UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

UNIDADE ACADÊMICA DE SERRA TALHADA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DmzVisor: Uma Abordagem para Segurança de Zonas Desmilitarizadas Corporativas em Redes Definidas por Software

Por

Valson da Silva Pereira

Serra Talhada,
Agosto/2018

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA DE SERRA TALHADA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

VALSON DA SILVA PEREIRA

DmzVisor: Uma Abordagem para Segurança de Zonas Desmilitarizadas Corporativas em Redes Definidas por Software

Projeto de Conclusão Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Unidade Acadêmica de Serra Talhada da Universidade Federal Rural de Pernambuco como requisito parcial à obtenção do grau de Bacharel.

Orientador: Prof. Me. Ygor Amaral Barbosa Leite de Sena

Serra Talhada,
Agosto/2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca da UAST, Serra Talhada - PE, Brasil.

P436d Pereira, Valson da Silva

DmzVisor: uma abordagem para segurança de zonas desmilitarizadas corporativas em redes definidas por software / Valson da Silva Pereira. – Serra Talhada, 2018.

85 f.: il.

Orientador: Ygor Amaral Barbosa Leite de Sena

Trabalho de Conclusão de Curso (Graduação em Bacharel em Sistema de Informação) – Universidade Federal Rural de Pernambuco. Unidade Acadêmica de Serra Talhada, 2018.

Inclui referências.

1. Segurança de redes. 2. Firewalls (Medidas de segurança para computadores). 3. Segurança da informação. I. Sena, Ygor Amaral Barbosa Leite de, orient. II. Título.

CDD 004

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA DE SERRA TALHADA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

VALSON DA SILVA PEREIRA

DmzVisor: Uma Abordagem para Segurança de Zonas Desmilitarizadas Corporativas em Redes Definidas por Software

Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Sistemas de Informação, defendida e aprovada por unanimidade em 29/08/2018 pela banca examinadora.

Banca Examinadora:

Prof. Me. Ygor Amaral Barbosa Leite de Sena
Orientador
Universidade Federal Rural de Pernambuco

Prof. Me. Héldon José Oliveira Albuquerque
Universidade Federal Rural de Pernambuco

Prof. Me. Hidelberg Oliveira Albuquerque
Universidade Federal Rural de Pernambuco

Dedico este trabalho ao Senhor Jesus de Nazaré,
por sempre cumprir suas promessas na minha vida e
à minha família por sempre me apoiar. Dedico também
ao meu orientador Ygor Amaral e a todos professores
e amigos da UFRPE/UAST.

AGRADECIMENTOS

Agradeço em primeiro lugar ao Senhor Jesus, autor e consumador da minha fé, que permitiu e me deu forças para chegar até aqui, à minha família que sempre me apoiou em minhas decisões.

Agradeço ao meu professor orientador Ygor Amaral Barbosa Leite de Sena, por sua dedicação, compreensão e compromisso para comigo, à professora Lilian Oliveira Ramires, pelo seu auxílio durante a disciplina de Pré Projeto de Conclusão de Curso e aos demais professores e amigos do curso de Bacharelado em Sistemas de Informação da UFRPE/UAST que sempre me apoiaram e contribuíram na minha formação acadêmica.

"E, se algum de vós tem falta de sabedoria, peça-a a Deus, que a todos dá liberalmente, e o não lança em rosto, e ser-lhe-á dada".

(Bíblia Sagrada, Tiago 1, 5)

RESUMO

O avanço na utilização da rede mundial de computadores nas últimas décadas fez a internet se tornar umas das principais ferramentas de comunicação ao redor do mundo. Entretanto, quanto mais as empresas fornecem serviços via web, mais tendem a expor de alguma forma suas informações importantes na rede. Dessa forma, é necessário que haja uma preocupação com relação à segurança desses serviços. Uma das recomendações para segurança de dados das organizações que fornecem serviços externos, é utilizar uma zona desmilitarizada (DMZ), que consiste no uso de um ou mais *firewalls* em sua configuração, porém pode ter um custo financeiro considerável para a empresa. Além disso, muitos equipamentos de *firewall* são fornecidos como uma caixa preta de soluções proprietárias com softwares embarcados pelo fabricante, de modo que são poucos flexíveis do ponto de vista de personalização. Todavia, através do paradigma de redes definidas por software (SDN) juntamente com o protocolo *OpenFlow*, que permite flexibilização com relação ao desenvolvimento de soluções de software para redes, tem-se o benefício de oferecer um produto de software como uma alternativa de baixo custo e personalizável. Por meio de SDN, a aplicação pode ser implementada em linguagem de programação de alto nível e fazendo uso de ferramentas gratuitas *open source*, de modo que, também possibilita a manutenibilidade do software para a devida adequação às necessidades do cliente. Em virtude disso, o objetivo geral deste trabalho foi desenvolver um *firewall* SDN corporativo como uma alternativa de baixo custo, utilizando ferramentas *open source*, que atua como um filtro de pacotes e isola o tráfego entre a rede local e a zona desmilitarizada, através de regras SDN e da implementação segura de mensagens de protocolos como o *Dynamic Host Configuration Protocol* (DHCP) e *Address Resolution Protocol* (ARP). Além de ter sido desenvolvido mecanismos de filtragem de pacotes para as camadas de rede e de transporte e oferecer mais segurança por meio de isolamento de redes, também foi desenvolvida uma interface gráfica web amigável, na qual o administrador é capaz de gerenciar o software controlador da rede criando regras de *firewall* em alto nível, não sendo necessário o usuário ter conhecimento sobre o protocolo *OpenFlow*. A avaliação da proposta foi composta por 02 cenários utilizando 6 máquinas virtualizadas pelo o VirtualBox. A proposta demonstrou que tanto suas regras de segurança ARP e DHCP, quanto às regras de *firewall* são eficazes nas redes que são protegidas pelo protótipo proposto nesse trabalho, conseguindo evitar ataques do homem-do-meio, falsificação de endereços IP e MAC, bem como realizar a filtragem e encaminhamento de pacotes da camada de rede e transporte de forma segura.

Palavras-chave: Segurança de Redes, Redes Definidas por Software, *OpenFlow*, *Firewall*, Zona Desmilitarizada, DHCP, ARP.

ABSTRACT

The advance in the use of the world-wide network of computers in the last decades made the internet become one of the main communication tools around the world. However, the more companies provide services via the web, the more they tend to expose their important information in the network in some way. In this way, there is a need for concern about the safety of these services. One of the recommendations for data security of organizations providing external services is to use a demilitarized zone (DMZ), which consists of the use of one or more firewalls in their configuration, but can have a considerable financial cost for the company. In addition, many firewall equipment is provided as a black box of proprietary solutions with embedded software by the manufacturer, so they are few flexible from a personalization point of view. However, through the Software-Defined Networking (SDN) paradigm along with the OpenFlow protocol, which allows for flexibility in developing software solutions for networks, one has the benefit of offering a software product as a low-cost alternative and customizable. Through the SDN, the application can be implemented in a high-level programming language and making use of free open source tools, so that it also enables the maintenance of the software to suit the needs of the client. As a result, the overall goal of this work was to develop a corporate SDN firewall as a low-cost alternative, using open source tools, which acts as a packet filter and isolates traffic between the local network and the demilitarized zone , through SDN rules and the secure implementation of protocol messages such as Dynamic Host Configuration Protocol (DHCP) and Address Resolution Protocol (ARP). In addition to the development of packet filtering mechanisms for the network and transport layers and to provide more security through network isolation, a friendly web graphic user interface has also been developed in which the administrator is able to manage the creating high-level firewall rules, so the user does not need to be aware of the OpenFlow protocol. The evaluation of the proposal was composed by 02 scenarios using 6 machines virtualized by VirtualBox. The proposal demonstrated that both its ARP and DHCP security rules and the firewall rules are effective in the networks that are protected by the prototype proposed in this work, being able to avoid man in the middle attacks, IP and MAC address spoof, as well as perform filtering and routing of network and transport layer packets securely.

Keywords: Network Security, Software-Defined Networking, OpenFlow, Firewall, Delimitarized Zone, DHCP, ARP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Programming Interface*
ARP	*Address Resolution Protocol*
BSD	*Berkeley Software Distribution*
CPqD	Centro de Pesquisa e Desenvolvimento em Telecomunicações
DHCP	*Dynamic Host Configuration Protocol*
DMZ	*Demilitarized Zone*
DNS	*Domain Name System*
EPL	*Eclipse Public License*
FTP	*File Transfer Protocol*
GPLv2	*General Public License version 2*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
IPv4	*Internet Protocol version 4*
LAN	*Local Area Network*
MAC	*Media Access Control*
MIT	*Massachusetts Institute of Technology*
NAT	*Network Address Translation*

NBR	Norma Brasileira Regulamentadora
NTT	*Nippon Telegraph and Telephone*
ODL	*OpenDayLight*
ONF	*Open Networking Foundation*
ONOS	*Open Networking Operating System*
OVS	*Open vSwitch*
RFC	*Request for Comments*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
SDN	*Software-defined Networking*
SFTP	*Secure File Transfer Protocol*
SGBD	Sistema de Gerenciamento de Banco de Dados

SUMÁRIO

1 Introdução

Neste capítulo inicial será apresentado o contexto histórico e as principais motivações deste trabalho. O capítulo está estruturado da seguinte forma: na Seção 1.1 é apresentado brevemente o contexto geral do trabalho. A motivação e justificativa estão expostas na Seção 1.2. Na Seção 1.3 contém a descrição dos objetivos geral e específicos desta monografia. E por fim, na Seção 1.4 é detalhado como o trabalho está organizado.

1.1 Contexto geral

O avanço na utilização da rede mundial de computadores nas últimas décadas fez a internet se tornar umas das principais ferramentas de comunicação ao redor do mundo. Milhões de usuários dos mais variados perfis estão conectados na web, desde usuários domésticos até grandes empresas. Diante dessa realidade a utilização da internet passou a ser parte estratégica nos mais diversos ramos de negócio, portanto, deixou de ser algo opcional no meio corporativo (STALLINGS, 2011), tornando-se cada vez mais comum as empresas oferecerem serviços online para seus clientes, aliando comodidade às necessidades de seu público.

Entretanto, quanto mais as empresas fornecem serviços via web, mais tendem a expor de alguma forma informações importantes na rede. Portanto, é necessário que haja uma preocupação com relação à segurança desses serviços que estão em contato com a rede externa da organização, filtrando o conteúdo de quem faz uso legítimo do serviço de quem está mal intencionado (TANENBAUM; WETHERALL, 2011), para preservar os ativos de informações da corporação.

Uma das soluções tradicionais para filtragem de conteúdo consiste no *firewall*, que baseia-se em regras de acesso, onde permite ou nega a passagem do tráfego entre as redes interna e externa da organização de maneira recíproca.

De acordo com Stallings (2011), as organizações que necessitam oferecer serviços ao público externo, um ou mais *firewalls* são utilizados para sua proteção, de modo que na região que se forma entre esses *firewalls* estão os serviços externos acessíveis ao público, essa região é denominada de Zona Desmilitarizada (DMZ).

A implantação da DMZ na rede é uma solução bastante interessante e eficaz, porém de acordo com Jiang et al. (2017), apesar de prover mais segurança em sistemas críticos, a utilização de mais de um *firewall* pode tornar mais caro e complexo o gerenciamento da rede.

Todavia, no final da década passada surgiu o conceito de Redes Definidas por Software (SDN), que consiste em um paradigma no qual oferece programabilidade à rede de computadores tornando seu núcleo mais simples, por meio da separação do plano de dados e plano de controle dos comutadores. Com isso é possível prover soluções flexíveis, personalizadas com gerenciamento centralizado e segurança a um baixo custo (MCKEOWN et al., 2008).

Diante desse contexto, neste trabalho será implementado um *firewall* corporativo baseado em redes definidas por software como uma alternativa de baixo custo. Este *firewall* atuará como filtro de pacotes que tem a finalidade de isolar o tráfego da rede interna com a DMZ, através de regras SDN e da implementação segura de mensagens de protocolos de rede como o *Dynamic Host Configuration Protocol* (DHCP) e *Address Resolution Protocol* (ARP). Será desenvolvido também um sistema web com interface gráfica de gerenciamento da aplicação para o administrador de segurança da rede.

1.2 Motivação e Justificativa

A necessidade de ser implementada uma DMZ em organizações que fornecem serviços ao público externo é fundamental para segurança de seus dados. No entanto, a adoção de uma rede DMZ normalmente requer um investimento considerável em equipamentos de *firewall*, o que pode ser uma barreira na aceitação desta tecnologia, devido o custo para a organização. Além disso, esses equipamentos são fornecidos como uma caixa preta de soluções proprietárias com softwares embarcados pelo fabricante, de modo que são poucos flexíveis do ponto de vista de personalização de soluções que se adequem ao cliente (SONG, 2013), além de dificultar a manutenção desse software, que consequentemente se torna dependente do fabricante.

Entretanto, através do desenvolvimento de um *firewall* baseado no paradigma de rede SDN, tem-se o benefício de oferecer um produto de software como uma alternativa de baixo custo e personalizada para a segurança das informações da organização, pois pode ser implementado em linguagem de programação de alto nível e fazendo uso de ferramentas gratuitas *open source*, de modo que, também possibilita a manutenibilidade do software para a devida adequação às necessidades do cliente. Além do mais, um *firewall* SDN traz em si as vantagens inerente às

redes definidas por software, que de acordo com Guedes et al. (2012), “a visão global da rede oferecida pelo controlador SDN permite que regras de acesso sejam desenvolvidas com base em informações abrangentes e não apenas o que seria possível com o uso de um *firewall* em um enlace específico da rede”.

Deste modo, com o software proposto neste trabalho, o administrador de rede terá a vantagem de realizar o gerenciamento por meio de um sistema web amigável, indicando quais dispositivos estarão na DMZ e na área da rede local (LAN), isolando o tráfego entre elas, além de definir regras de filtragem de conteúdo para ambas as redes.

1.3 Objetivos

Será apresentado a seguir os objetivos geral e específicos deste trabalho.

1.3.1 Objetivo Geral

O objetivo geral é desenvolver um *firewall* SDN corporativo como uma alternativa de baixo custo, utilizando ferramentas *open source*, que atua como um filtro de pacotes e isola o tráfego entre a rede local e a zona desmilitarizada, para auxiliar na segurança da rede.

1.3.2 Objetivos Específicos

- Desenvolver mecanismos de segurança que proporcionarão o isolamento do tráfego de rede via SDN;
- Desenvolver uma aplicação SDN para empregar regras de *firewall*;
- Elaborar um sistema web com uma interface gráfica amigável para a gerência do *firewall*;
- Avaliar o protótipo através do teste de eficácia das regras de segurança e isolamento das redes.

1.4 Organização do Trabalho

Esta monografia está organizada da seguinte forma: no Capítulo 2 é apresentada a fundamentação teórica deste trabalho, através da discussão dos temas de segurança de redes explorando sua importância e conceitos abordados nesse trabalho, tais como, *firewalls* e DMZ. Logo em seguida é apresentado a definição de redes definidas por software, assim como suas vantagens e estrutura deste paradigma, além dos seus principais componentes, como o protocolo *OpenFlow*, controladores e *softwares switches*. Por fim é discutido sobre os protocolos DHCP e ARP, além da apresentação de artigos publicados de trabalhos relacionados à esta abordagem. O Capítulo 3 traz a descrição da proposta deste trabalho, por meio de seus métodos e materiais utilizados. O Capítulo 4 são apresentados a forma de avaliação e os resultados obtidos. Por fim, o Capítulo 5 denota a conclusão desta monografia juntamente com os trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo será discutido os conceitos no qual este trabalho está fundamentado. O capítulo está estruturado da seguinte forma: na Seção 2.1 é descrita a relevância da área de segurança de redes nas corporações, além da importância do firewall e da DMZ. A abordagem sobre redes definidas por software, bem como suas vantagens e componentes é denotada a partir da Seção 2.2. Na Seção 2.3 é apresentado o protocolo DHCP. A Seção 2.4 trás uma explicação sobre o protocolo ARP. Os trabalhos relacionados são apresentados na Seção 2.5 e por fim na Seção 2.6 é exposto um resumo do capítulo.

2.1 Segurança de redes

Ao longo dos últimos anos com o crescimento da utilização da internet, a segurança das redes de computadores foi uma das principais preocupações das organizações. De acordo com Kurose e Ross (2013), a maioria das empresas estão conectadas à internet e podem ser vítimas de ataques cibernéticos. Sendo assim, a área corporativa têm atentado para a segurança da informação, pois proteger seus ativos de informações requer uma atenção especial. Diante disso, é perceptível que a segurança de redes é um requisito estratégico fundamental para o meio corporativo.

Existem três princípios que são fundamentais para a segurança tanto de dados como também para serviços informatizados (STALLINGS, 2011):

- **Confidencialidade**: princípio relacionado ao controle de acesso e divulgação de informações sensíveis da organização, com a finalidade de proteção de privacidade;
- **Integridade**: visa a prevenção contra a modificação ou destruição não autorizada de informações;
- **Disponibilidade**: garantia de acesso e utilização da informação confiável por entidades legítimas.

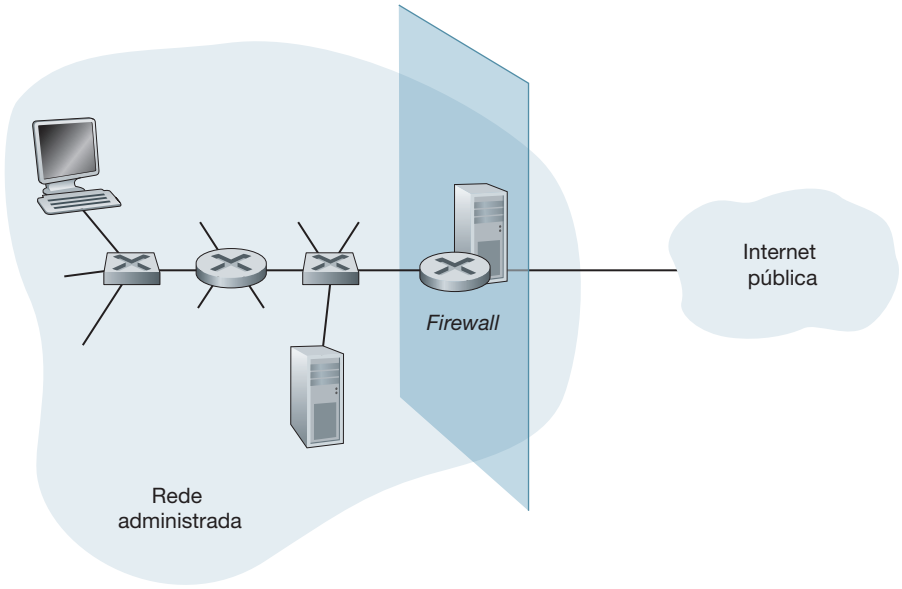
Em ambientes corporativos, no qual os ativos de informações são bens importantes para

as organizações, se faz necessário que estes princípios de segurança estejam consolidados. Para Nakamura e Geus (2007), a segurança dos dados das organizações tem que abranger todos os pontos possíveis, por se tratar de uma defesa mais complexa, no qual um único ponto de falha explorado pode inutilizar os demais pontos que foram protegidos. Sendo assim, dentre outros cuidados que devem ser tomados, está incluso mecanismos de segurança, como por exemplo, o *firewall*, que segundo Kurose e Ross (2013), auxiliam no impedimento de ataques à rede. A seguir será descrito como os *firewalls* auxiliam na segurança operacional da rede.

2.1.1 *Firewalls*

Os *firewalls* são equipamentos de hardware e software que atuam na segurança entre a rede interna da organização e a rede externa, que geralmente é a internet, como está descrito na Figura 2.1. De acordo com Tanenbaum e Wetherall (2011), os *firewalls* em sua abordagem tradicional, filtram pacotes que entram e sai da rede, obedecendo critérios de filtragem definidos pelo administrador.

Figura 2.1 – Arquitetura de rede com *firewall*



Fonte: (KUROSE; ROSS, 2013).

Os critérios de segurança do *firewall* tanto descrevem quais são os tipos de acessos legítimos aos recursos disponibilizados na rede, como também os que podem ser considerados uma tentativa de acesso fraudulenta, por meio do monitoramento das portas de protocolos de origem e destino.

Um dos tipos elementares de *firewall* é o filtro de pacotes (*packet filtering*). Esse tipo de *firewall* desempenha a importante função de rotear pacotes entre *hosts* de diferentes redes seletivamente, permitindo a passagem do pacote ou rejeitando o tráfego do mesmo (CHAPMAN; ZWICKY; RUSSELL, 2001). A análise dos pacotes é feita apenas em informações dos campos de cabeçalhos de protocolos das camadas de rede e transporte, conforme demonstra a Tabela 2.1. De acordo com Komar, Beekelaar e Wettern (2003), pode-se implementar um *firewall* filtro de pacotes para analisar entre outros, os seguintes campos de cabeçalho:

- **endereço IP de origem**: endereço IP do *host* remetente do pacote, mas lembrando que, não necessariamente reflete no ip original desse *host*, pois o mesmo pode utilizar um mecanismo de tradutor de endereços tal como o *Network Address Translation* (NAT), que traduz endereços privados de rede em endereços públicos válidos na internet;
- **endereço IP de destino**: endereço IP do *host* destinatário;
- **id do protocolo superior**: campo do cabeçalho do protocolo IP que indica qual protocolo é usado na camada superior para tratar a carga útil (*payload*) que o datagrama IP carrega. Por exemplo: um datagrama ICMP tem o id de valor 1, um segmento TCP possui id número 6 e um datagrama UDP é identificado pelo número 17;
- **número de porta *Transmission Control Protocol* (TCP) ou *User Datagram Protocol* (UDP)**: reflete o número da porta TCP/UDP na qual o protocolo da camada de aplicação atende as requisições, como por exemplo, *Hypertext Transfer Protocol* HTTP atende requisições na porta TCP 80, já o DHCP atende as requisições do cliente na porta UDP 68 e no lado do servidor escuta as requisições na porta UDP 67. Com relação ao TCP, também pode ser implementado verificações para o estado de estabelecimento de conexão, oferecendo suporte a inspeção de *flags* TCP, como por exemplo a *flag* TCP SYN, que indica o início de uma conexão entre dois hosts;
- **tipo de mensagem *Internet Control Message Protocol* (ICMP)**: suporta tipos de mensagens ICMP tais como: ICMP ECHO REQUEST, que consiste na mensagem de uma requisição ICMP (pode ser facilmente utilizada através de envio de um ping para um destinatário) e ICMP ECHO REPLY (resposta a um ping).

