing application virality behaviour. *In*: **IEEE Consumer Communications and Networking Conference (CCNC)**. [*S.l.*: *s.n.*], 2015. p. 746–754.

KUHRER, M. *et al.* Exit from hell? reducing the impact of amplification DDoS attacks. *In*: **USENIX Security Symposium 14**. [*S.l.*: *s.n.*], 2014. p. 111–125.

LI, Y.; CHEN, M. Software-defined network function virtualization: A survey. **IEEE Access**, IEEE, v. 3, p. 2542–2553, 2015.

LIAO, H.-J. *et al.* Intrusion detection system: A comprehensive review. **Journal of Network and Computer Applications**, v. 36, n. 1, p. 16 – 24, 2013. ISSN 1084-8045. Disponível em: http://www.sciencedirect.com/science/article/pii/S1084804512001944.

LIN, H. *et al.* Identifying application-layer ddos attacks based on request rhythm matrices. **IEEE Access**, v. 7, p. 164480–164491, 2019.

LIN, X. *et al.* A measurement study on Linux container security: Attacks and countermeasures. *In*: **Proceedings of the Computer Security Applications Conference**. [*S.l.*: *s.n.*], 2018. p. 418–429.

LOIC. **Low Orbit Ion Cannon (LOIC).** GITHUB, 2017. Disponível em: https://github.com/neweracracker/loic. Acesso em: 02 de Outubro de 2017.

LOPEZ, M. A. *et al.* Toward a monitoring and threat detection system based on stream processing as a virtual network function for big data. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, v. 31, n. 20, p. e5344, 2019.

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, 2016.

MOHAMMADI, R.; JAVIDAN, R.; CONTI, M. Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. **IEEE Transactions on Network and Service Management**, IEEE, v. 14, n. 2, p. 487–497, 2017.

Nikolov, D.; Kordev, I.; Stefanova, S. Concept for network intrusion detection system based on recurrent neural network classifier. *In*: **IEEE International Scientific Conference Electronics - ET**. [*S.l.*: *s.n.*], 2018. p. 1–4.

PARK, W.; AHN, S. Performance comparison and detection analysis in Snort and Suricata environment. **Wireless Personal Communications**, Springer, v. 94, n. 2, p. 241–252, 2017.

PATIL, R.; DUDEJA, H.; MODI, C. Designing an efficient security framework for detecting intrusions in virtual network of cloud computing. **Computers & Security**, Elsevier, v. 85, p. 402–422, 2019.

PETTERS, J. **IDS vs. IPS: What is the Difference?** ESTADOS UNIDOS: Varonis, 2020. Disponível em: https://blogvaronis2.wpengine.com/ids-vs-ips/. Acesso em: 10 outubro de 2020.

Qayyum, M.; Hamid, W.; Shah, M. A. Performance analysis of snort using network function virtualization. *In*: **2018 24th International Conference on Automation and Computing (ICAC)**. [*S.l.*: *s.n.*], 2018. p. 1–6.

ROSA, R.; BERTOLDO, C.; ROTHENBERG, C. E. Take your vnf to the gym: A testing framework for automated nfv performance benchmarking. **IEEE Communications Magazine**, v. 55, p. 110–117, 01 2017.

ROSA, R. *et al.* Network function virtualization: Perspectivas, realidades e desafios. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, 2014.

Rosa, R. V.; Rothenberg, C. E.; Szabo, R. Vbaas: VNF benchmark-as-a-service. *In*: **2015 Fourth European Workshop on Software Defined Networks**. [*S.l.*: *s.n.*], 2015. p. 79–84.

SIEKLIK, B.; MACFARLANE, R.; BUCHANAN, W. J. Evaluation of tftp DDoS amplification attack. **Computers Security**, v. 57, p. 67 – 92, 2016. ISSN 0167-4048. Disponível em: http://www.sciencedirect.com/science/article/pii/S0167404815001285.

SIMON, W.; MITNICK, K. **A Arte De Enganar: ATAQUES DE HACKERS: CONTROLANDO O FATOR HUMANO NA**. MAKRON, 2003. ISBN 9788534615167. Disponível em: https://books.google.com.br/books?id=B2yJPgAACAAJ.

SNORT. **Visão Geral do SNORT**. EUA: Snort ORG, 2019. Disponível em: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node3.html. Acesso em: 25 de Janeiro de 2020.

SOUSA, J. *et al.* Redes de petri híbridas diferenciais: aplicação na modelagem e no gerenciamento dinâmico de energia de redes de sensores sem fio. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, SciELO Brasil, v. 18, n. 3, p. 278–291, 2007.

SU, T.-J. *et al.* Attack detection of distributed denial of service based on splunk. *In*: **2016 International Conference on Advanced Materials for Science and Engineering (ICAMSE)**. [*S.l.*: *s.n.*], 2016. p. 397–400.

TANENBAUM, A. S. **Redes de Computadores**. 5. ed. [*S.l.*]: Campus, 2003.

TIWARI, M. *et al.* Intrusion detection system. **International Journal of Technical Research and Applications**, v. 5, p. 2320–8163, 04 2017.

UFRJ. **Denial of Service - Negação de Serviço (DoS).** UFRJ, 2015. Disponível em: https://www.gta.ufrj.br/grad/15_1/dos/pages/dos.html. Acesso em: 02 de Setembro de 2021.

van Cleeff, A.; Pieters, W.; Wieringa, R. J. Security implications of virtualization: A literature study. *In*: **International Conference on Computational Science and Engineering**. [*S.l.*: *s.n.*], 2009. v. 3, p. 353–358.

VMWARE. **VMWare Virtualizatio**. VMware Publisher, 2019. Disponível em: https://www.vmware.com/br/solutions/virtualization.html. Acesso em: 10 de Agosto de 2019.

WANG, G.; NG, T. E. The impact of virtualization on network performance of amazon EC2 data center. *In*: IEEE. **2010 Proceedings IEEE INFOCOM**. [*S.l.*], 2010. p. 1–9.

YUSOF, M. A. M.; ALI, F. H. M.; DARUS, M. Y. Detection and defense algorithms of different types of DDoS attacks. **International Journal of Engineering and Technology**, IACSIT Press, v. 9, n. 5, p. 410, 2017.

ZEEK. **What Is Zeek?** Zwwk Documentation, 2021. Disponível em: https://docs.zeek.org/en/current/about.html. Acesso em: 02 Novembro de 2021.

Zhou, Z. *et al.* The study on network intrusion detection system of snort. *In*: **International Conference on Networking and Digital Society**. [*S.l.*: *s.n.*], 2010. v. 2, p. 194–196.