Tabela 2.1 – Regras de um *firewall* filtro de pacotes.

Interface de chegada	IP de Origem	IP de Destino	Protocolo	Porta	Ação
2	*	*	TCP	21	Recusa
2	*	*	TCP	23	Recusa
1	*	128.5.0.2/16	TCP	80	Permite
2	*	*	UDP	43	Recusa
1	*	128.5.0.2/16	ICMP	-	Recusa

Fonte: adaptada de (COMER, 2017).

Além do tipo básico de *firewall* que se comporta como filtro de pacotes, para Tanenbaum e Wetherall (2011), existem no mercado mais dois tipos importantes, que são:

- ***Firewalls* em estado de conexão**: utilizam campos do cabeçalho TCP/IP para tratar das conexões, no qual é possível estabelecer regras unilaterais que permitem a conexão entre dois computadores, desde que obrigatoriamente a conexão seja iniciada por um computador em específico;
- ***Gateways* em nível de aplicação**: analisa os pacotes mais a fundo do cabeçalho TCP/IP na camada de aplicação. Por meio desta análise é possível saber para quais tipos de tarefas os protocolos de aplicação estão sendo utilizado na rede.

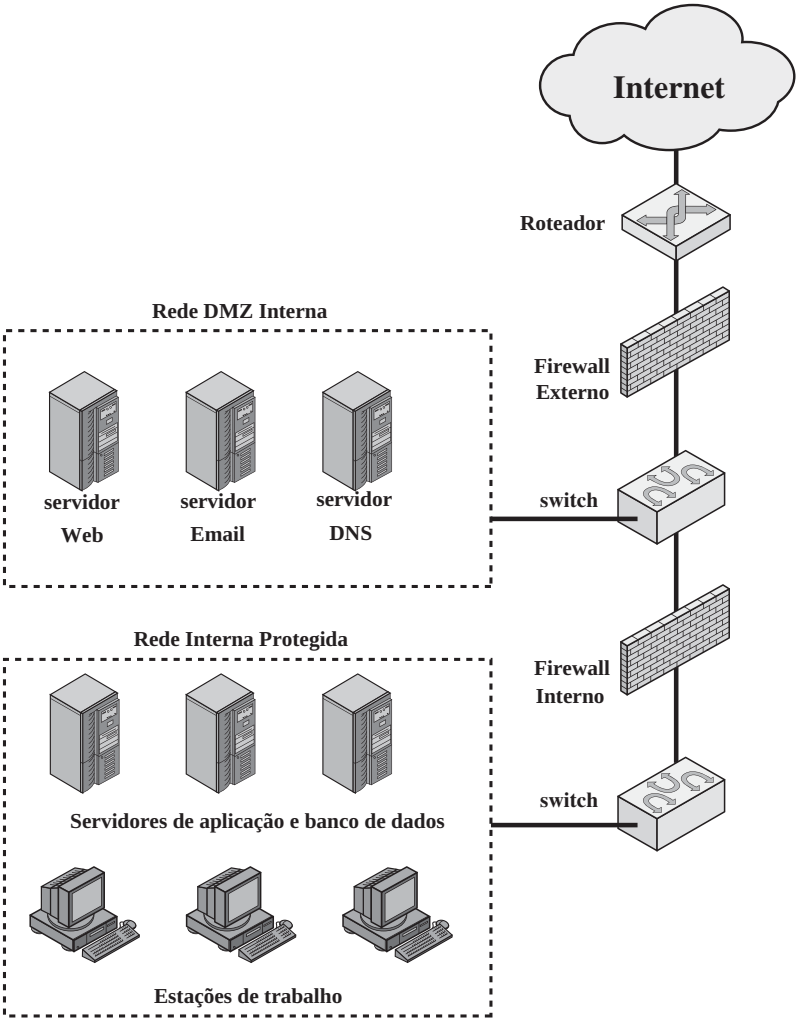
A presença de um *firewall* filtrando conexões é essencial para a segurança de uma rede organizacional. De acordo com Comer (2016, p. 456), "*firewalls* são a mais importante ferramenta de segurança para tratar de conexões de rede entre duas organizações que não confiam uma na outra". Uma das maneiras de organizar *firewalls* para obter maior segurança é através de uma DMZ, que será discutido a seguir na Subseção 2.1.2.

2.1.2 *Demilitarized zone* (DMZ)

A *delimitarized zone* (DMZ) ou zona desmilitarizada, é uma área de segurança de perímetro protegida por um *firewall* que faz contato com a rede externa, ou seja, a internet. A DMZ comporta dispositivos que oferecem serviços a ser acessados por um público externo da organização, como por exemplo, serviços web, email e DNS (STALLINGS, 2011), conforme descrito na Figura 2.2.

A Norma Brasileira Regulamentadora ISO/IEC 27002:2013 (NBR ISO/IEC 27002:2013), que trata de indicar normas e diretrizes para a segurança da informação das organizações (ABNT,

Figura 2.2 – Exemplo de configuração dos *firewalls* em uma DMZ.



Fonte: (STALLINGS, 2011).

2013), recomenda que as redes de uma organização sejam segregadas em diferentes domínios de acordo com seus níveis de confiança, como por exemplo, tipos de serviços oferecidos e permissões de acesso por tipo de usuário.

Dentre os mecanismos para segregar redes pode-se adotar a criação de uma DMZ, visto que, seu propósito é separar a rede interna da externa, pois na rede externa há um maior risco de segurança, tendo em vista que, está provendo serviços em contato diretamente com a internet. Segundo Torres (2014), se algum atacante conseguir obter acesso a algum dispositivo localizado na DMZ, ele não conseguirá obter acesso à rede interna, devido ao *firewall* instalado no ponto de entrada dessa rede, na qual estão os dados mais importantes da organização. Dessa forma, a implantação da DMZ em empresas que necessitam fornecer serviços a serem usados externamente, é uma segurança adicional importante, que pode evitar danos maiores em caso de um ataque hacker.

2.2 *Software-Defined Networking* (SDN)

O núcleo das redes de computadores tradicionais é formada por diversos tipos de dispositivos, dentre ele estão, os roteadores, comutadores, *firewalls* e etc. Entretanto, nesses equipamentos estão implementados inúmeros protocolos, tornando complexo o gerenciamento de uma rede (NUNES et al., 2014). Outro problema presente nas redes tradicionais, é a dificuldade de desenvolver e adotar novos protocolos, devido essa complexidade em seu núcleo, inibindo a inovação, o que levou a pesquisadores classificá-la como uma infraestrutura "ossificada", do ponto de vista para testar resultados sobre pesquisas científicas na área de rede de computadores (MCKEOWN et al., 2008). Além desses impasses apresentados no paradigma de redes tradicional, vale destacar também as seguintes limitações (ONF, 2012a):

- **políticas inconsistentes**: para implementar políticas, como por exemplo, controle de acesso, segurança e QoS em grandes redes, pode requerer que os responsáveis por essa implementação configure milhares de dispositivos e mecanismos espalhados por toda a rede, o que pode ocasionar inconsistências nas políticas de segurança da rede, tornando-a vulnerável e desencadeando diversas consequências negativas para a organização;
- **incapacidade de escalabilidade**: a medida que grandes redes, como por exemplo, redes que suportam data centers crescem, a rede também tende a crescer juntamente com sua complexidade de ter que adicionar centenas de milhares de dispositivos para configurar e gerenciar manualmente, o que torna a escalabilidade da rede complicada;
- **dependência do fornecedor**: a busca das empresas e operadoras de estar sempre atualizadas e de acordo com as demandas do mercado, juntamente com sua dinâmica de suprir rapidamente as necessidades de seus usuários esbarra na dependência do fornecedor. Os fornecedores oferecem produtos com ciclos de atualizações de seus equipamentos que variam anos, além de não fornecer uma API aberta para as empresas adequarem os equipamentos às suas necessidades.

Diante deste cenário, no final da década passada, a comunidade científica apresentou um novo paradigma de rede chamado de *Software-Defined Networking* (SDN), ou simplesmente, redes definidas por software. O paradigma SDN retira o plano de controle dos comutadores, que é o componente responsável pela gerência do *switch*, deixando somente o plano de dados que tem a função de conduzir os pacotes. A partir do desacoplamento desses componentes, o

núcleo da rede se torna mais simples, com o controle lógico da inteligência localizado em um componente de software centralizado e programado fora dos dispositivos do núcleo (NADEAU; GRAY, 2013).

A partir dessa inteligência centralizada fornecida por redes SDN, a tarefa de gerência e pesquisa de redes é facilitada, uma vez que, é possível ter uma visão geral do que acontece em toda a rede, além de flexibilidade e total programabilidade, sem a necessidade de alterar as configurações da rede em baixo nível (CASADO et al., 2012b), pois a programação da configuração de rede é realizada em alto nível por meio de uma *Application Programming Interface* (API) de comunicação padronizada. Com a utilização de uma API de acesso à rede padronizada, como SDN propõe, é possível se ter uma rede com dispositivos de diferentes fornecedores (UNDERDAHL; KINGHORN, 2015), recebendo instruções de uma mesma aplicação SDN, o que permite uma independência de determinado tipo ou marca de equipamentos.

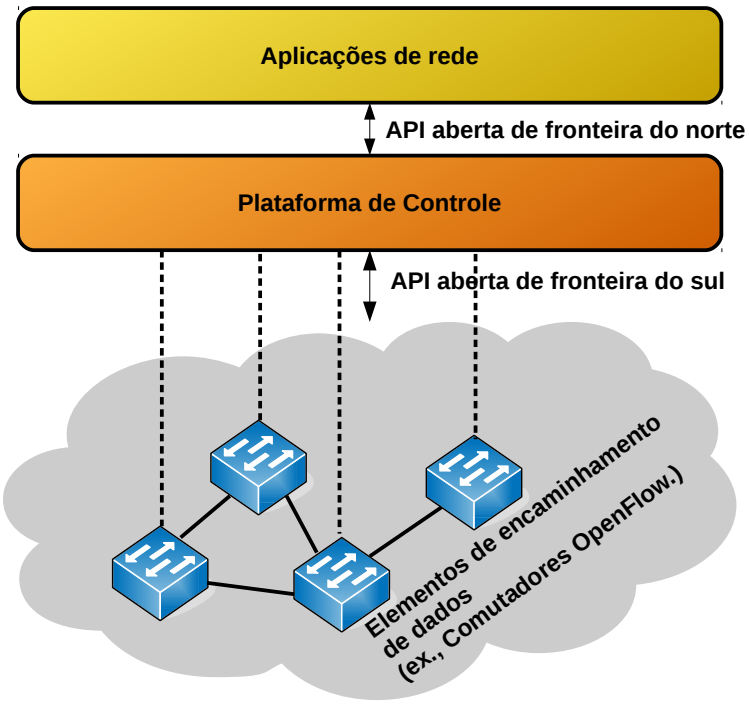
Em razão desses benefícios, as redes definidas por software, segundo Stallings (2015), possuem escalabilidade sob demanda e segurança integrada, uma vez que, oferece suporte ao crescimento da rede de acordo com mudanças de requisitos, além disso, facilita o desenvolvimento de soluções seguras integradas à rede como um serviço. Desse modo as políticas de segurança podem ser implementadas na medida que as funcionalidades da rede são desenvolvidas.

A arquitetura básica de uma rede SDN é mostrada na Figura 2.3. A arquitetura de uma rede SDN é composta por (KREUTZ et al., 2015):

- **Elementos de encaminhamento de dados**: comutadores de rede operando apenas com o plano de dados, sem o plano de controle;
- **API aberta de fronteira do sul**: define o protocolo de comunicação entre os elementos de encaminhamento de dados e a plataforma de controle, que dentre os mais utilizados nesta API está o protocolo de comunicação *OpenFlow* (BRAUN; MENTH, 2014);
- **Plataforma de controle ou controlador SDN**: entidade que contém a inteligência da rede centralizada, sendo assim o "cérebro da rede", pelo qual são programados os elementos de encaminhamento de dados;
- **API aberta de fronteira do norte**: responsável por oferecer suporte aos desenvolvedores de aplicações da rede abstraindo para um alto nível as instruções de baixo nível, usada pela a API aberta de fronteira do sul.;
- **Aplicações de rede**: soluções de software desenvolvidas para operar sobre uma rede SDN.

Devido as suas utilidades e arquitetura simplificada, as redes definidas por software

Figura 2.3 – Visão simplificada de uma arquitetura SDN.



Fonte: (KREUTZ et al., 2015).

têm obtido destaque, tanto no meio acadêmico, como também no meio comercial. Aplicações SDN têm sido adotadas para facilitar atividades das mais diversas finalidades inerentes à redes de computadores. Dessa forma, Guedes et al. (2012) ressalta a aplicação deste paradigma para as seguintes tarefas:

- **controle de acesso**: em virtude das redes definidas por software permitir uma perspectiva completa da rede, aplicações de controle de acesso, tais como, *firewalls* a exemplo da proposta desta monografia e o Ethane (CASADO et al., 2007) que é um sistema precursor de controle de acesso distribuído envolvendo SDN, podem ser implementadas para gerenciar as políticas de segurança da rede em sua totalidade;
- **gerência de recursos da rede**: simplificar o gerenciamento de redes foi uma das motivações para o surgimento das redes SDN, grandes empresas de tecnologia têm investido em pesquisa e desenvolvimento de sistemas neste sentido, inclusive em suas redes de larga escala, exemplo disso são os projetos B4 (JAIN et al., 2013) e Espresso (YAP et al., 2017) do Google e o Edge Fabric (SCHLINKER et al., 2017) do Facebook;
- **gerenciamento de redes domiciliares**: com a popularização da internet, sobretudo da chamada internet das coisas, as residências tendem a possuir diversos tipos de dispositivos

de borda (*smartphones*, *tablets*, tv's, pc's, etc.) conectados à grande rede, porém gerenciar e configurar comutadores de rede ainda é um desafio para usuários comuns. Soluções fomentadas neste propósito, como a aplicação HomeNetRescue (ALVES et al., 2018), oferecem serviços baseado em SDN para a comodidade do usuário, que consiste em detectar, resolver falhas e configurar equipamentos em redes domésticas;

- **eficiência de consumo de energia em *data centers***: em face de grandes *data centers* consumir bastante recurso e cada vez mais há uma preocupação ambiental com relação ao consumo energético, as redes definidas por software podem ser utilizadas para provimento de consumo eficiente de energia dos dispositivos destes grandes parques tecnológicos. A partir do estado global da rede, é possível identificar equipamentos da rede que estão ociosos, por meio de estatísticas do tráfego, e assim acionar um desligamento desses equipamentos ou fazê-los trabalhar conforme a demanda de recurso, ou então, com sua capacidade reduzida. Trabalhos propostos para esta finalidade, como o precursor SDN Elastic Tree (HELLER et al., 2010) e Conterato (2016), obtiveram resultados relevantes.

Conforme estes aspectos apresentados, percebe-se os benefícios relevantes e a ampla aplicabilidade que as redes definidas por software podem oferecer em áreas distintas. A seguir na Subseção 2.2.1, será apresentado o protocolo de API aberta de fronteira do sul, o *OpenFlow*, bem como sua significância para as redes SDN e suas aplicações.

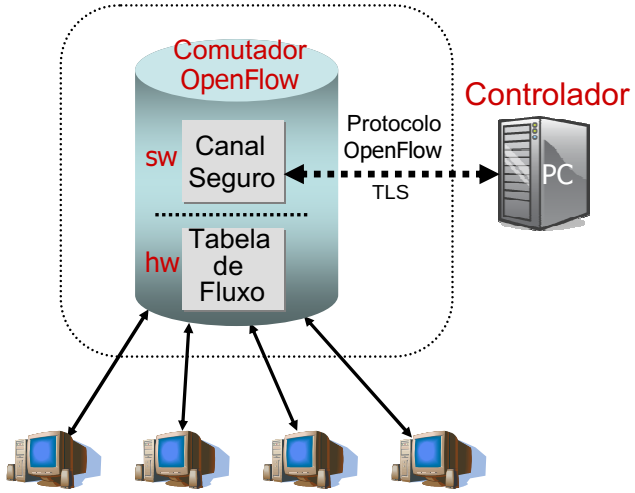
2.2.1 Protocolo *OpenFlow*

O *OpenFlow* (OF) consiste em um protocolo padronizado e aberto desenvolvido por pesquisadores da Universidade de Stanford que possibilita a programação de tabelas de fluxos em diferentes roteadores e comutadores na rede (MCKEOWN et al., 2008). Este protocolo permite a centralização da gerência de uma rede de computadores, é por meio dele, que o conceito de SDN é implementado de fato.

Além de possibilitar o gerenciamento de múltiplos comutadores de rede utilizando apenas um único controlador SDN, o *OpenFlow* também proporciona a análise de tráfego através de software, facilitando as tarefas corriqueiras de um administrador de redes (LARA; KOLASANI; RAMAMURTHY, 2014). Em virtude destas vantagens, o *OpenFlow* se tornou o principal protocolo utilizado na API aberta de fronteira do sul em redes definidas por software.

Podemos observar em sua arquitetura descrita na Figura 2.4, que o *OpenFlow* é composto por (MCKEOWN et al., 2008):

Figura 2.4 – Arquitetura *OpenFlow*



Fonte: (MCKEOWN et al., 2008).

- **Protocolo *OpenFlow***: protocolo que utiliza como proteção dos dados o *Transport Layer Security* (TLS) e que tem a função de operar a comunicação entre o controlador SDN e o comutador.
- **Canal seguro**: conecta o comutador ao controlador SDN, permitindo envio de pacotes e comandos entre ambos;
- **Tabela de fluxo**: tabela composta por fluxos de entrada gerenciadas pelo controlador SDN via canal seguro, que possui uma ação associada a cada fluxo, indicando como o comutador deve processar este fluxo por meio de comparação a nível de hardware no plano de dados.

O protocolo continua evoluindo e agregando novas funcionalidades a cada versão de lançamento. O *OpenFlow* possui cinco versões até o momento da produção deste trabalho, conforme denota a Tabela 2.2.

Tabela 2.2 – Evolução das principais funcionalidades do *OpenFlow* ao longo das versões.

Funções presentes	**Versões *OpenFlow***					
	1.0	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**
Múltiplas tabelas de fluxo		X	X	X	X	X
Suporte a IPV6			X	X	X	X
Múltiplos controladores simultâneos			X	X	X	X
Suporte a interface de fibra ótica					X	X

Fonte: adaptada (JUNIOR, 2016).

2.2.1.1 Componentes da Tabela de Fluxo

A tabela de fluxo desempenha um papel crucial numa rede SDN *OpenFlow*. As tomadas de decisões relacionadas ao destino de pacotes que ingressam na rede, são determinadas a partir de regras, que são baseadas em informações de campos de cabeçalhos de protocolos armazenadas nas tabelas de fluxo dos comutadores *OpenFlow* (ROTHENBERG et al., 2010). Os principais componentes da tabela de fluxo são (ONF, 2012b):

- ***match fields***: são campos que representam porta de ingresso do pacote no *switch* e cabeçalhos de protocolos. A combinação desses campos são utilizados para formar as regras da tabela de fluxo. Para comutadores que desejam implementar o *OpenFlow* em seu plano de dados, a especificação do protocolo define requisitos de quais campos são de implementação obrigatória ou não, conforme está descrito na Tabela 2.3. Portanto, a quantidade suportada de *match fields* opcionais entre diferentes *switches* podem variar, mesmo ambos utilizando a mesma versão do protocolo;
- **prioridade de execução do fluxo**: cada fluxo possui uma prioridade (número inteiro) associada, que determina sua ordem de execução no plano de dados;
- **ações**: são as decisões instaladas pelo controlador no plano de dados para executar ao receber os fluxos de entrada, tais como: encaminhar o pacote do fluxo para uma porta do *switch*, encapsular e encaminhar o pacote via canal seguro para o próprio controlador tomar alguma decisão a respeito, modificar campos de cabeçalhos e encaminhar o pacote modificado para uma porta de saída, ou ainda, descartar o pacote. Cada fluxo pode ter uma ou mais ações associadas;
- **contador de fluxo**: variável de número inteiro que incrementa a cada vez que determinado fluxo entrada é identificado no plano de dados, faz parte das variáveis de estatística da rede suportada pelo *OpenFlow*, juntamente com o contador de bytes de fluxo, entre outras;

A empresa *Nicira Networks* adicionou suporte a alguns campos que não estão previstos na documentação oficial do *OpenFlow* 1.3, como por exemplo, o *match field* TCP_FLAGS, que oferece suporte às flags de estado de conexão TCP (SYN, SYN+ACK, ACK), todavia, são de implementação opcional.

O fluxo de entrada é identificado por seus *match fields* e sua prioridade. Sendo assim, o plano de dados busca casar essas informações entre o fluxo de entrada e os fluxos instalados na

Tabela 2.3 – Principais *match fields* suportados pelo *OpenFlow* versão 1.3

Nome do Campo	Requisito	Descrição
OXM_OF_IN_PORT	obrigatório	Porta de ingresso do pacote no *switch*.
OXM_OF_ETH_DST	obrigatório	Endereço de destino Ethernet.
OXM_OF_ETH_SRC	obrigatório	Endereço de origem Ethernet.
OXM_OF_ETH_TYPE	obrigatório	Tipo Ethernet do pacote *OpenFlow*.
OXM_OF_IP_PROTO	obrigatório	Número de protocolo IPV4 ou IPV6.
OXM_OF_IPV4_SRC	obrigatório	Endereço de origem IPV4.
OXM_OF_IPV4_DST	obrigatório	Endereço de destino IPV4.
OXM_OF_IPV6_SRC	obrigatório	Endereço de origem IPV6.
OXM_OF_IPV6_DST	obrigatório	Endereço de destino IPV6.
OXM_OF_TCP_SRC	obrigatório	Porta de origem TCP.
OXM_OF_TCP_DST	obrigatório	Porta de destino TCP.
OXM_OF_UDP_SRC	obrigatório	Porta de origem UDP
OXM_OF_UDP_DST	obrigatório	Porta de destino UDP.
OXM_OF_ICMPV4_TYPE	opcional	Tipo ICMP.
OXM_OF_ICMPV4_CODE	opcional	Código ICMP.
OXM_OF_ARP_OP	opcional	Código de operação ARP.
OXM_OF_ARP_SPA	opcional	Endereço IPV4 de origem no payload ARP.
OXM_OF_ARP_TPA	opcional	Endereço IPV4 alvo no payload ARP.
OXM_OF_ARP_SHA	opcional	Endereço Ethernet de origem no payload ARP.
OXM_OF_ARP_THA	opcional	Endereço Ethernet alvo no payload ARP.

Fonte: adaptada (ONF, 2012b).

tabela, para então executar as ações (ONF, 2012b). Quando o valor de determinado *match field* não importar, ele pode ser omitido na instalação do fluxo na tabela.

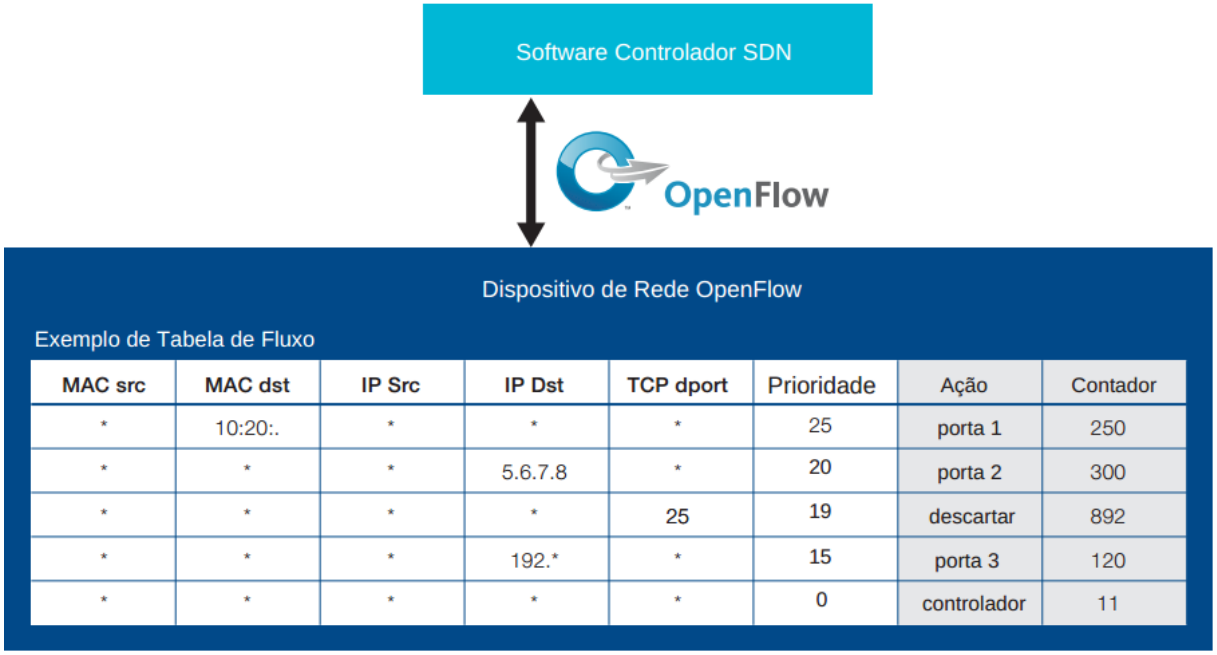
Por padrão, é instalado um fluxo especial chamado ***table-miss flow entry***. Este fluxo possui prioridade zero e todos *match fields* omitidos, para que, o *switch* envie para o controlador decidir o destino de fluxos de entrada que não estão instalados na tabela, como pode ser observado na Figura 2.5.

Portanto, é encaminhado para o controlador apenas o primeiro pacote de determinado fluxo na rede, já que a partir dos próximos pacotes, o fluxo estará instalado na tabela com alguma ação a ser executada, desde que a ação instalada não seja justamente a de enviar os pacotes do fluxo para o controlador.

2.2.1.2 Controladores

Os controladores são os softwares inteligentes numa rede *OpenFlow*/SDN, pois são eles que gerenciam a rede, agindo como um sistema operacional. Além de fornecer bibliotecas

Figura 2.5 – Tabela de fluxo *OpenFlow*.



Fonte: (ONF, 2012a).

para manipular recursos da rede, o controlador também estabelece regras por meio de API programável em linguagem de alto nível, que são executadas pelo plano de dados dos comutadores (SHALIMOV et al., 2013). Será descrito nesta Subseção alguns dos principais controladores *OpenFlow* de código aberto apresentados na Tabela 2.4.

Tabela 2.4 – Principais controladores *OpenFlow*

Controlador	Linguagem	Licença
NOX	C++	GPL v3
POX	Python	Apache 2.0
Beacon	Java	GPL v2
OpenDayLight	Java	EPL
ONOS	Java	Apache 2.0
Ryu	Python	Apache 2.0

Fonte: adaptada (SENA, 2014).

NOX foi um importante controlador *OpenFlow*, por ter sido o precursor que surgiu juntamente com o protocolo. Foi Implementado em C++ por profissionais da Nicira Networks e pesquisadores da Universidade de Stanford e Berkeley no ano de 2008 (GUDE et al., 2008). Com o passar dos anos e a evolução do *OpenFlow*, o NOX se tornou um controlador obsoleto, tendo em vista que, encerrou seu ciclo de atualizações.

O controlador POX, segundo Sena (2014), é uma versão melhorada em Python do controlador NOX. Apesar de ser um controlador ainda bastante utilizado, o POX é suportado

apenas pela versão 1.0 do *OpenFlow*.

O *Beacon* é um dos primeiros controladores *OpenFlow* open source implementado em Java. Lançado em 2010, tem sido amplamente utilizado em pesquisa e ensino, além disso, serviu de base para outros controladores mais complexos, a exemplo do controlador *Floodlight* (ERICKSON, 2013).

OpenDayLight (ODL), é um controlador implementado em Java que faz parte de um projeto de código aberto mantido pela *The Linux Foundation*. A intenção desse projeto é criar um controlador com código robusto para abranger a maioria dos componentes de uma arquitetura SDN, para obter aceitação entre fabricantes e usuários, além de manter uma comunidade crescente para auxiliar no seu desenvolvimento e utilizá-lo em produtos comerciais (KHONDOKER et al., 2014). Em razão dessas circunstâncias, o ODL é um dos controladores mais utilizados atualmente.

O controlador *Open Network Operating System* (ONOS), é uma plataforma de controle distribuído focado nos requisitos de desempenho, escalabilidade e disponibilidade de grandes redes SDN (BERDE et al., 2014). o ONOS também possibilita uma visão global e centralizada da rede mesmo que fisicamente esteja em uma arquitetura distribuída em múltiplos servidores.

O *Ryu SDN Framework* é um controlador baseado em componentes que possui licença Apache 2.0. Desenvolvido inteiramente em Python, o Ryu fornece uma coleção de APIs que facilitam a implementação de aplicações para uma rede SDN, além disso, suporta as cinco versões do *OpenFlow* juntamente com as extensões da *Nicira Networks* (NTT, 2018).

2.2.1.3 Mensagens *OpenFlow*

A comunicação entre um controlador e um *switch* numa rede *OpenFlow*, ocorre por meio de mensagens padronizadas. Essas mensagens são de caráter informativo ou de mudanças de estado do *switch*. Existem três tipos de mensagens presentes em todas as versões do *OpenFlow* (ONF, 2009):

- **controlador para o *switch***: São mensagens iniciadas pelo controlador, algumas delas requerem uma resposta do *switch*. Esse tipo de mensagem permite que o controlador gerencie o estado lógico do comutador de modo proativo. Exemplos dessa classe de mensagens são: ***modify-state***, que se encarrega de adicionar, editar e deletar fluxos da tabela e a mensagem ***packet-out*** que envia um pacote do controlador para uma porta

especifica do *switch* sem a necessidade de instalar fluxo na tabela;

- **assíncrona**: são mensagens enviadas do *switch* para o controlador sem solicitação. Algumas dessas mensagens podem fazer com que o controlador opere de forma reativa. Essa classe é composta por diversas mensagens de estado. Uma das principais mensagens desse tipo é a ***packet-in***, através dela, o *switch* envia para o controlador todos os fluxos de entrada que não possuem ações na tabela. Além da mensagem *packet-in*, outras comumente utilizadas são ***flow-removed*** e ***port-status***, que respectivamente, informa ao controlador quando um fluxo é removido da tabela e quando uma porta do *switch* muda de estado;
- **simétrica**: são mensagens enviadas em ambas direções sem a necessidade de solicitação. A mensagem ***hello*** por exemplo, é permutada entre o *switch* e o controlador no início da conexão entre os dois.

2.2.1.4 *Software Switches*

Os *switches OpenFlow* são classificados em dois tipos: comutadores dedicado e comutadores habilitado. Os comutadores dedicados são aqueles que não suportam roteamentos nível 2 e nível 3 nativamente, apenas encaminham os fluxos de entrada baseado na tabela de fluxos (FARIAS et al., 2011). Já os comutadores habilitado, segundo Sena (2014), são *switches* comuns, no qual o protocolo *OpenFlow* é acrescentado, geralmente pela a troca do seu *firmware*.

Em substituição ao *firmware* original do comutador comum, é instalado um sistema operacional com o *software switch*, que consiste na implementação de um *switch OpenFlow* virtual que opera somente com o plano de dados (GORANSSON; BLACK; CULVER, 2017). Isso permite o funcionamento de uma rede SDN em equipamentos simples de rede a um baixo custo. Alguns dos *software switches* de código aberto mais utilizados podem ser observados na Tabela 2.5.

Tabela 2.5 – Implementações de *software switches*

Software Switch	Linguagem	Licença
Pantou	C	GPL v2
CPqD Ofsoftswitch13	C	BSD-Like
LINC	Erlang	Apache 2.0
Open vSwitch	C	Apache 2.0

Fonte: adaptada (GORANSSON; BLACK; CULVER, 2017).

O Pantou foi desenvolvido pela a Universidade de Stanford e tem o propósito de transformar um *switch wireless* simples em um comutador *OpenFlow* habilitado. O Pantou faz

uso do *OpenWRT* 10.03, codinome *BackFire*, que consiste em um sistema operacional Linux utilizado em dispositivos de rede embarcado para roteamento (AZODOLMOLKY, 2013). No entanto o Pantou suporta apenas a versão 1.0 do *OpenFlow*.

O CPqD Ofsoftswitch13 é um *OpenFlow 1.3 software switch* criado por pesquisadores da Unicamp. Segundo Fernandes (2015), o propósito deste projeto é oferecer um simples comutador em software com requisitos básicos de desempenho, para prover funcionalidades inerente à versão 1.3 do protocolo. A exemplo do Pantou, o CPqD Ofsoftswitch13 fornece uma versão que utiliza o *OpenWRT BackFire* para instalação em equipamentos de baixo custo que possibilita experimentos em redes reais (OFSOFTSWITCH13; CPQD, 2017).

LINC é um projeto de *software switch* com código aberto fomentado pela a FlowForwarding, que é uma comunidade que promove projetos de software livre que envolvem *OpenFlow* com licença Apache 2. O LINC suporta versões 1.2 e 1.3 do *OpenFlow*, e faz uso da arquitetura de hardware genérica x86, possibilitando sua instalação em múltiplas plataformas como Linux, Solaris, MacOS, FreeBSD (DUMKA, 2018).

O Open vSwitch (OVS) é uma das implementações de *switch* mais conhecida e mais utilizada. Mantido pela a comunidade *Linux Foundation*, o OVS foi desenvolvido como um software que executa parte de seus módulos no kernel Linux, podendo ser instalado em diversas distribuições deste sistema operacional (HU, 2014). Além disso, versões mais atuais do OVS como a 2.9, suporta todas versões do *OpenFlow* inclusive com as extensões da *Nicira Networks* (OPENVSWITCH, 2018).

2.3 *Dynamic Host Configuration Protocol* (DHCP)

O *Dynamic Host Configuration Protocol* (DHCP), é um protocolo da camada de aplicação que permite a distribuição dinâmica e automática de endereços IP na rede. Segundo a RFC - 2131 (DROMS, 1997), o DHCP é um protocolo que utiliza o modelo cliente-servidor. O servidor opera alocando e configurando endereços IP dinamicamente para os demais *hosts* da rede. A utilização do DHCP pode evitar erros causados por uma configuração manual de IP, como por exemplo, o uso de um mesmo endereço IP em mais de uma máquina, ou então, um IP inválido atribuído a um determinado *host*.

2.3.1 Mensagens DHCP

Por ser um protocolo de natureza cliente-servidor, o protocolo estabelece suas configurações de rede por meio de troca de mensagens entre estas duas entidades. O DHCP utiliza o UDP como protocolo de transporte, no qual, o cliente envia e recebe mensagens DHCP na porta UDP 68, e o servidor utiliza a porta UDP 67 para esta mesma finalidade (KUROSE; ROSS, 2013).

De acordo com Droms (1997), as principais mensagens DHCP são:

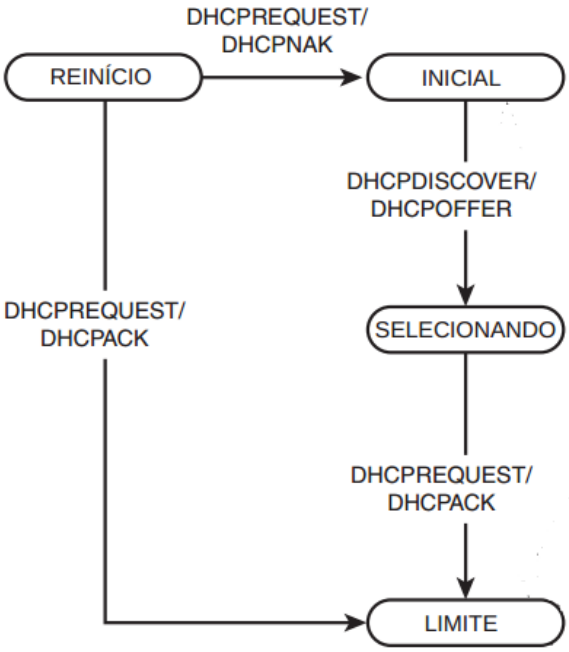
- **DHCPDISCOVER**: mensagem inicial enviada pelo cliente para toda rede por *broadcast*, para encontrar algum servidor DHCP e obter um endereço IP e outros parâmetros de configuração;
- **DHCPOFFER**: mensagem do servidor para o cliente após receber um DHCPDISCOVER. Nesta mensagem o servidor oferece IP e os demais parâmetros de configuração ao cliente;
- **DHCPREQUEST**: mensagem que o cliente envia para o servidor após receber um DHCPOFFER. Por essa mensagem o cliente requisita ao servidor o endereço IP que lhe foi ofertado, juntamente com as configurações de rede;
- **DHCPACK**: após o servidor DHCP receber um DHCPREQUEST, ele confirma o endereço atribuído e suas configurações ao cliente;
- **DHCPNACK**: mensagem de exceção enviada do servidor para o cliente após receber um DHCPREQUEST. Esta mensagem indica ao cliente que ele está solicitando um IP de uma rede/sub-rede incorreta. Neste caso, o cliente pode ter sido movido para uma outra rede/sub-rede ou sua concessão de tempo para uso daquele IP está expirado. O cliente recebe o DHCPNACK, libera o IP antigo e volta a enviar DHCPDISCOVER para toda a rede.

O processo do cliente para obter um IP via DHCP pode ser descrito como uma máquina de estado finita, como ilustra a Figura 2.6. Segundo Droms e Lemon (2003) esta máquina possui os seguintes estados:

1) **inicial**: estado que se encontra um cliente quando não possui um IP válido;
2) **selecionando**: estado quando o cliente está negociando com o servidor o seu endereço IP e suas configurações;
3) **limite**: estado final, após o cliente obter um IP válido;

4) **reinício**: ao reiniciar a conexão, o cliente é encaminhado para este estado. Se o servidor confirmar o IP ele passa para o estado de limite novamente. Caso o cliente receba como resposta um DHCPNAK, ele retorna ao estado inicial.

Figura 2.6 – Máquina de estado finito para obtenção de endereço IP.



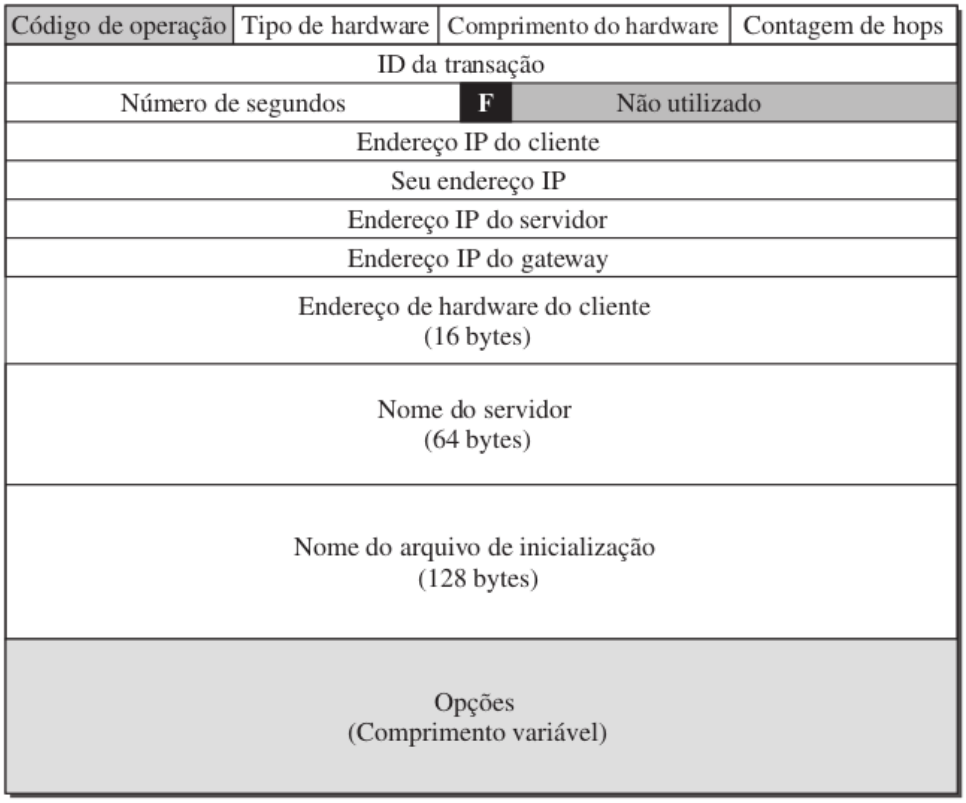
Fonte: (DROMS; LEMON, 2003).

A Figura 2.7 mostra o formato de uma mensagem DHCP. Todas as mensagens descritas utilizam o mesmo formato. Entretanto, o tamanho de cada mensagem é variável, pois depende da quantidade de parâmetros de configuração que ela possui.

O formato da mensagem DHCP possui os seguintes campos (FOROUZAN, 2010):

- **código de operação**: campo de 1 byte que define o tipo de mensagem. Sendo o valor 1 para mensagem cliente (pedido) ou o valor 2 para mensagem do servidor (resposta);
- **tipo de hardware**: campo de 1 byte que define o tipo de rede física. Por exemplo, para Ethernet o valor é 1;
- **comprimento do hardware**: campo de 1 byte que informa o tamanho do endereço físico. Para endereços físicos Ethernet esse valor é 6;
- **contagem de hops**: campo de 1 byte que informa o número máximo de saltos que a mensagem pode alcançar;
- **ID da transação**: ID de 4 bytes para identificação da troca de mensagem entre cliente e servidor;
- **número de segundos**: campo de 2 bytes que expressa tempo da transação;

Figura 2.7 – Formato da mensagem DHCP



Fonte: (FOROUZAN, 2010).

- **flags (F)**: flag de 1 bit, no qual o cliente aciona se desejar uma resposta por *broadcast* do servidor;
- **endereço IP do cliente**: campo de 4 bytes que armazena o IP do cliente. Quando esse valor está ausente é atribuido o endereço 0.0.0.0;
- **seu endereço IP**: campo de 4 bytes configurado pelo servidor indicando o endereço IP que o cliente pode obter;
- **endereço IP do servidor**: campo de 4 bytes, configurado pelo servidor que indica seu endereço IP para o cliente;
- **endereço IP do gateway**: campo de 4 bytes, no qual o servidor indica o IP do roteador da rede;
- **endereço de hardware do cliente**: endereço físico de 16 bytes, que indica o endereço MAC do cliente;
- **nome do servidor**: campo opcional de 64 bytes configurado pelo servidor indicando seu nome de identificação na rede;
- **nome do arquivo de inicialização**: campo de 128 bytes opcional, no qual o servidor pode indicar ao cliente o caminho de um arquivo para ele recuperar configurações de inicialização posteriormente;

- **opções**: campo de lista de opções variáveis. Esse campo pode possuir até 312 bytes. Nele contém obrigatoriamente o tipo da mensagem DHCP, e opcionalmente os demais parâmetros de configuração.

Os **parâmetros de configurações** da mensagem DHCP estão descritos na RFC – 2132, que traz todas as opções que uma mensagem pode possuir. Entre as principais opções pode-se destacar os seguintes parâmetros: a máscara de sub-rede, tempo de concessão do endereço IP, lista de servidores DNS, lista de roteadores da rede, nome de identificação do *host* na rede, etc. (ALEXANDER; GRAPHICS; DROMS, 1997).

2.3.2 Considerações com Relação à Segurança

De acordo com Droms (1997), pelo fato das mensagens DHCP não ter nenhum tipo de autenticação, a rede pode ser vítima de servidores falsos, que se passam por servidores DHCP confiáveis e distribuem configurações incorretas e maliciosas.

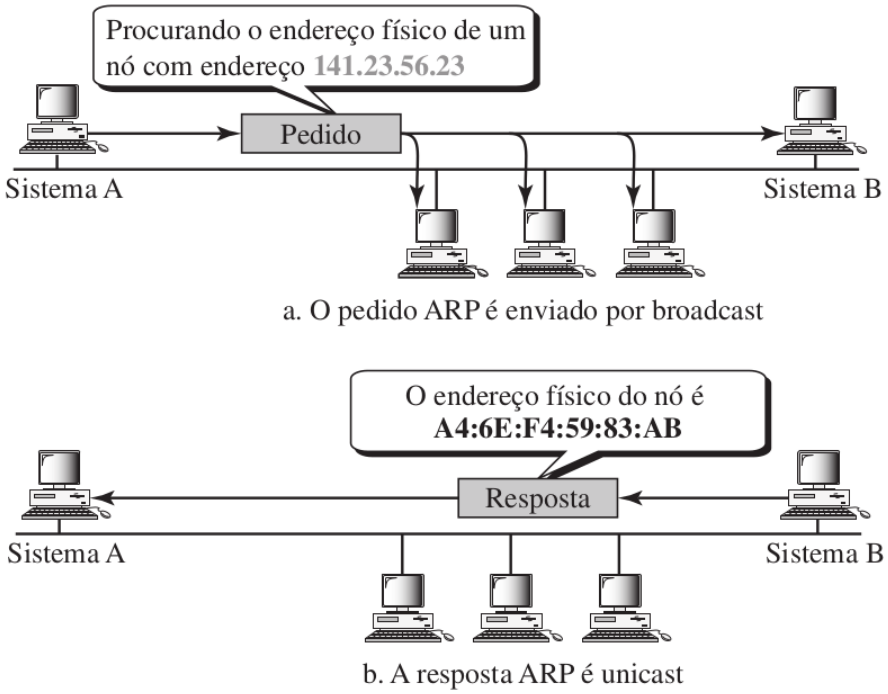
Esforços do meio científico tem almejado mitigar essa falta de autenticação do protocolo DHCP, exemplo disso, é o trabalho proposto por (DUANGPHASUK; KUNGPISDAN; HAN-KLA, 2011), que oferece um mecanismo de segurança para o protocolo, utilizando certificados digitais.

2.4 *Address Resolution Protocol* (ARP)

O *Address Resolution Protocol* (ARP), é um protocolo da camada de enlace definido pela RFC – 826, que de acordo com Plummer (1982), permite a resolução de endereços lógicos (endereço da camada de rede) para endereços físicos (endereço MAC) .

O funcionamento do ARP é simples, conforme explica Forouzan (2010), quando um nó deseja enviar um datagrama IP para outro nó na mesma rede, ele primeiro faz uso do ARP para descobrir o endereço MAC do destinatário. Um pacote ARP é enviado para todos os *hosts* da rede via *broadcast* com os dados do remetente (IP e MAC de origem) e apenas com o IP do destinatário, solicitando que o dono daquele IP responda com o seu endereço MAC, como pode ser visto na Figura 2.8.

Figura 2.8 – Operação ARP.



Fonte: (FOROUZAN, 2010).

Ao receber o MAC do destinatário via resposta *unicast*, o remetente armazena na sua tabela ARP o IP e o MAC que lhe foram informados (GURGEL et al., 2015). Após isso, os dois nós estão aptos a se comunicarem na rede normalmente.

Quando o remetente deseja se comunicar com um nó externo, ele envia um pedido ARP com o IP do roteador de sua rede, o roteador responde ao pedido com seu MAC. A partir disso o remetente envia para o seu roteador o pacote destinado ao nó externo (KUROSE; ROSS, 2013). Sendo assim, o roteador da rede se torna o responsável de repassar o pacote para o nó externo correto.

2.4.1 Mensagens ARP

As duas mensagens mais importantes do protocolo ARP são: ARP REQUEST que envia uma solitação de endereço físico e ARP REPLY, que consiste na resposta da solicitação. Os cabeçalhos das mensagens ARP são formados por 4 octetos por linha, porém a mensagem ARP não tem um tamanho fixo, ela depende do tamanho dos endereços ao qual é feito o mapeamento (COMER, 2017). A Figura 2.9 ilustra o formato da mensagem ARP ao mapear endereço IPv4 em endereço Ethernet.

Figura 2.9 – Formato da mensagem ARP.

Fonte: (COMER, 2017).

O formato da mensagem ARP possui os seguintes campos (COMER, 2017):

- **tipo de hardware**: tipo de interface de rede, por exemplo, o valor é 1 para tipo de hardware Ethernet;
- **tipo de protocolo**: tipo de protocolo de rede para ser mapeado em endereço físico, no caso IPv4 possui um valor hexadecimal de 0x800;
- **hlen e plen**: tamanho do endereço de hardware e tamanho do endereço de protocolo em bytes respectivamente, Ethernet possui o valor hlen de 6 bytes e IPv4 possui o valor plen de 4 bytes;
- **operação**: tipo da operação da mensagem, 1 para ARP REQUEST e 2 para ARP REPLY;
- **hard emissor e emissor IPv4**: Endereço físico (MAC) e endereço lógico de rede (IPv4) do rementente da mensagem respectivamente;
- **hard destino e destino IPv4**: Endereço físico (MAC) e endereço lógico de rede (IPv4) do destinatário da mensagem respectivamente, quando a operação é 1 (ARP REQUEST) o valor 0 é atribuido ao hard destino, pois é desconhecido.

2.4.2 Vulnerabilidades do Protocolo

Apesar de ser um protocolo bastante relevante para o funcionamento da rede, sobretudo, de uma rede local, o ARP possui sérios problemas com relação à segurança. Segundo afirma Goodrich e Tamassia (2013), semelhantemente ao DHCP, por não ter um sistema de autenticação da mensagem ARP, qualquer computador da rede pode responder maliciosamente a um ARP REQUEST, ou até mesmo enviar ARP REPLY ofensivo sem solicitação, fazendo com que a vítima atualize sua tabela ARP com informações falsas. Esse tipo de ataque também é conhecido como envenenamento da tabela ARP.

Quando um *host* malicioso informa dados falsificados em mensagens ARP's envolvendo duas ou mais vítimas, ele tem condições de interceptar todos os pacotes enviados entre essas vítimas, esse ataque é denominado de homem-do-meio (GURGEL et al., 2015).

De acordo com Basta, Basta e Brown (2014), existem algumas soluções que buscam mitigar esse tipo de ataque ARP, como por exemplo, métodos que bloqueia a alteração das tabelas ARP e sua configuração manual, ou seja, a utilização de forma estática.

2.5 Trabalhos Relacionados

O NFShunt é um *Firewall* híbrido baseado em SDN proposto por (MITEFF; HAZELHURST, 2015), composto pelo *Netfilter*[1] e por regras de política de segurança desenvolvidas no controlador POX. Sua finalidade é proteger a *Science DMZ*, que é um tipo de rede de alto desempenho para suportar pesquisas de dados intensivos (DART et al., 2013). Considerando isso, o NFShunt tem o propósito de fornecer segurança sem comprometer o desempenho da *Science DMZ*, que é um requisito essencial para seu funcionamento correto. A proposta apresenta bons resultados de desempenho, entretanto para utilizar o NFShunt é necessário ter conhecimentos específicos da sintaxe de comandos do *IPtables*, que é uma espécie de interface gráfica em linha de comando do *NetFilter* (NEGUS; BRESNAHAN, 2014). Todavia, a proposta dessa monografia é oferecer um *firewall* com interface gráfica web intuitiva para o usuário, no qual ele não necessita possuir conhecimentos específicos em qualquer ferramenta.

Em (KUMAR; SRINATH, 2016), é proposto um *firewall* SDN dedicado para filtragem de pacotes na camada de transporte em redes *full mesh* utilizando o controlador POX. Esse tipo de rede pode ser complexo para escolher um ponto central para posicionar um *firewall* tradicional, tendo em vista que, todos os dispositivos do núcleo da rede estão conectados ponto a ponto. Entretanto os desenvolvedores dessa solução se beneficiaram com a visão geral da rede proporcionada pelo paradigma SDN, para implementar o software. O *firewall* apresentado, foi avaliado em ambiente virtual, utilizando seis comutadores com um *host* por comutador. Ao contrário do trabalho proposto por (KUMAR; SRINATH, 2016), o *DmzVisor* será implementado utilizando o controlador Ryu e avaliado em ambiente virtual de topologia estrela.

O trabalho de (KAUR et al., 2015), consiste em um *firewall* SDN que demonstra como uma aplicação dessa natureza pode ter funcionalidades similares a de um equipamento tradicional,

[1] *Netfilter*: *firewall* nativo do sistema operacional GNU/Linux.

porém sem utilizar um hardware dedicado. O software tem como objetivo filtrar pacotes de algumas camadas de rede, prevenir ataques de *IP Spoofing*, dentre outros aspectos, utilizando o controlador POX. O *firewall* permite ou rejeita o tráfego baseado no MAC de origem e destino, endereço IP de origem e destino, além de porta de destino. A avaliação da aplicação foi realizada em ambiente virtual com tráfegos de protocolos como ICMP e HTTP. Além de desempenhar estas funcionalidades do software apresentado por (KAUR et al., 2015), o trabalho proposto por esta monografia traz o isolamento de redes, como uma segurança adicional para os ativos de informação das corporações.

O *firewall* exposto por (BADOTRA; SINGH, 2018), consiste numa aplicação SDN desenvolvida no controlador Ryu, que provê segurança para as camadas de transporte e aplicação, por meio da filtragem de pacotes. O sistema é capaz de bloquear o acesso de pacotes TCP, ICMP e web nestas camadas, no qual, a regra pode ser aplicada em um único *host* ou na rede inteira. A avaliação dessa proposta consistiu no bloqueio de determinados sites da web e do protocolo ICMP. Ao contrário desta aplicação, o *software* apresentado neste trabalho fornece segurança às camadas de enlace, rede e transporte. Além do mais, o *DmzVisor* também permite regras para máquinas específicas (servidores da DMZ) ou para toda a rede (LAN).

A proposta de (ZERKANE et al., 2016) oferece um *firewall OpenFlow* reativo de conexões de estado. Esse *firewall* busca tratar conexões TCP, por meio de monitoramento do seu estado, para mitigar ataques do tipo DoS de inundação de TCP SYN. A aplicação foi desenvolvida fazendo uso do controlador Ryu e fornecendo ao usuário uma interface gráfica. O trabalho apenas avaliou o estado de conexão de um servidor HTTP, por meio de uma topologia composta por 3 clientes, no qual, cada cliente realizava diversas consultas simultâneas ao servidor web. No entanto, diferentemente deste *firewall* de conexão de estado, o *DmzVisor* tem a responsabilidade de aplicar regras de segurança em duas redes distintas e, além disso, mantê-las separadas uma da outra.

A proposta deste trabalho é oferecer certa comodidade ao administrador de rede através de um *firewall* amigável com regras para as camadas de rede e transporte de fácil gerenciamento, e além disso, toda a segurança que segrega as redes em LAN e DMZ pelo tipo dos *hosts* conectados, mantendo ambas as redes isoladas. A Tabela 2.6 faz um comparativo entre os trabalhos relacionados.

Tabela 2.6 – Comparativo entre trabalhos relacionados e trabalho proposto.

Trabalho	Propósito do *Firewall*	Versão *OpenFlow*	Controlador	Interface de usuário
(MITEFF; HAZELHURST, 2015)	Aplicação híbrida que utiliza NetFilter para segurança da Science DMZ.	1.0	POX	IPTables
(KUMAR; SRINATH, 2016)	Sistema de filtro de pacotes na camada de transporte em redes full mesh.	1.0	POX	Controlador
(KAUR et al., 2015)	Filtragem de pacotes baseado nas camadas de um a quatro.	1.0	POX	Controlador
(BADOTRA; SINGH, 2018)	Filtragem de pacotes nas camadas de transporte e aplicação.	Não Informa	Ryu	Controlador
(ZERKANE et al., 2016)	Aplicação reativa para monitoramento de estado de conexões TCP.	Não Informa	Ryu	Interface gráfica web
Trabalho proposto	Isolamento de tráfego entre redes e filtragem de pacotes nas camadas de rede e transporte.	1.3	Ryu	Interface gráfica web

Fonte: Elaborado pelo autor.

2.6 Resumo do Capítulo

Este capítulo discutiu o referencial teórico no qual está fundamentado este trabalho. Também foi abordado os trabalhos relacionados com a finalidade de demonstrar a importância do trabalho proposto.

O capítulo inicia abordando a relevância da segurança de redes para as organizações, incluindo a importância da utilização do *firewall* para proteção dos ativos da empresa, bem como suas variações de tipos e funcionalidades. Assim como foi tratado a respeito do propósito da DMZ.

Logo após foi discutido sobre o paradigma de redes SDN, discutindo suas vantagens e benefícios, assim como também, apresentando aplicações desse novo modelo em diferentes áreas e inclusive entre grandes empresas de tecnologia a exemplo do Google e Facebook. Também foi explorado os principais componentes da arquitetura SDN, sobretudo, o protocolo *OpenFlow* juntamente com suas características e tecnologias empregadas. Além disso foram apresentados a estrutura e funcionamento dos protocolos DHCP e ARP, bem como suas devidas vulnerabilidades

com relação à segurança em razão da falta de autenticação desses protocolos.

Por fim, foi retratado artigos de trabalhos relacionados com o trabalho proposto, por meio da descrição de cinco trabalhos da área publicados nos últimos três anos, apontando suas características e diferenças em relação à proposta desta monografia.

3 *DmzVisor*: SDN *Firewall*

Neste capítulo será discutido a proposta desta monografia. Na Seção 3.1 é apresentado a visão geral da aplicação proposta, assim como sua arquitetura com a descrição dos módulos que compõem o protótipo e materiais utilizados. A Seção 3.2 explana sobre os métodos desenvolvidos para a segurança de protocolos e isolamento das redes. A Seção 3.3 traz uma narrativa sobre a implementação e funcionamento do firewall. O Sistema web proposto para gerência do firewall é exposto na Seção 3.4, e por fim a Seção 3.5 denota o resumo deste capítulo.

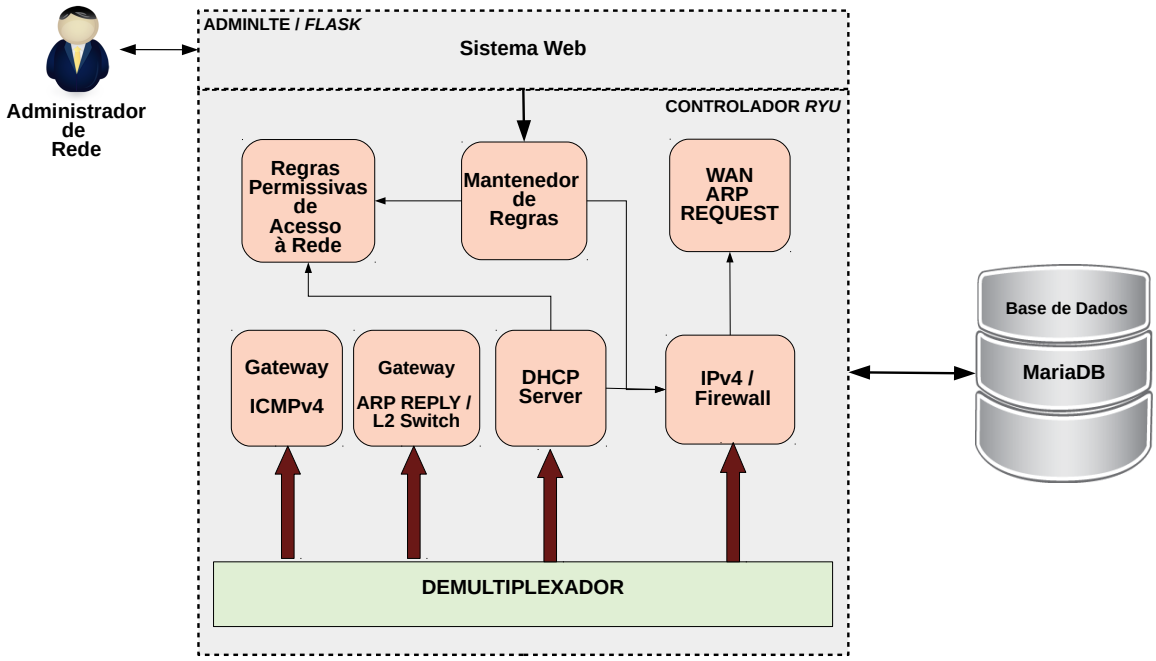
3.1 Contexto Geral do Protótipo

O principal propósito do *DmzVisor* além de oferecer uma interface web para gerência do *firewall*, é dividir a rede interna de uma organização entre LAN e DMZ, conforme interesse do administrador de rede, mantendo ambas isoladas para garantir a segurança dos ativos da organização.

A proposta foi implementada utilizando o ***Ryu*** (NTT, 2018), um controlador SDN simples que traz consigo toda a praticidade e robustez da linguagem Python, juntamente com o ***OpenFlow* 1.3** (ONF, 2012b), que é uma versão estável do protocolo que suporta os requisitos necessários para a implementação desta aplicação. A arquitetura simplificada do *DmzVisor* é descrita na Figura 3.1 e possui os seguintes módulos e componentes:

- **Módulos implementados no controlador *Ryu***:
  - **Demultiplexador**: este módulo tem o importante papel de receber os fluxos oriundos do *switch* via *packet-in*. O demultiplexador faz uma análise do protocolo do fluxo e o encaminha para o módulo correto fazer o devido tratamento;
  - ***Gateway* ICMPv4**: módulo que apenas responde mensagens ICMP ECHO REQUEST (ping) destinadas aos *gateways* da rede. Sendo assim, quando um fluxo ICMPv4 chega ao Demultiplexador ele analisa se o destino é o *gateway* da rede e

Figura 3.1 – Arquitetura Simplificada do *DmzVisor*



encaminha o fluxo para este módulo. Caso o destino seja outro *host* em uma outra rede, o fluxo é enviado para o módulo IPv4 / *Firewall*;

– ***Gateway* ARP REPLY / L2 *Switch***: Atende às requisições ARP destinadas aos *gateways* e faz toda a segurança de mensagens ARP entre as máquinas em conjunto com a comutação de nível 2, ou seja, a comutação entre os *hosts* da mesma rede;

– **DHCP Server**: É o responsável por distribuir os devidos endereços IPs para os dispositivos na LAN e DMZ, de acordo com o perfil de cada *host* conectado na rede;

– **IPv4 / *Firewall***: principal módulo do controlador, com a finalidade de instalar as regras de roteamento IPv4 nas redes LAN e DMZ com base nas regras de *firewall* definidas pelo administrador;

– **Regras permissivas de acesso à rede**: módulo responsável por instalar regras que possibilita a inspeção pelo controlador dos pacotes sem rota definida dos novos *hosts*, a partir disso, o controlador poderá criar as devidas rotas. Sem essas regras permissivas os *hosts* ingressantes ficariam isolados;

– **Mantenedor de regras**: Este módulo é uma *thread* dedicada que faz a comunicação com a aplicação web. O mantenedor de regras permanece à escuta de requisições do sistema web recebendo comandos tais como: adicionar, editar, excluir regras e as aplica com o auxílio de outros módulos;

  - **WAN ARP REQUEST**: módulo que envia exclusivamente mensagem ARP REQUEST para obter endereços MAC de elementos da rede que estejam conectados através da porta WAN;

- **Sistema Web**: Componente web para a gerência dos recursos providos pelo *firewall*. O *front-end* do sistema foi desenvolvido utilizando o template administrativo **AdminLTE**. O AdminLTE é um projeto *open source* com Licença MIT que faz uso do framework Bootstrap (ADMINLTE, 2017). Sua adoção como base do *front-end* do *DmzVisor* foi motivado por ser um template bastante amigável e personalizável. Já o *back-end* da aplicação foi construída com **Flask** (FLASK, 2018), um Microframework Python de código aberto com licença BSD focado no lado do servidor em aplicações web. A utilização desse *framework* permitiu o reuso de código desenvolvido no controlador, sobretudo, das classes de conexão com a base de dados.
- **Base de dados**: Componente para a persistência de dados da aplicação. Nesta base de dados contém o mapeamento de configuração dos *hosts* conectados em cada rede, assim como as regras de *firewall* definidas pelo usuário. A utilização dessa persistência, permite que, ao reiniciar o *switch* ou controlador da aplicação, nenhuma configuração realizada seja perdida. Depois que é reestabelecida a conexão entre o *switch* e o controlador, o módulo mantenedor de regras busca e aplica todas as regras e configurações da base de dados, oferecendo assim, resiliência às redes e à aplicação. Para esta finalidade foi utilizado o sistema de gerenciamento de banco de dados (SGBD) **MariaBD** (MARIADB, 2018), que consiste em um SGBD licenciado pela GPLv2, sendo também apresentado como um *fork* de código aberto do MySQL.

3.2 Segurança e Isolamento das Redes

Conforme já descrito anteriormente, o *DmzVisor* é responsável pelo roteamento seguro entre as redes interna (LAN e DMZ) e a externa (internet) tomando como base as regras de *firewall*. Em virtude disso, o controlador opera como uma espécie de *gateway* virtual e servidor DHCP para cada uma dessas redes internas. Cada *gateway* virtual hospedado no controlador é independente e possui seu próprio endereço IP e MAC, conforme apresenta a Tabela 3.1.

Tabela 3.1 – Informações dos *Gateways* virtuais criados pelo DmzVisor no controlador *OpenFlow*.

Rede	*Gateway* Virtual	Endereço MAC
LAN	10.0.0.1/24	0A:E4:1C:D1:3E:44
DMZ	192.168.0.1/24	0D:2C:6E:3C:D1:6D
WAN (Internet)	172.16.0.1/16	0B:2B:C9:4B:E3:57

Vale ressaltar que esses *gateways* não são módulos do controlador. O controlador apenas se identifica com endereços diferentes de acordo com qual *host* esteja se comunicando, baseado na rede a qual está inserido.

3.2.1 Ingresso do *Host* na Rede via DHCP

O servidor DHCP é o primeiro e principal mecanismo de isolamento das redes do *DmzVisor*. Este servidor foi desenvolvido de maneira personalizada. Ele opera fornecendo endereços IP com base no perfil de cada *host*, se for um *host* da LAN, o DHCP entregará um IP apropriado as configurações da LAN, da mesma forma, se for um *host* da DMZ, será entregue um IP apropriado a essa rede. Quem define o perfil de cada *host* conectado é o administrador de rede.

A princípio para que todo *host* ingresse na rede via DHCP de uma forma segura, foi necessário alterar o comportamento mais utilizado em uma rede *OpenFlow*. O fluxo *table-miss flow entry*, muito comum nas soluções SDN, foi substituido por um fluxo com a maior prioridade, que redireciona para o controlador todos os pacotes DHCP, conforme descrito na Tabela 3.2. Apenas o controlador está apto a responder essas requisições na rede, dessa forma, mitiga-se o risco de algum outro *host* malicioso responder como servidor DHCP, qualquer tentativa será descartada. A partir deste fluxo DHCP inicial são desencadeadas as demais regras do *DmzVisor*.

Tabela 3.2 – Fluxo DHCP instalado em substituição do *table-miss flow entry.*

ETH_TYPE	IP_PROTO	UDP_SRC	UDP_DST	Prioridade	Ação
IP	UDP	67	68	80	Controlador

Com a retirada do *table-miss flow entry*, o plano de dados passa a descartar automaticamente todos os pacotes de fluxos que não estão instalados na tabela do comutador *OpenFlow*. Desse modo, tem-se a possibilidade de oferecer mais segurança, pois, o *switch* somente tomará decisões baseado em fluxos confiáveis previamente instalados pelo controlador, sendo assim, rejeitando de maneira nativa os fluxos desconhecidos na rede. Dessa forma, enquanto um *host*

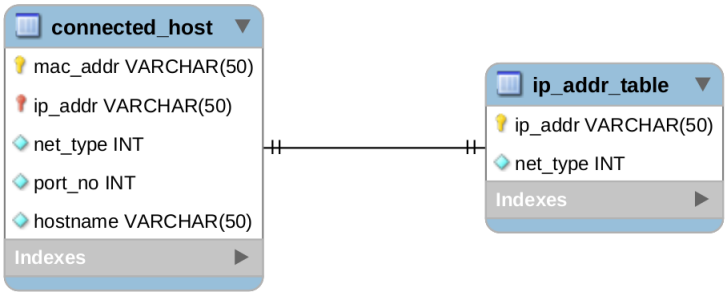
ingressante não se identificar através do DHCP, nenhum pacote vindo desse *host* será comutado ou processado, dificultando também as tentativas maliciosas de entrar na rede através de configurações manuais.

Foram implementados no módulo DHCP do controlador as três mensagens de resposta do servidor ao cliente, que são elas: DHCPOFFER, DHCPACK e DHCPNAK, esta última utilizada apenas na mudança de rede. Por padrão todos os *hosts* da rede que obtém endereço IP via DHCP pela primeira vez ingressam na rede LAN. Esse ingresso inicial consiste nas seguintes etapas:

1) O *host* envia em *broadcast* o pacote DHCPDISCOVER;
2) apesar do envio ter sido em *broadcast*, apenas o controlador recebe o pacote DHCPDISCOVER;
3) o módulo demultiplexador encaminha a mensagem para o módulo DHCP Server;
4) o módulo DHCP Server seleciona um endereço IP disponível na base de dados do tipo LAN. No SGBD há uma tabela previamente preenchida com os IP's das redes, juntamente com seu status (se o IP está disponível ou não);
5) o módulo constrói o pacote da mensagem DHCPOFFER e encaminha para o *host* via mensagem *OpenFlow packet-out*, com o IP fornecido e as configurações da rede LAN como: tempo de concessão de IP ilimitado, IP dos servidores DNS e DHCP (próprio *gateway*), máscara de sub-rede, IP do *gateway* LAN;
6) ao receber do *host* a mensagem DHCPREQUEST, o módulo DHCP envia a mensagem DHCPACK confirmando a entrega do endereço IP e as configurações;
7) após o envio do DHCPACK, o controlador instala fluxos de segurança do serviço de DHCP para o novo *host*. Além do mais, são inseridas as regras de *firewall* existentes no banco por meio do módulo IPv4/*Firewall*, juntamente com as regras permissivas de acesso à rede.

Durante a troca de mensagens entre o *host* e o controlador, o módulo DHCP armazena no SGDB as informações do dispositivo contidas nas mensagens e o número da porta na qual ele está conectado no *switch*. A partir do momento que o *host* obtém endereço IP por meio do módulo DHCP, as próximas conexões DHCP ou reinício desse dispositivo na rede, assim como também todas as regras de segurança, inclusive as regras de *firewall*, serão baseadas nos dados que foram mapeados para o banco. A Figura 3.2, mostra o modelo relacional e os dados que são mapeados de cada *host*.

Figura 3.2 – Modelo ER dos dados armazenados sobre os *hosts* da rede



Os fluxos de segurança para o serviço de DHCP instalados após a atribuição do endereço, são para vincular a porta do *switch* ao endereço MAC do *host* conectado. Com a adição deste fluxo vinculatório o controlador passa apenas a aceitar pacotes DHCP daquela porta se o endereço Ethernet de origem for o MAC que a ela está vinculado. Um exemplo é exposto na Tabela 3.3, no qual o *host* A foi mapeado na porta 2 do *switch*, o controlador descarta qualquer pacote DHCP oriundo da porta 2 que não seja o MAC do *host* A.

Tabela 3.3 – Fluxos de Segurança DHCP

IN_PORT	ETH_SRC	ETH_TYPE	IP_PROTO	UDP_SRC	UDP_DST	Prioridade	Ação
Porta 2	MAC *Host* A	IP	UDP	67	68	82	Controlador
Porta 2	*	IP	UDP	67	68	81	Descarta
*	*	IP	UDP	67	68	80	Controlador

Estes fluxos de segurança impedem que um *host* que teve sua segurança comprometida com o endereço MAC modificado de forma fraudulenta ingresse na rede. Se o fluxo não associar com nenhuma das duas regras vinculatórias Porta/Mac, significa que um novo dispositivo foi encontrado na rede numa porta que estava livre, sendo assim, é executado o fluxo DHCP padrão.

3.2.2 Regras Permissivas de Acesso à rede

As regras permissivas de acesso à rede, como visto anteriormente, são instaladas após a obtenção de endereço IP ao novo *host*, pois não há nenhuma regra ainda para ele se comunicar com outros *hosts* no plano de dados, somente os fluxos de segurança DHCP.

Em virtude disso, a aplicação insere fluxos para as máquinas confiáveis permitindo sua comunicação com outros dispositivos conectados. São fluxos enviados para o controlador que consistem nas seguintes regras:

- fluxo que permite o envio de ICMP ECHO REQUEST (ping) para o *gateway* de sua rede. Porém, esse envio é bloqueado para os *gateways* das outras redes, por questões de

isolamento e segurança do controlador.

- os fluxos do protocolo IP destinados à internet que não possui ação definida são enviados ao módulo IPv4 / *firewall* para realizar o roteamento entre as redes, de acordo com as regras de filtragem de pacotes. É importante evidenciar que, fluxos de roteamento entre as redes LAN e DMZ não necessitam ser enviadas para o controlador, pois todos os *hosts* dessas redes estão mapeados e o controlador já insere as regras de encaminhamento de forma proativa;
- os fluxos ARP REQUEST são destinados ao módulo específico de resposta no controlador, que por sua vez insere regras de segurança para troca de mensagens ARP e realiza a comutação de pacotes de nível 2 entre *hosts* da mesma rede;

3.2.2.1 Comutação de Pacotes Nível 2 e Segurança das Mensagens ARP

Recapitulando os fundamentos do protocolo ARP, quando um *host* deseja se comunicar com outro na mesma rede ele primeiro envia um ARP REQUEST solicitando o endereço MAC da máquina de destino. No caso do *host* confiável na rede gerenciada pelo *DmzVisor*, para estabelecer uma transmissão de dados com outras máquinas de sua rede, o *host* envia em *broadcast* o pacote ARP REQUEST, entretanto, o controlador intercepta a mensagem, impedindo inicialmente que outros *hosts* recebam. O controlador tem esse comportamento para poder verificar se o destino é confiável. Quando o ARP REQUEST chega no controlador, o módulo demultiplexador distribui para o módulo *gateway* ARP REPLY / L2 *Switch*.

Ao receber o pacote ARP REQUEST do demultiplexador, o módulo *gateway* ARP REPLY / L2 *Switch* examina se o pacote é destinado ao *gateway* ou a outro *host* da mesma rede. Caso seja para o *gateway*, o próprio controlador, constrói um pacote ARP REPLY e responde ao *host*. Já no segundo caso, o módulo analisa se o dispositivo de destino realmente está mapeado na base de dados. Comprovando que o *host* de destino está devidamente mapeado, o módulo instala regras de comutação de pacotes de nível 2, ou seja, regras que permitem máquinas confiáveis da mesma rede se comunicarem de forma livre.

Além das regras de comutação de pacotes de nível 2, são instalados fluxos para a troca segura de mensagens ARP entre esses *hosts*. Devido a vantagem de controladores SDN possuírem uma visão global da rede, estes fluxos seguros para mensagens ARP são inseridos de maneira *unicast*, dessa forma, mesmo que o destino seja *broadcast*, o *switch* comutará os pacotes

ARP apenas para o *host* que possui o IP de destino, tanto para o ARP REQUEST quanto para o ARP REPLY, baseado na porta de ingresso dos pacotes no *switch* e nos cabeçalhos MAC e IP de origem e destino. Isso mitiga ataques como envenenamento de tabela ARP e homem-do-meio. Após a instalação dessas regras, os fluxos de nível 2 entre estes *hosts* confiáveis de origem e destino, incluindo os pacotes ARP, não serão mais interceptados pelo controlador. É importante enfatizar que tanto as regras permissivas de acesso à rede como as regras de comutação e segurança ARP são aplicadas da mesma maneira para as redes LAN e DMZ.

Portanto qualquer mensagem ARP que partir de um *host* em uma das redes mantidas pelo *DmzVisor*, que viola um desses três atributos (porta do *switch*, MAC e IP), será descartada automaticamente, pois não haverá regras para fluxos com combinações diferentes do que está persistido no banco de dados. Se por exemplo houver uma clonagem ou mudança de endereço MAC ou IP, o *host* malicioso deixará de fazer parte da rede permanecendo inacessível.

Um outro ponto a evidenciar é que não só as mensagens ARP são compostas por esses fluxos de tríplices atributos (porta do *switch*, MAC e IP) de origem e de destino, mas sim todos os fluxos do *DmzVisor*, inclusive as regras de *firewall*, com a óbvia exceção do fluxo de segurança DHCP, pois nessa situação o IP da máquina não é informado no cabeçalho de origem. Diante disso, uma simples mudança de IP que pode ser até um endereço da mesma rede ou das outras adjacentes, feita manualmente pelo usuário da máquina ou por algum ataque *hacker*, terá como consequência o isolamento total deste dispositivo da rede, pois apenas o administrador tem a possibilidade de mudar as configurações de rede via interface gráfica.

3.2.3 Mudança de um *Host* para outra Rede

Até agora foram apresentados fluxos *OpenFlow* para determinar a segurança das redes, porém, esses fluxos não sofrem qualquer interferência do usuário, ou seja, são implementações padrão da aplicação proposta. Entretanto, a mudança de rede é uma ação privativa e explícita do administrador de rede que, consiste em mudar um determinado *host* que está na LAN para a DMZ, ou vice e versa. Esse gerenciamento ocorre via uma interface web amigável. O usuário informa que deseja mudar o dispositivo da rede. Após isso, é necessário que a conexão do *host* seja reinicializada, pois o protocolo DHCP implementado nas máquinas atuais não oferecem suporte à mensagem DHCPFORCERENEW que é enviada do servidor para o cliente obrigando sua renovação de endereço IP sem necessidade de reinício da conexão. Enquanto a conexão não

for reinicializada, o *host* ficará sem conectividade, evitando o risco na segurança de continuar tendo acesso a sua antiga rede.

Ao reiniciar sua conexão, o *host*, por padrão, tentará solicitar o mesmo endereço IP que possuía, enviando uma mensagem DHCPREQUEST, porém o controlador responderá com uma mensagem DHCPNAK, informando ao *host* que ele não faz mais parte daquela rede. O dispositivo libera o endereço IP da rede anterior e volta a mandar uma mensagem DHCPDISCOVER para o controlador, inicializando todo o processo de obtenção do endereço IP. Por sua vez o módulo do servidor DHCP realiza todo o processo de atribuição de endereço, já descrito anteriormente na Subseção 3.2.1, tomando como base as configurações de IP da nova rede.

Por fim, são deletadas todas as regras de *firewall* inerentes àquela máquina, uma vez que correspondiam a sua situação de visibilidade na rede anterior, além disso, as regras permissivas de acesso à rede são atualizadas para a nova rede, de acordo com o novo endereço IP vinculado. Durante a troca dessas mensagens o módulo DHCP atualiza o IP do *host* no banco de dados.

3.3 *Firewall*

Todo o tráfego de entrada e saída entre as redes organizacionais e a internet são submetidas às regras de filtragem de pacotes da aplicação. A manutenção (adição, edição e exclusão) das regras de *firewall* das redes LAN e DMZ, são de responsabilidades do administrador.

As regras impostas para a rede LAN são de maneira generalizada, ou seja, todos os *hosts* da LAN possuem as mesmas regras de *firewall*. Em contrapartida, para a rede DMZ, o usuário pode definir regras de filtragem específicas para cada uma das máquinas, isso porque, cada dispositivo da DMZ pode oferecer diferentes serviços e em razão dessa situação, a aplicação oferece essa possibilidade da instalação de regras de acordo com o serviço disponibilizado pelo *host* da zona desmilitarizada.

3.3.1 Protocolos Suportados e Formato das Regras

As regras de *firewall* do *DmzVisor* que consistem na filtragem de pacotes da camada de rede e transporte, suporta os seguintes protocolos:

- **IPv4**;

- **ICMP (ECHO REQUEST e ECHO REPLY)**;
- **TCP**;
- **UDP**.

A proposta fornece um formato de regras em alto nível, no qual, o usuário não necessita possuir conhecimentos específicos em *OpenFlow* para gerenciar o *firewall*. O módulo IPv4 / *Firewall* se encarrega de traduzir essas regras de alto nível em fluxos da tabela do plano de dados. As regras possuem o seguinte formato:

- **Direção (*Direction*)**: consiste se a regra é aplicada nas conexões de entrada (***Inbound***) ou saída (***Outbound***) em relação à rede;
- **Prioridade (*Priority*)**: ordem de aplicabilidade da regra. São oferecidos 4 níveis de prioridades: baixa (***Low***), média (***Medium***), alta (***High***) e crítica (***Critical***);
- **Protocolo (*Protocol*)**: indica em qual protocolo a regra deve ser empregada;
- **Porta (*Port*) TCP/UDP**: número da porta de conexão TCP ou UDP. Se não for definido o número da porta a regra será aplicada em toda a extensão do protocolo. Caso a regra seja para outros protocolos (ICMP ou IPv4) este campo deve ser ignorado;
- **Ação (*Action*)**: A atuação que o plano de dados deve ter com relação à regra, ou seja, se encaminha (***Forward***) o pacote ou descarta (***Drop***);

3.3.2 Aplicação das Regras

A aplicação das regras acontece por intermédio da interface gráfica. Quando o usuário adiciona uma regra, o sistema web envia essa regra para o módulo mantenedor de regras do controlador. Por sua vez, o mantenedor encaminha a regra para o módulo IPv4 / *Firewall* que traduz e aplica em forma de fluxos *OpenFlow*. Conforme exposto anteriormente, quando o *host* faz parte da DMZ as regras são instaladas apenas para aquela determinada máquina, já para a rede LAN as regras são adicionadas para todos os *hosts* da rede. Ao chegar um novo *host* na LAN, o módulo DHCP por intermédio do IPv4 / *Firewall* instala as regras LAN já existentes no banco.

Com relação à comunicação com dispositivos externos as redes gerenciadas pelo *Dmz-Visor*, quando as máquinas da LAN ou DMZ desejam se comunicar com um *host* externo, o *gateway* da WAN envia um ARP REQUEST para o primeiro elemento da rede, sendo normal-

mente o roteador do provedor (no experimento realizado neste trabalho, por simplificação, a internet é representada por um computador externo, como descrito no Capítulo 4). Porém ao receber o ARP REPLY do exterior, o controlador faz uma verificação de segurança nos cabeçalhos de origem, para comparar se o dispositivo exterior está tentando utilizar algum endereço IP ou MAC que é considerado confiável nas redes protegidas pelo *DmzVisor*. Se de fato o *host* da web estiver usando um MAC ou IP das redes da organização, o controlador descarta o pacote, caso contrário, o controlador salva o IP e MAC desta máquina e faz o roteamento com base nas regras de *firewall*.

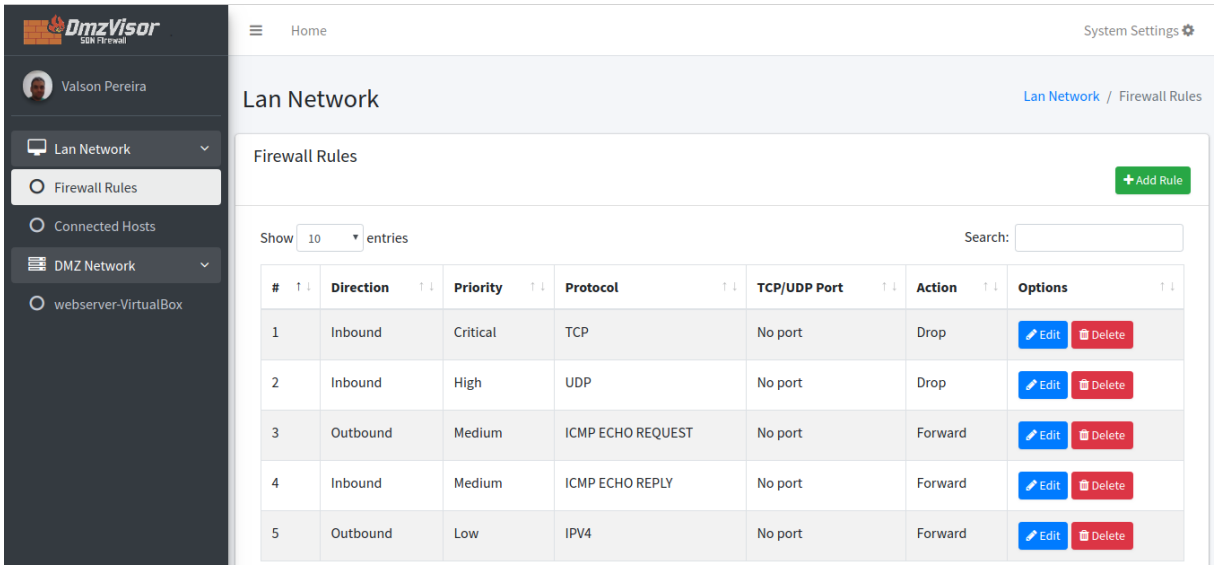
3.3.2.1 Particularidade na Aplicação de Regra TCP

Ao adicionar pela interface web, uma regra TCP na direção *inbound* para descartar tentativas de estabelecimento de conexões de entrada, o controlador precisa instalar no plano de dados duas regras de fluxo para atingir esse objetivo: a primeira com uma prioridade mais alta, que bloqueia pacotes TCP com a flag SYN, impedindo solicitação para início de uma conexão; entretanto, a segunda com uma prioridade mais baixa, é criada para permitir a chegada de pacotes TCP com as flags SYN+ACK e ACK. Isso impede que não possa ser iniciada nenhuma conexão com o dispositivo, mas que o dispositivo em questão tenha condições de receber a resposta de conexões que ele próprio iniciou.

3.4 Sistema Web para Gerência do *Firewall*

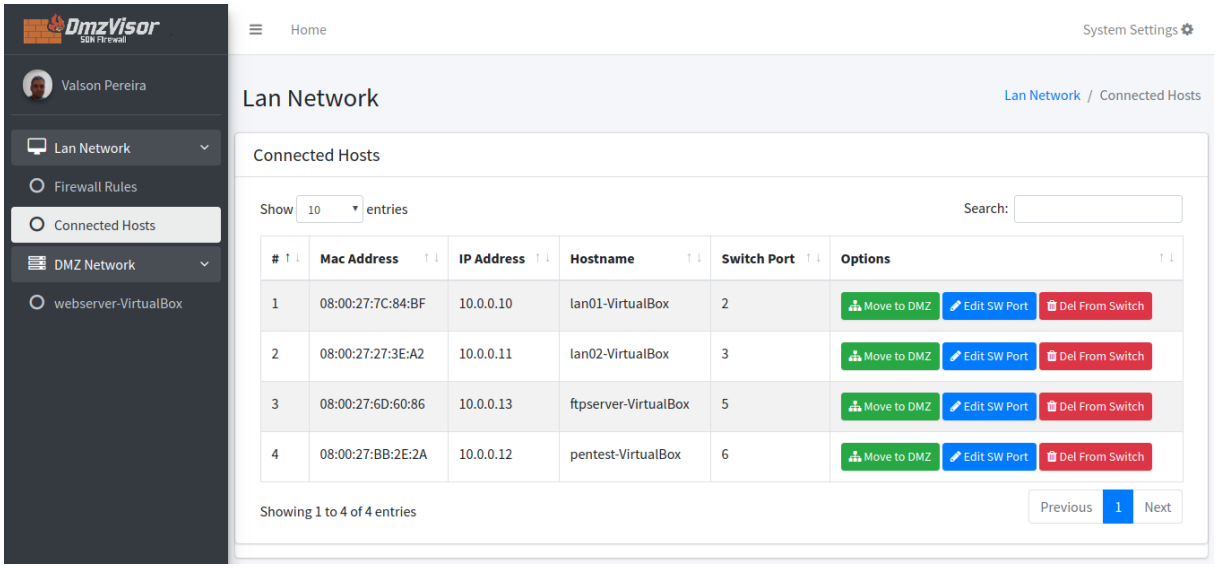
O sistema web desenvolvido para gerenciar o *DmzVisor* fornece ao usuário uma interface amigável na qual, tarefas administrativas podem ser executadas com facilidade. O sistema web se comunica com o controlador através do módulo mantenedor de regras. O protótipo apresenta em sua tela principal as regras de *firewall* padrão que já oferecem um bom nível de segurança para a rede LAN, conforme pode ser vista na Figura 3.3. As regras padrão da LAN consistem em: bloquear conexões de entrada TCP e UDP (regras 01 e 02), permitir o envio de requisições e recebimento de respostas de pacotes ICMP (ping) (regras 02 e 03) e encaminhar IPv4 para fora da rede (regra 04). Por meio desta interface o usuário tem a possibilidade adicionar, editar e excluir regras.

Figura 3.3 – Tela principal do *DmzVisor* com regras padrão de *firewall* para rede LAN



Já a aba ***Connected Hosts*** exibe uma tela com todas as informações dos *hosts* que estão conectados à rede LAN, como pode ser observado na Figura 3.4. Além disso é possível mover determinado *host* para DMZ, atualizar a porta do *switch* caso necessário, no qual todas as regras de segurança do *host* serão transferidas para a nova porta de destino. Também é permitido a exclusão de um *host* da rede, que consiste em deletar todas as regras da máquina, liberando a porta para o uso de um novo dispositivo na rede.

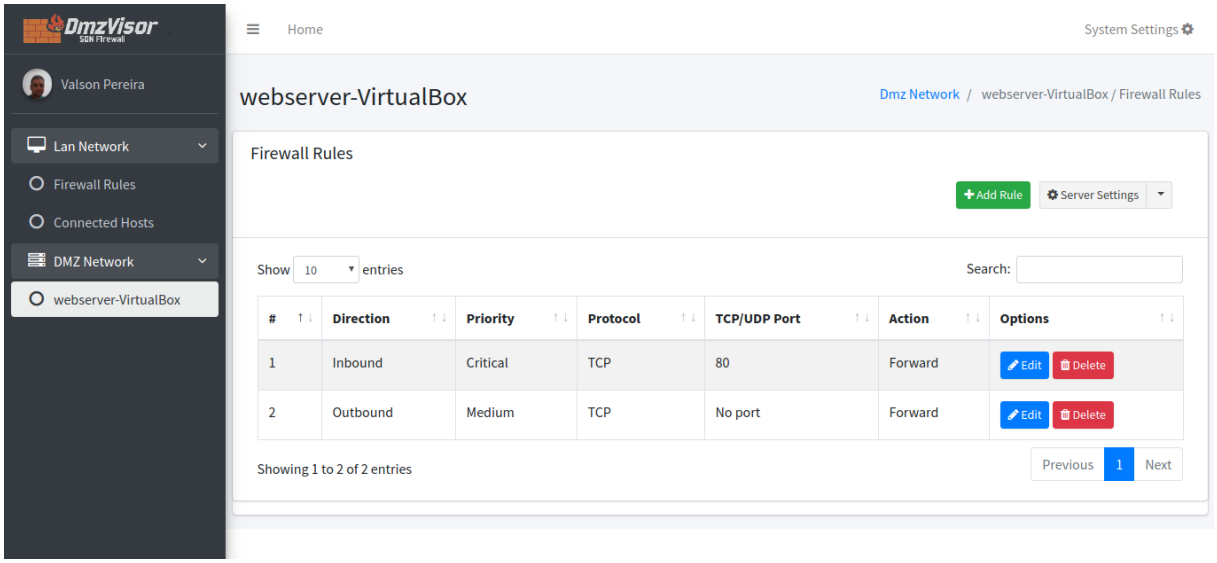
Figura 3.4 – Interface com todas as máquinas conectadas na rede LAN



Em ***DMZ Network*** são listados individualmente os *hosts* servidores da DMZ, bem como suas regras de *Firewall*. A DMZ ainda traz em ***Server Settings*** as mesmas funcionalidades da aba *Connected Hosts* da rede LAN, como mover o servidor para a outra rede, neste caso a LAN. Vale destacar que não é permitido deletar o servidor do *switch* por motivos de segurança.

Entretanto é possível atualizar a porta do servidor no *switch*, além de verificar suas configurações de rede, como denota a Figura 3.5.

Figura 3.5 – Interface do usuário com regras de um servidor instalado na rede DMZ.



3.5 Resumo do Capítulo

Este capítulo apresentou a proposta deste trabalho implementado inteiramente com ferramentas *open source*. Foi visto que além de ser desenvolvido mecanismos de filtragem de pacotes para as camadas de rede e transporte, o protótipo também foi implementado para oferecer mais segurança por meio de isolamento de redes e proteção à protocolos vulneráveis pela ausência de autenticação, a exemplo do ARP e DHCP.

Outro fator relevante abordado na metodologia desta aplicação foi a simplicidade disponibilizada ao usuário. Por meio da interface gráfica web, o administrador é capaz de gerenciar o software controlador da rede, além disso, é possível criar regras de *firewall* em alto nível, não sendo necessário o usuário ter conhecimento sobre o protocolo *OpenFlow*.

4 Avaliação da Proposta e Resultados

Neste capítulo será discutido a avaliação e os resultados obtidos nesta monografia. Na Seção 4.1 mostra uma descrição sucinta do ambiente de avaliação, bem como da composição dos cenários utilizados. Na Seção 4.2 é apresentado a avaliação, os resultados e sua discussão. Por fim a Seção 4.3 traz um resumo deste presente capítulo.

4.1 Descrição do Ambiente de Avaliação

O *DmzVisor* foi avaliado em um ambiente composto por 6 máquinas virtuais Linux Lubuntu 16.04 utilizando o virtualizador VirtualBox (ORACLE, 2018). Todas as máquinas foram conectadas em um único *software switch*, o *Open vSwitch* 2.5.4. O protótipo foi hospedado localmente, ou seja, na mesma máquina que virtualiza o ambiente, um computador Core i5 com 8 GB de RAM com sistema operacional Linux Ubuntu Mate 16.04.

Para a realização dos experimentos foram elaborados dois cenários, cenário 01 com foco no teste da segurança da rede LAN em relação às outras duas (DMZ e Internet) e o cenário 02 focado nos testes da rede DMZ em relação à LAN e à internet. Os dois cenários mapeados pelo módulo DHCP do *DmzVisor* são detalhados nas Tabelas 4.1 e 4.2 respectivamente, assim como também no diagrama de rede na Figura 4.1. As informações dos *gateways* virtuais foram apresentadas na Tabela 3.1.

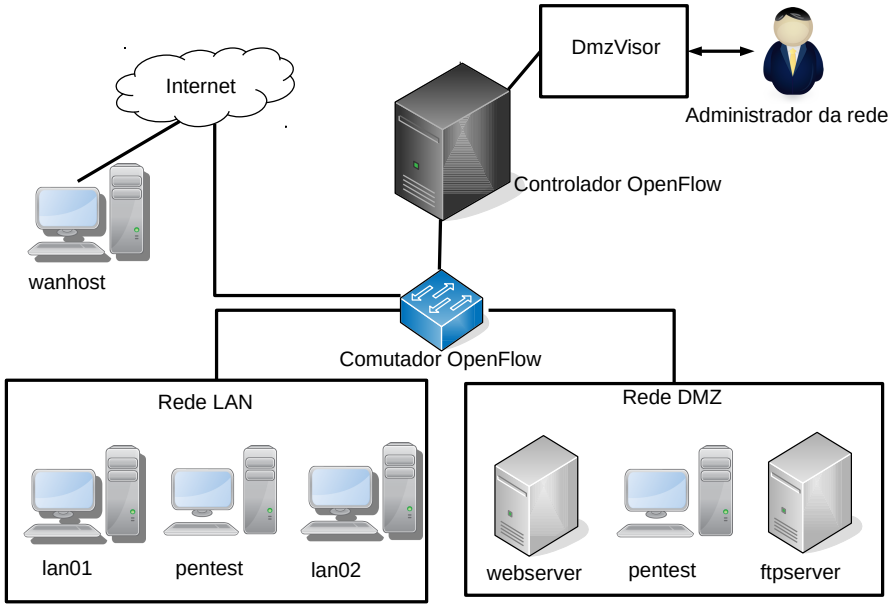
Tabela 4.1 – Composição do cenário 01 para testes na rede LAN.

Rede	***Hosts***			
	Mac	**IP**	**Hostname**	**Porta do *Switch***
Internet WAN	08:00:27:38:8D:01	172.16.0.2	wanhost-VirtualBox	1 (WAN)
LAN	08:00:27:7C:84:BF	10.0.0.10	lan01-VirtualBox	2
	08:00:27:27:3E:A2	10.0.0.11	lan02-VirtualBox	3
	08:00:27:BB:2E:2A	**10.0.0.12**	**pentest-VirtualBox**	**6**
DMZ	08:00:27:93:A4:69	192.168.0.10	webserver-VirtualBox	4
	08:00:27:6D:60:86	192.168.0.11	ftpserver-VirtualBox	5

Tabela 4.2 – Composição do cenário 02 para testes na rede DMZ.

Rede	***Hosts***			
	Mac	**IP**	**Hostname**	**Porta do *Switch***
Internet WAN	08:00:27:38:8D:01	172.16.0.2	wanhost-VirtualBox	1 (WAN)
LAN	08:00:27:7C:84:BF	10.0.0.10	lan01-VirtualBox	2
	08:00:27:27:3E:A2	10.0.0.11	lan02-VirtualBox	3
DMZ	**08:00:27:BB:2E:2A**	**192.168.0.2**	**pentest-VirtualBox**	**6**
	08:00:27:93:A4:69	192.168.0.10	webserver-VirtualBox	4
	08:00:27:6D:60:86	192.168.0.11	ftpserver-VirtualBox	5

Figura 4.1 – Topologia dos cenários de avaliação da proposta.



Como pode ser observado nessas tabelas e figura mencionadas, a única diferença da composição dos cenários é a localização do *host* **pentest-VirtualBox**, que foi uma máquina utilizada para executar os testes de segurança interna de cada uma das redes, como por exemplo, o teste das mensagens ARP e falsificação dos endereços MAC/IP.

O **webserver-VirtualBox** possui um servidor web na escuta da porta tcp/80 e o **ftpserver-VirtualBox** possui um serviço *Secure File Transfer Protocol* (SFTP) que utiliza a porta tcp/22. Por fim foi instalado também um servidor web na porta tcp/80 no *host* correspondente que simboliza a internet nos experimentos, o **wanhost-VirtualBox**. Como esperado, o servidor DHCP implementado no controlador atende apenas as requisições locais dos *hosts* da LAN e DMZ, dessa forma, o endereço IP do *host* correspondente foi configurado manualmente. O controlador instala as regras de roteamento baseado nas regras de *firewall* para os pacotes vindos das redes externas com destino para as redes da organização (LAN e DMZ) e responde as

requisições ARP recebidas pela porta WAN no *gateway* virtual, assim como também é capaz de enviar requisições ARP para os componentes de rede conectados na porta WAN.

Os experimentos conduzidos em cada cenário buscaram avaliar os seguintes aspectos:

- **Conectividade dentro da mesma rede**: verificar a conectividade proporcionada pela comutação de nível 2 entre os *hosts* confiáveis da mesma rede que ingressaram de forma autêntica via DHCP;
- **segurança na troca de mensagens ARP**: realizar uma breve auditoria utilizando ferramentas que atacam as vulnerabilidades do protocolo ARP;
- **isolamento das redes**: analisar o comportamento da rede diante de uma situação, na qual um *host* da organização que teve a sua segurança comprometida tenta passar para a outra rede utilizando endereço IP e MAC falsificados;
- **Regras de *firewall***: aferir a eficácia das regras de *firewall* impostas em cada uma das redes.

4.2 Avaliação e Resultados

4.2.1 Teste de Conectividade na mesma Rede

Para testar as regras de comutação nível 2 e a conectividade entre os *hosts* da mesma rede assim como a comunicação com cada *gateway* foi utilizado o comando ping, conforme demonstra a Figura 4.2, que mostra o ping entre os *hosts* lan01 e lan02, a Figura 4.3 que apresenta o ping entre o webserver e o ftpserver, por fim, as Figuras 4.4 e 4.5 mostram o ping a partir dos *hosts* lan01 e webserver para seus respectivos *gateways* virtuais.

Figura 4.2 – Conectividade entre os dois *hosts* da LAN.

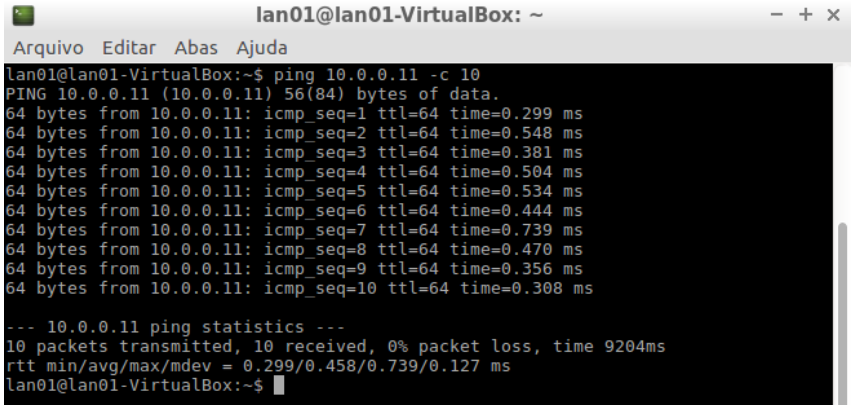


Figura 4.3 – Conectividade entre servidores da DMZ.

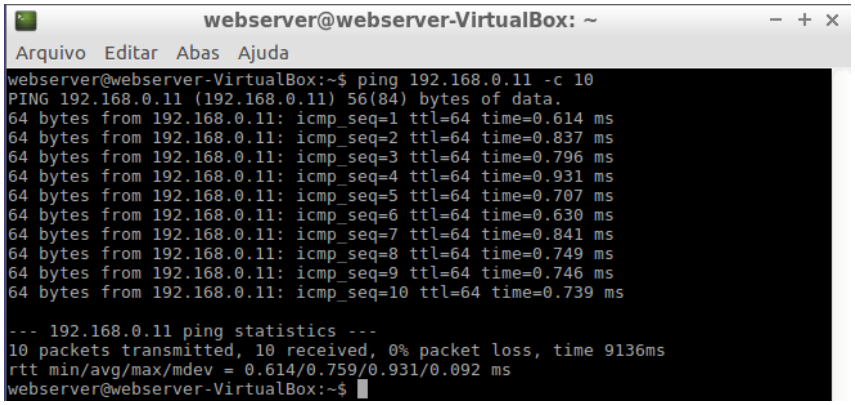


Figura 4.4 – Conectividade entre *hosts* e o *gateway* virtual da rede LAN.

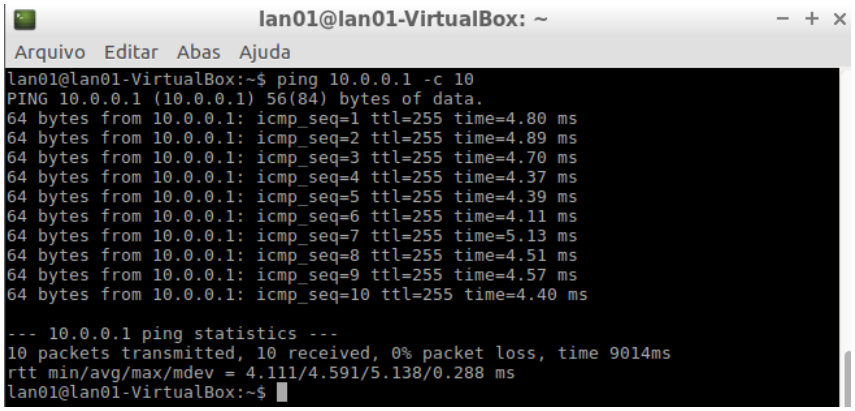
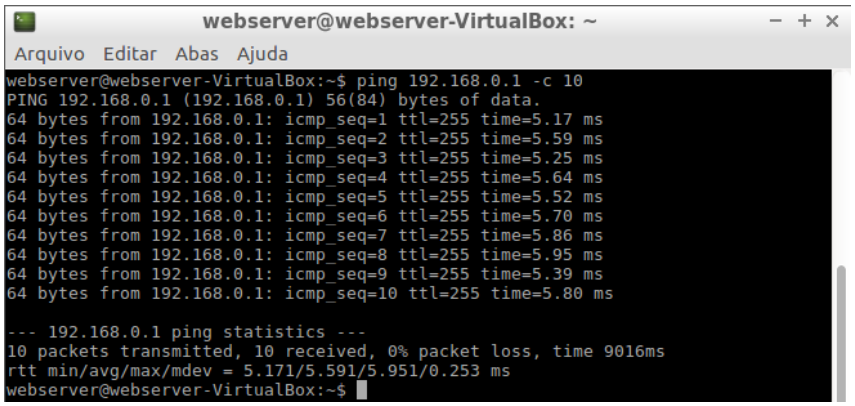


Figura 4.5 – Conectividade entre servidores e o *gateway* virtual da rede DMZ.

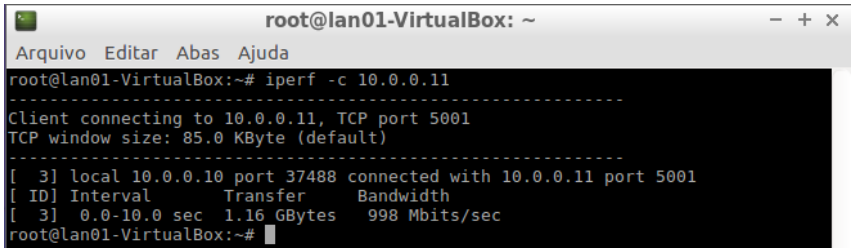


Diante dos resultados de conectividade apresentados, percebe-se pelas estatísticas fornecidas pelo ping no campo **avg** que, em ambas as redes o tempo médio do ping entre *hosts* (0,608 ms) é bastante inferior comparado ao tempo do ping entre os *hosts* e o *gateway* virtual (5,091 ms) representado pelo controlador. Isso acontece porque o encaminhamento do ping entre as máquinas é feito inteiramente pelo plano de dados do *Open vSwitch*. Em virtude disso, o encaminhamento de pacotes é mais rápido e eficiente pelo fato do plano de dados ser um

módulo que trabalha no espaço do kernel, esse caminho rápido é denominado de *fastpath*. Em compensação, o encaminhamento do pacote feito para o controlador, no caso do ping para o *gateway* virtual, é denominado *slowpath* pois seu envio passa pelo controlador, que está no espaço do usuário submisso a disputa dos recursos providos pelos algoritmos de escalonamento do sistema operacional.

Para fornecer resultados de conectividade e desempenho entre *hosts* da mesma rede, no cenário 01 foi levantado um servidor iperf (NLANR/DAST, 2018) no *host* lan02. O iperf é uma ferramenta de teste de desempenho de redes. O lado cliente do iperf foi executado no lan01, conforme demonstra a Figura 4.6.

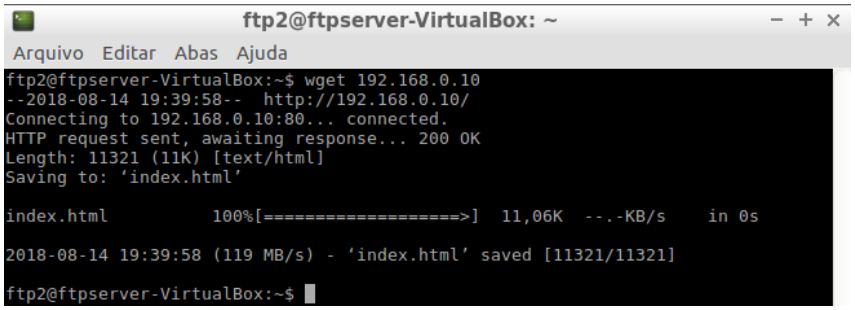
Figura 4.6 – Avaliação de desempenho entre os *hosts* da rede LAN, através da ferramenta iperf.



Já no cenário 02 foi feita a utilização do serviço entre servidores da rede DMZ. Na Figura 4.7 pode-se observar o *host* webserver acessando o servidor SFTP provido pelo ftpserver e a Figura 4.8 apresenta o *host* ftpserver acessando o servidor do webserver.

Figura 4.7 – Acesso remoto ao servidor SFTP a partir do servidor Web.

Figura 4.8 – Acesso remoto ao servidor Web a partir do servidor SFTP.

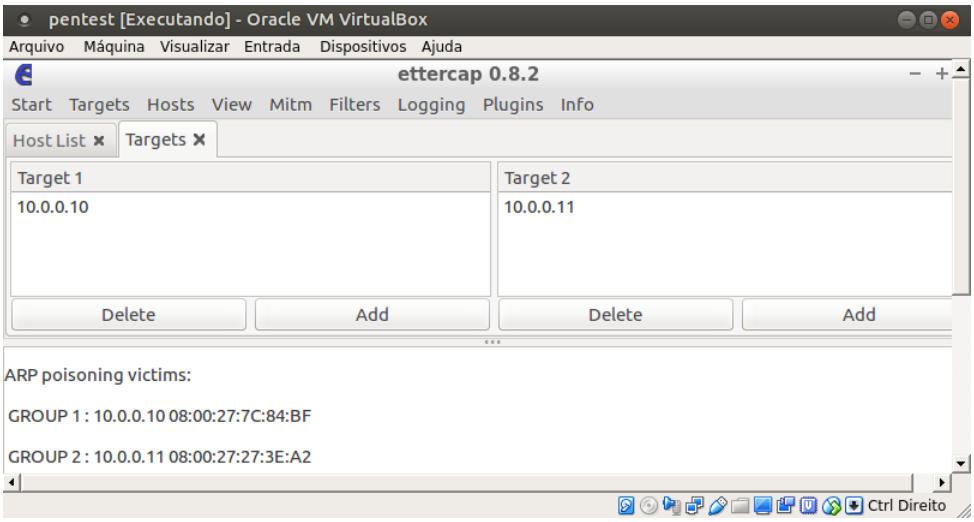


A conectividade dentro da mesma rede, demonstra que o *DmzVisor* realiza também comutação de nível 2, normalmente como um *switch* comum, onde a troca de pacotes entre as máquinas acontece de forma livre.

4.2.2 Verificação da segurança de Mensagens ARP

Para a verificação de segurança das mensagens ARP e o impedimento do ataque do homem-do-meio foi utilizada a ferramenta Ettercap (ETTERCAP, 2018), instalada no *host* pentest. Para o cenário 01, foram selecionados como alvos (*targets*) do ataque os *hosts* lan01 e lan02, identificados por alvo 1 e alvo 2, com seus respectivos IPs, conforme denota a Figura 4.9.

Figura 4.9 – *Host* pentest executando o Ettercap com alvos da rede LAN.



Para comprovar de fato que o Ettercap envia pacotes ARP falsificados para ambos os alvos, foi utilizado o Wireshark (WIRESHARK, 2018) para verificar o tráfego ARP da porta 6, na qual está conectado o *host* pentest. A Figura 4.10 mostra o pacote ARP Reply capturado na porta 6 com destino ao lan01, nesse pacote o invasor está tentando se identificar associando o seu endereço MAC ao IP do lan02. De forma semelhante, na Figura 4.11 o pentest tenta se passar por lan01 para enganar o lan02. Conforme visto nessas ilustrações, o Ettercap realmente distribui mensagens ARP Reply maliciosas informando que os endereços 10.0.0.10 e 10.0.0.11 estão no seu endereço MAC com intuito de envenenar as tabelas ARP das máquinas alvejadas.

Todavia, essas mensagens maliciosas não chegam nos seus alvos, são descartadas no plano de dados do *Open vSwitch*. Como resultado, as tabelas dos alvos permanecem seguras e sem alterações fraudulentas, ou seja, o IP 10.0.0.10 continua associado ao endereço MAC

08:00:27:7c:84:bf e o IP 10.0.0.11 ainda está vinculado ao endereço MAC 08:00:27:27:3e:a2, correspondendo aos endereços dos *hosts* lan01 e lan02 respectivamente, de acordo com a Tabela 4.1. Inclusive percebe-se também que os alvos mantém o IP e MAC original do atacante, comprovando a conectividade existente entre os mesmos.

Figura 4.10 – Tabela ARP lan01 e pacote ARP Reply malicioso.

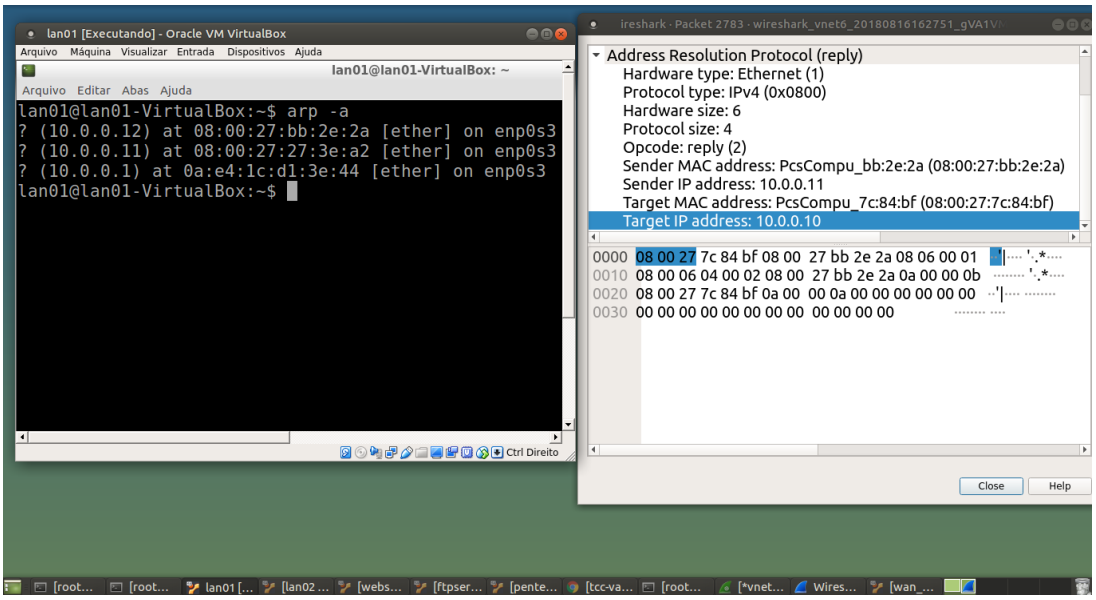
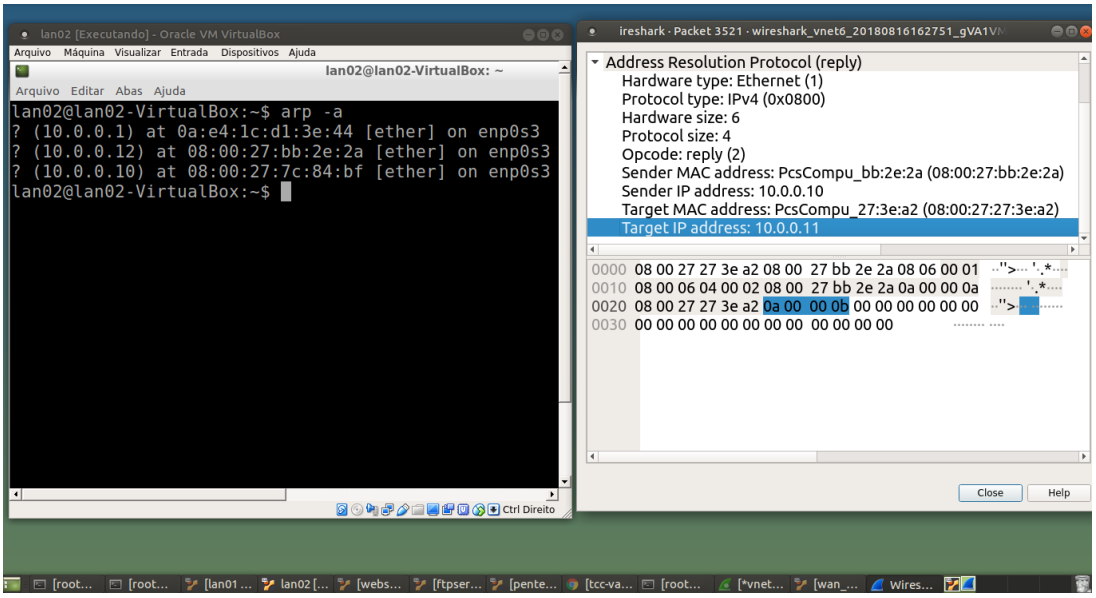


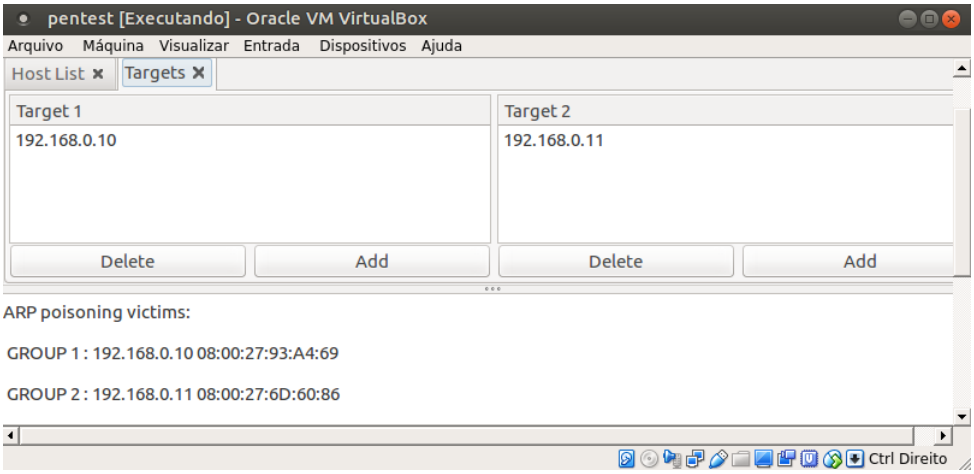
Figura 4.11 – Tabela ARP lan02 e pacote ARP Reply malicioso.



Isso ocorre devido ao *DmzVisor*, instalar regras determinando que o plano de dados apenas envie às portas de destino, pacotes ARP que possuam em seus cabeçalhos a combinação de porta do *switch*, endereços MAC e IP de acordo com o mapeamento realizado na base de dados. Portanto, o plano de dados descarta qualquer pacote ARP malicioso que seria distribuído na rede.

A segurança das mensagens ARP também foi verificada no experimento do cenário 02, tendo como alvos do ataque os *hosts* webserver e ftpserver da DMZ, identificados por alvo 1 e alvo 2, com seus respectivos IPs, como denota a Figura 4.12.

Figura 4.12 – *Host* pentest executando o Ettercap com alvos da rede DMZ.



A Figura 4.13 mostra o pacote ARP Reply capturado na porta 6 com destino ao webserver, nesse pacote o pentest tenta se identificar vinculando o seu endereço MAC ao IP do ftpserver. De forma análoga, é apresentado na Figura 4.14 que o pentest tenta se apresentar com o endereço IP do webserver para enganar o ftpserver. De acordo com o que pode ser observado nessas ilustrações, o Ettercap de fato distribui as mensagens ARP Reply maliciosas informando que os endereços 192.168.0.10 e 192.168.0.11 estão associados a interface com seu endereço MAC com intuito de envenenar as tabelas ARP das máquinas alvejadas.

Os experimentos do cenário 02, também confirmaram a eficácia do *DmzVisor* em mitigar os ataques com pacotes ARP maliciosos, uma vez que conseguiu impedir o envenenamento das tabelas ARP dos servidores na DMZ. Essas mensagens maliciosas não são comutadas para seus alvos, porque são descartadas no *switch*.

Dessa forma, como as tabelas dos alvos não foram afetadas, o IP 192.168.0.10 continua associado ao endereço MAC 08:00:27:93:a4:69 e o IP 192.168.0.11 ainda está vinculado ao endereço MAC 08:00:27:6d:60:86, correspondendo aos endereços dos *hosts* webserver e ftpserver respectivamente, de acordo com a Tabela 4.2.

Figura 4.13 – Tabela ARP webserver e pacote ARP Reply malicioso.

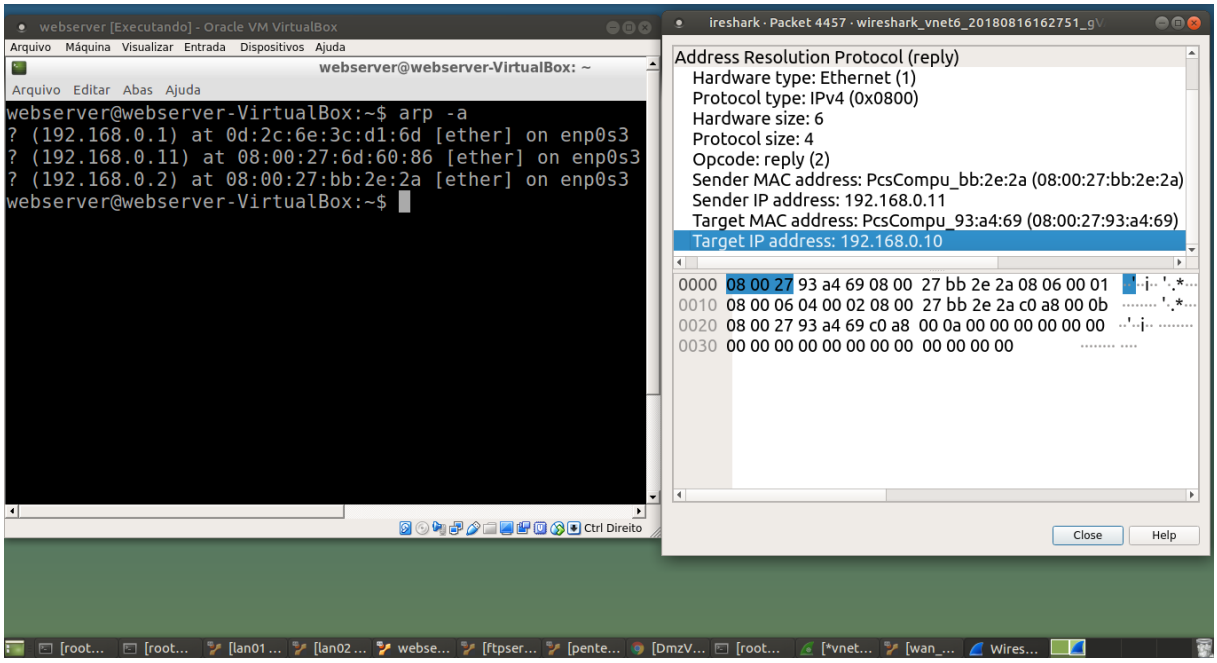
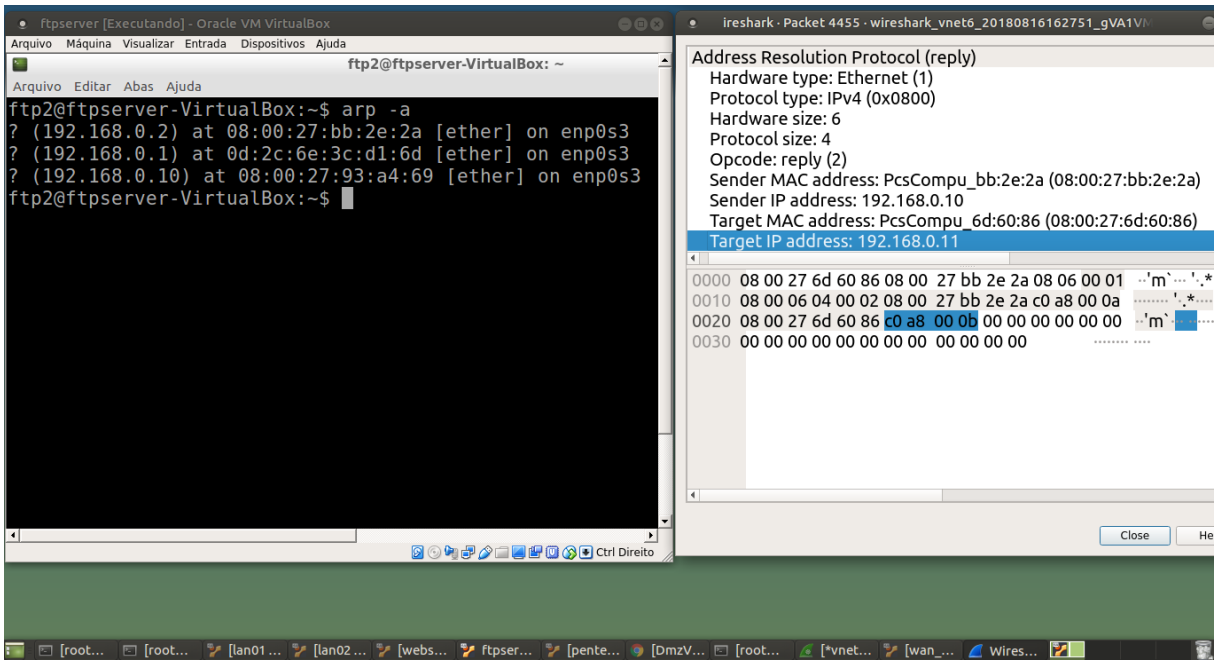


Figura 4.14 – Tabela ARP ftpserver e pacote ARP Reply malicioso.



4.2.3 Avaliação do Isolamento entre Redes e Proteção DHCP

A avaliação do isolamento das redes consistiu na análise de uma situação simulada na qual o *host* pentest que era um computador legítimo da rede, teve sua segurança comprometida devido a algum ataque *hacker*.

Nesse sentido, foi falsificado o IP e o endereço MAC da máquina para tentar ingressar

na rede oposta com um IP supostamente válido. Isto é, no cenário 01 o pentest está localizado inicialmente na LAN, posteriormente, após ser comprometido, o IP da máquina pentest foi alterado para um IP da DMZ. No cenário 02 o pentest começa inicialmente como *host* válido na DMZ e seu IP foi trocado para um IP da LAN. O endereço IP foi mudado manualmente, em contrapartida o MAC foi alterado com o auxílio da ferramenta GNU Mac Changer (ORTEGA, 2018).

De acordo o que demonstra as Figuras 4.15 e 4.16 do cenário 01, mesmo com um IP válido da rede DMZ e conectado no mesmo *switch*, o *host* pentest que originalmente pertence à rede LAN, não responde a nenhuma comunicação realizada pelos servidores da DMZ. Isso se deve ao fato dele ter violado as regras de segurança da rede, pois no plano de dados não há rotas para ele com esse endereço MAC e IP do tipo DMZ. Sendo assim, ele permanece inacessível por qualquer outro *host*. Cabe salientar que a recíproca é verdadeira, ou seja, a máquina maliciosa também não consegue acessar os outros dispositivos na rede.

Figura 4.15 – Conectividade sem sucesso entre servidor web da DMZ e *host* LAN malicioso.

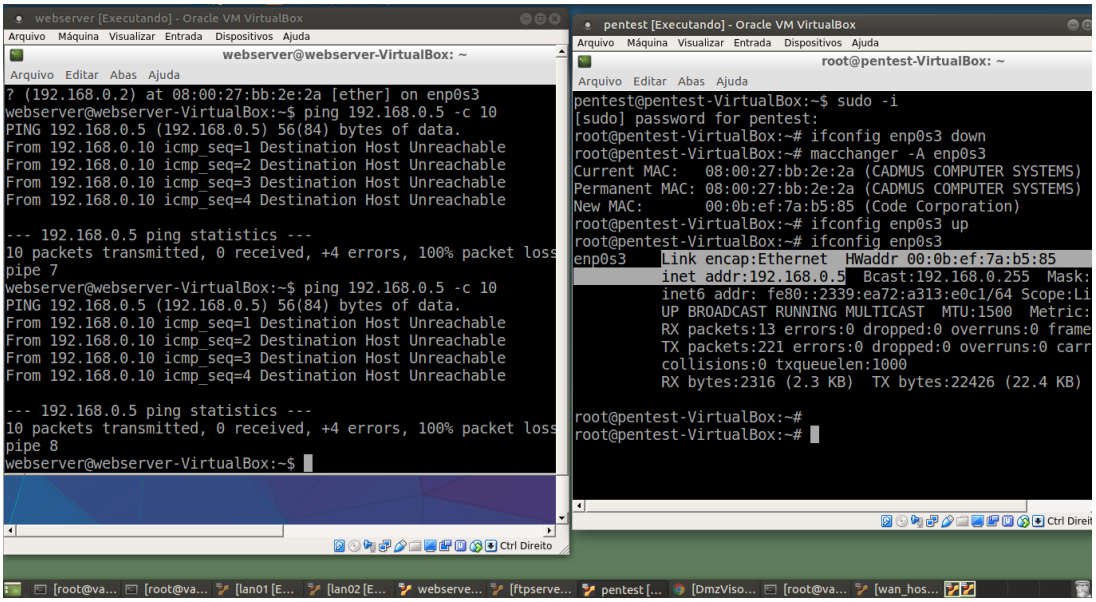
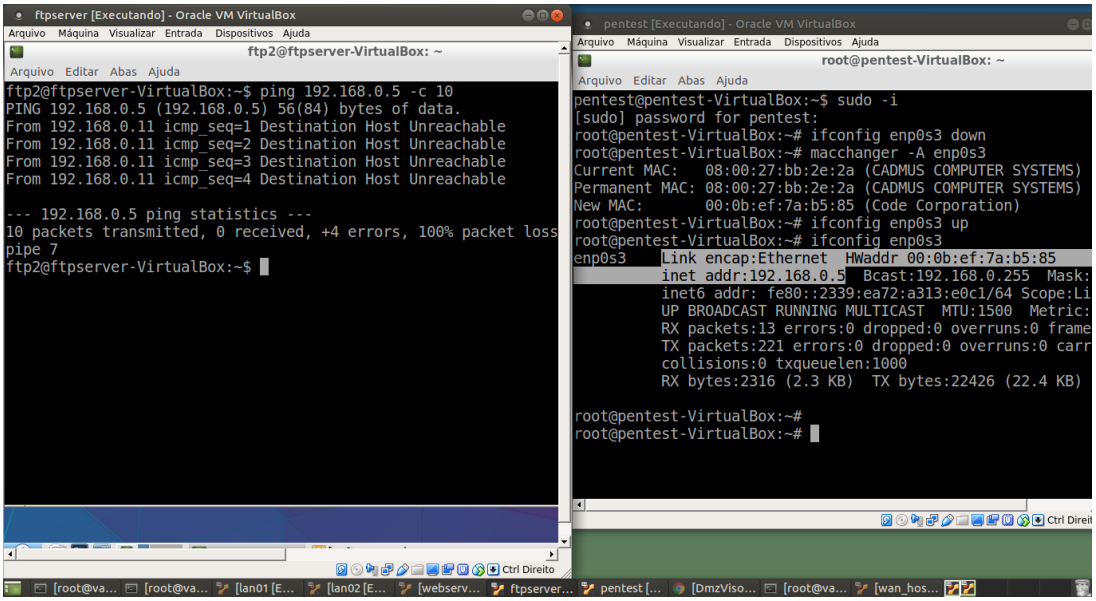


Figura 4.16 – Conectividade sem sucesso entre servidor FTP da DMZ e *host* LAN malicioso.



No cenário 02 foi realizado o mesmo experimento, como é demonstrado nas Figuras 4.17 e 4.18. Desse modo, o *host* pentest, até então uma máquina legítima da DMZ, tem seu IP e MAC modificados para endereços válidos da rede LAN. Os resultados obtidos são os mesmos do cenário 01, o que demonstra que a segurança de isolamento se aplica em ambas as redes da organização.

Figura 4.17 – Conectividade sem sucesso da máquina lan01 e *host* malicioso da DMZ.

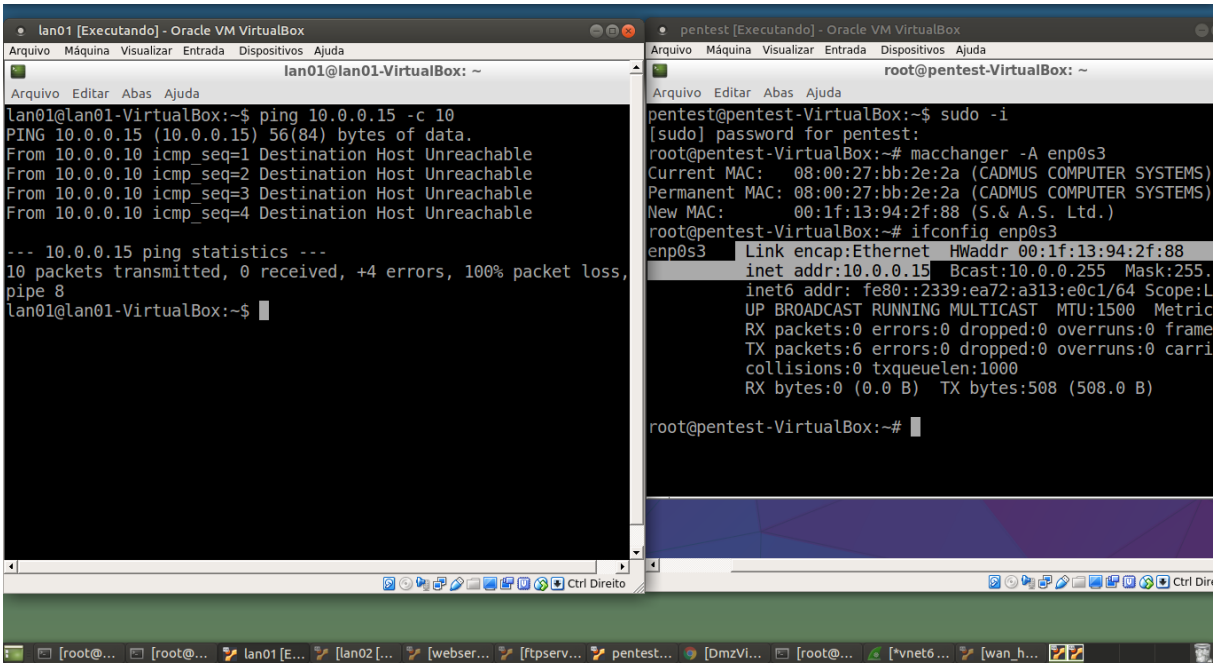
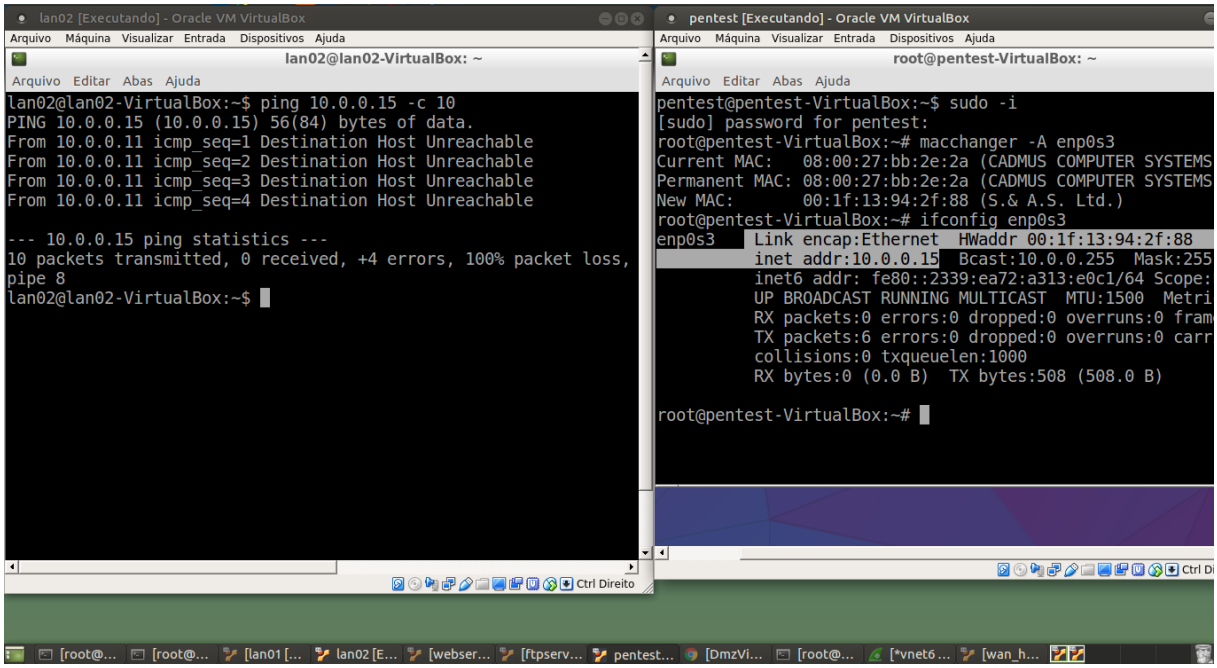
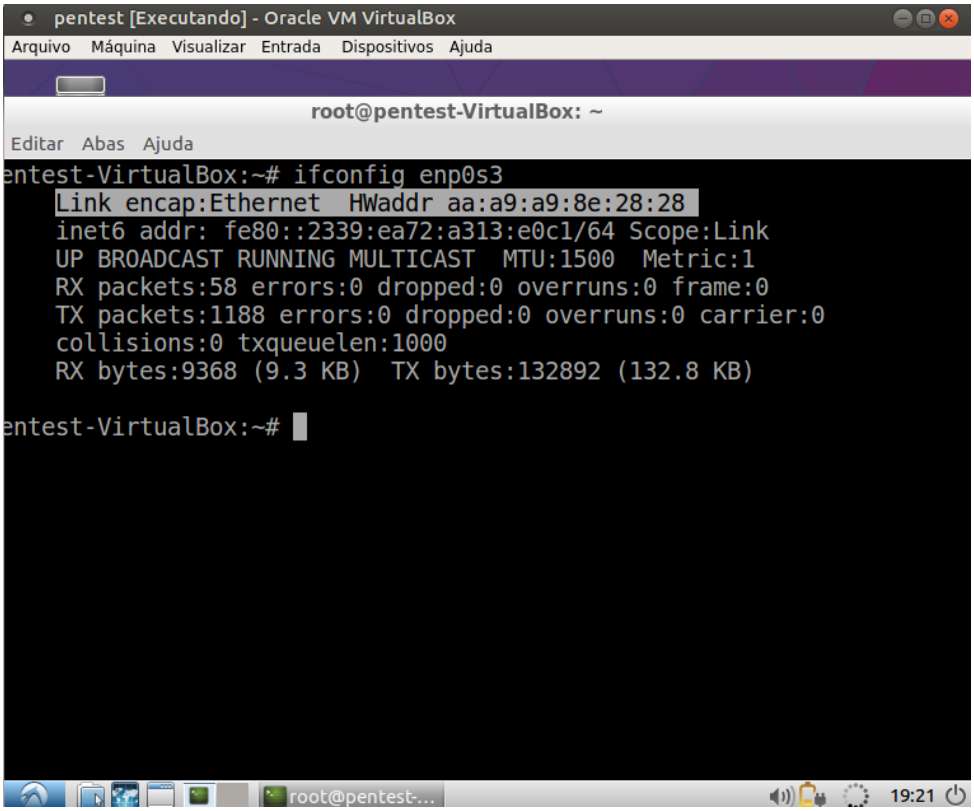


Figura 4.18 – Conectividade sem sucesso da máquina lan02 e *host* malicioso da DMZ.



A máquina maliciosa também não consegue obter endereço via DHCP ao tentar reiniciar sua conexão com um endereço MAC falsificado, como está ilustrado na Figura 4.19. Diante disso pode ser constatado a eficácia das regras de segurança DHCP que vincula o MAC original mapeado na base de dados à porta do *switch*, denotados na Tabela 3.3 do Capítulo 3.

Figura 4.19 – Tentativa mal sucedida de obter endereço IP com MAC falsificado.



4.2.4 Teste das Regras de *Firewall*

O teste das regras de *firewall* de cada cenário buscou verificar principalmente a filtragem de portas de cada uma das redes da organização (LAN e DMZ) em relação de uma com a outra e à internet, representada pelo wanhost. Para escanear as portas abertas de cada máquina das redes foi utilizado a ferramenta hping3 (HPING, 2014).

Para o primeiro cenário foram avaliadas as regras de *firewall* padrão para a rede LAN, que foram apresentados na Seção 3.4 do Capítulo 3 da proposta, na Figura 3.3. Para verificar a eficácia da regra 01 da rede LAN, na qual bloqueia todos os pacotes com a flag TCP_SYN, foi executado localmente o hping3 com a flag `-syn` ativada em todas as 65535 portas, para mostrar as portas abertas de cada *host* da LAN, após isso foi executado o hping3 nos *hosts* das outras redes em direção à LAN, conforme é apresentado nas Figuras 4.20 e 4.21.

Figura 4.20 – *Host* lan01 com portas TCP abertas e escaneamento feito pelo wanhost (rede externa).

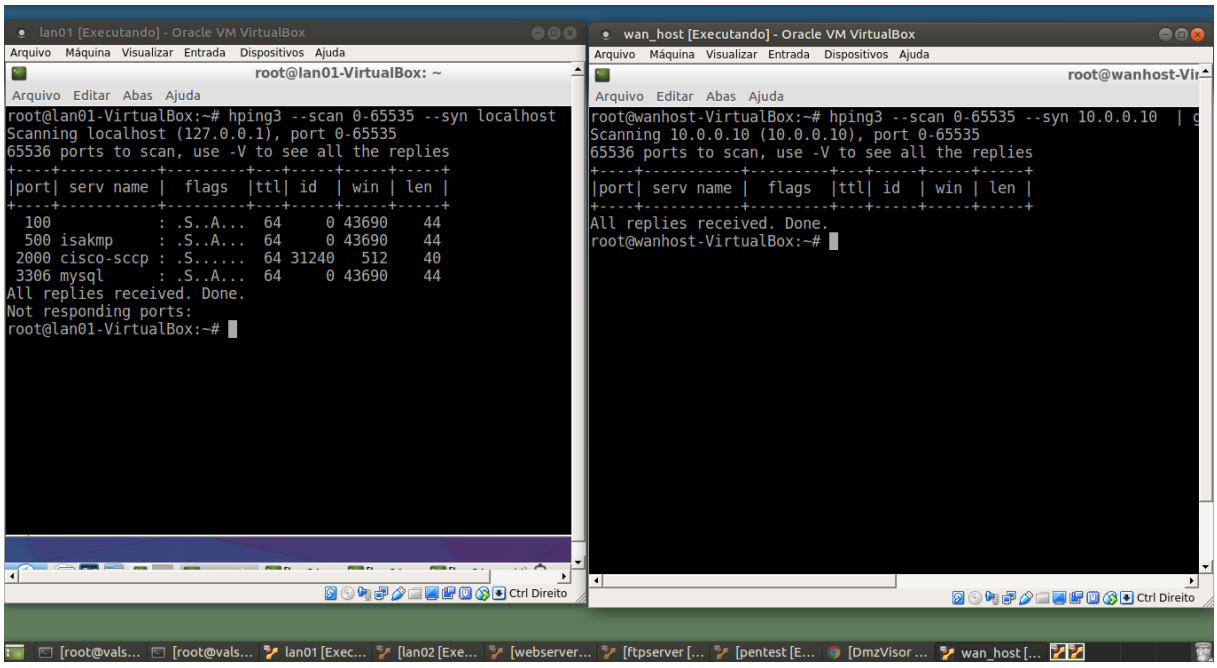
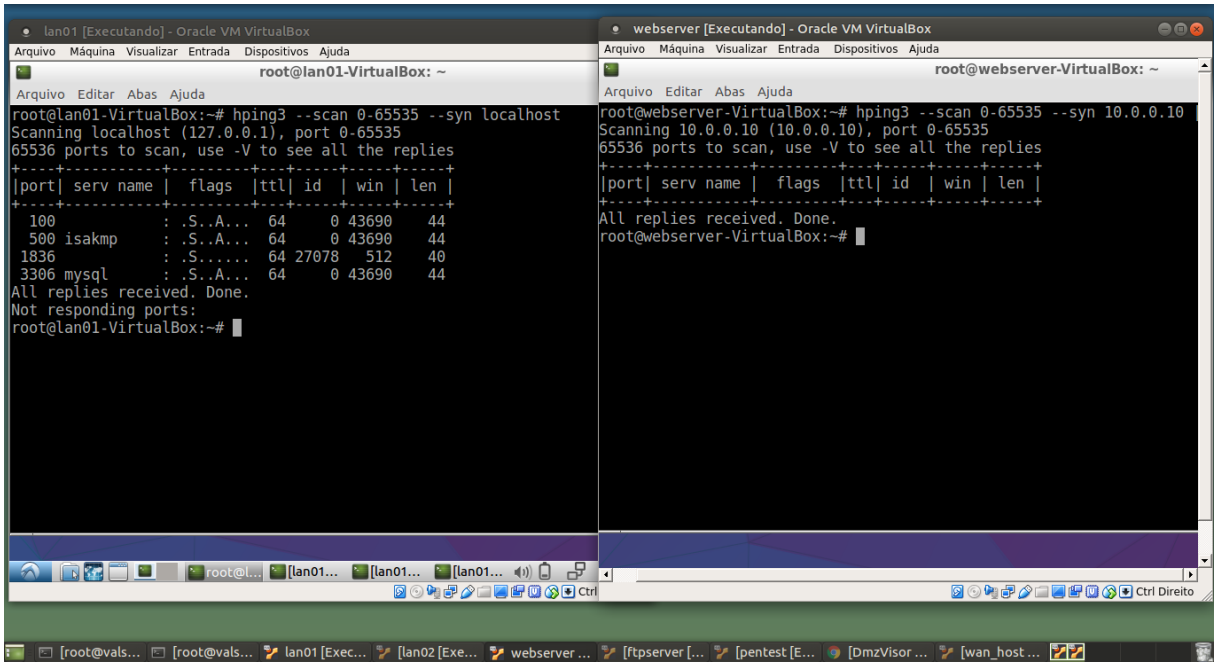


Figura 4.21 – *Host* lan01 com portas TCP abertas e escaneamento feito pelo webserver (rede DMZ).



Ao observar as figuras mencionadas, percebe-se que mesmo com portas abertas os *hosts* LAN estão protegidos em relação à DMZ e à rede externa. O *DmzVisor* demonstra que a regra 01 da LAN realiza a filtragem adequada de pacotes TCP das outras redes que tentam iniciar uma conexão com máquinas internas da LAN. Conclui-se ainda que, o administrador de rede pode instalar um servidor interno para ser acessado localmente pelas as máquinas LAN, pois estará protegido em relação a rede externa e a DMZ.

Apesar de não ser enxergada as portas das máquinas na rede LAN, nada impede que esses dispositivos acessem serviços em outras redes, isso se deve às regra 05 e 01, que permite a saída de pacotes IPv4 da LAN e a entrada de pacotes TCP desde que não seja com a flag SYN ativada, isto é apresentado nas Figuras 4.22, 4.23 e 4.24.

Figura 4.22 – *Host* lan01 acessando serviço da rede externa.

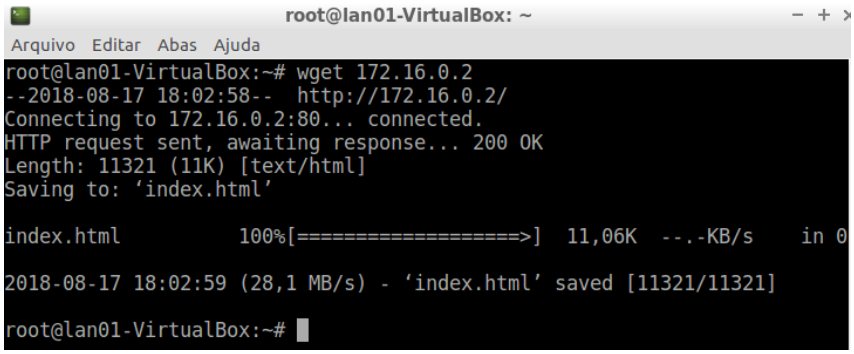


Figura 4.23 – *Host* lan01 acessando o servidor Web da DMZ.

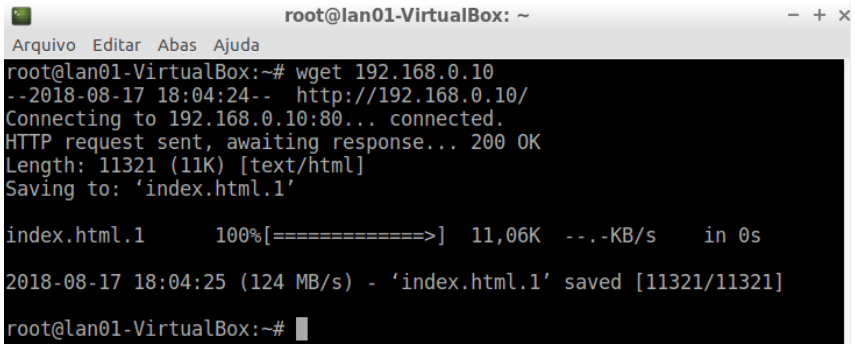
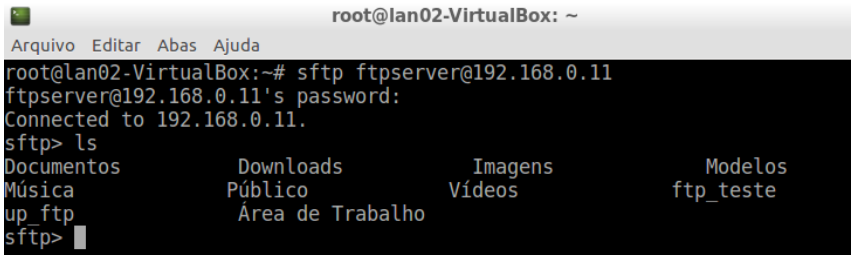


Figura 4.24 – *host* lan02 acessando o serviço SFTP na DMZ.



A regra que bloqueia a entrada pacotes UDP na LAN foi testada com a ferramenta iperf, na qual o servidor UDP está em uma máquina na LAN e os clientes são as máquinas em outras redes que tentam realizar a comunicação UDP, porém sem sucesso, pois a regra 2 da LAN, que pode ser visualizada na Figura 3.3, impede essa comunicação. Como o protocolo UDP não é orientado a conexão o cliente iperf enviou todos os 893 pacotes desejados, entretanto, todos foram descartados no *switch*, não chegando nenhum no servidor iperf, como pode ser constatado nas Figuras 4.25 e 4.26.

Figura 4.25 – Máquina da rede externa tentando sem sucesso realizar uma comunicação UDP com lan01.

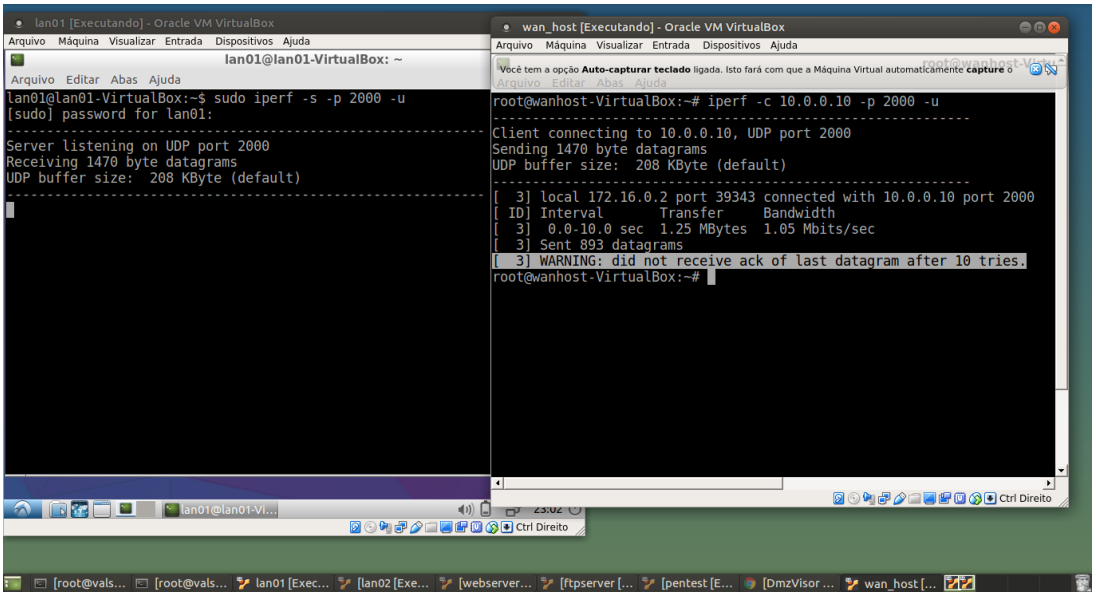
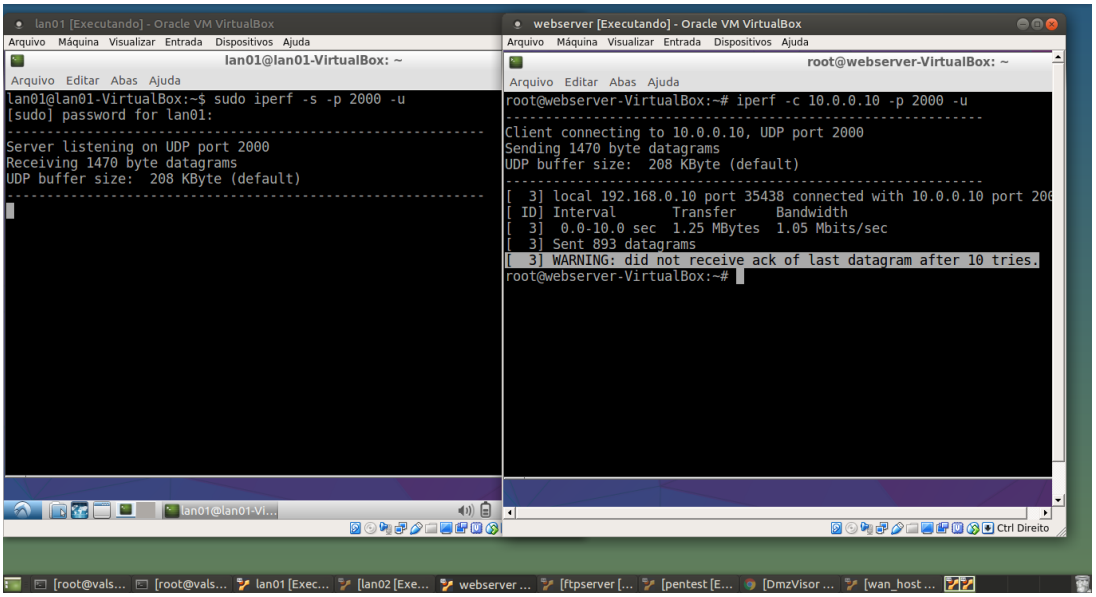


Figura 4.26 – Servidor da DMZ tentando sem sucesso realizar uma comunicação UDP com lan01.



Como ilustrado na Figura 3.3, as regras de ICMP (regras 03 e 04) da LAN, permitem que *hosts* na LAN enviem pings para fora da sua rede, porém pings iniciados fora da LAN destinados a algum *host* na LAN são bloqueados, pois não possuem regras que permita tal ação. Os resultados dos experimentos que comprovam esse comportamento podem ser visualizados nas Figuras 4.27 e 4.28.

Figura 4.27 – *Host* lan01 obtém sucesso ao pingar *host* externo. O caminho inverso não obtém êxito.

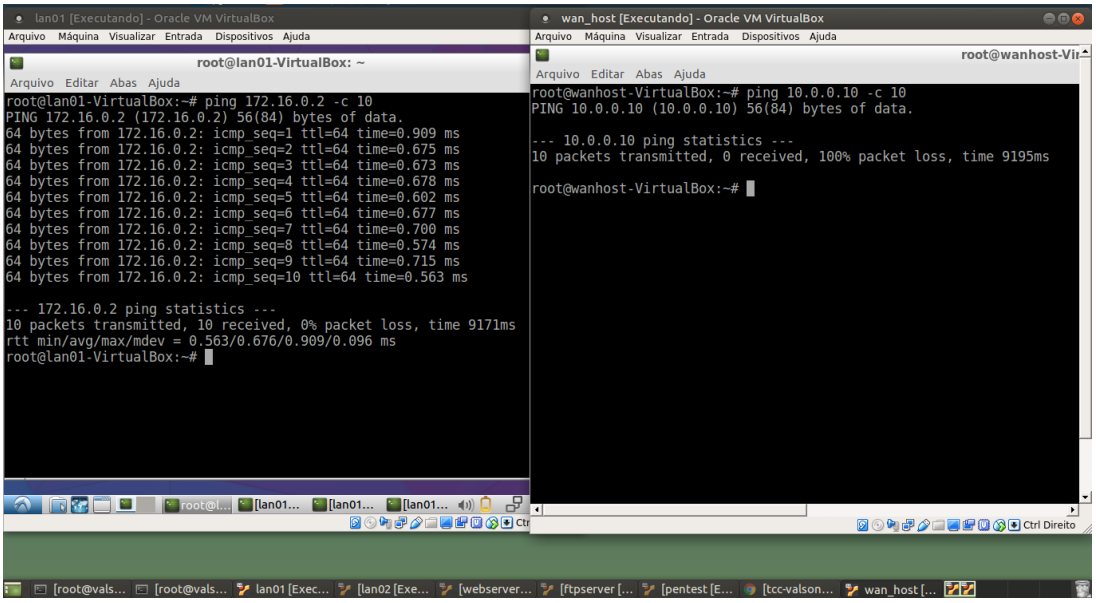
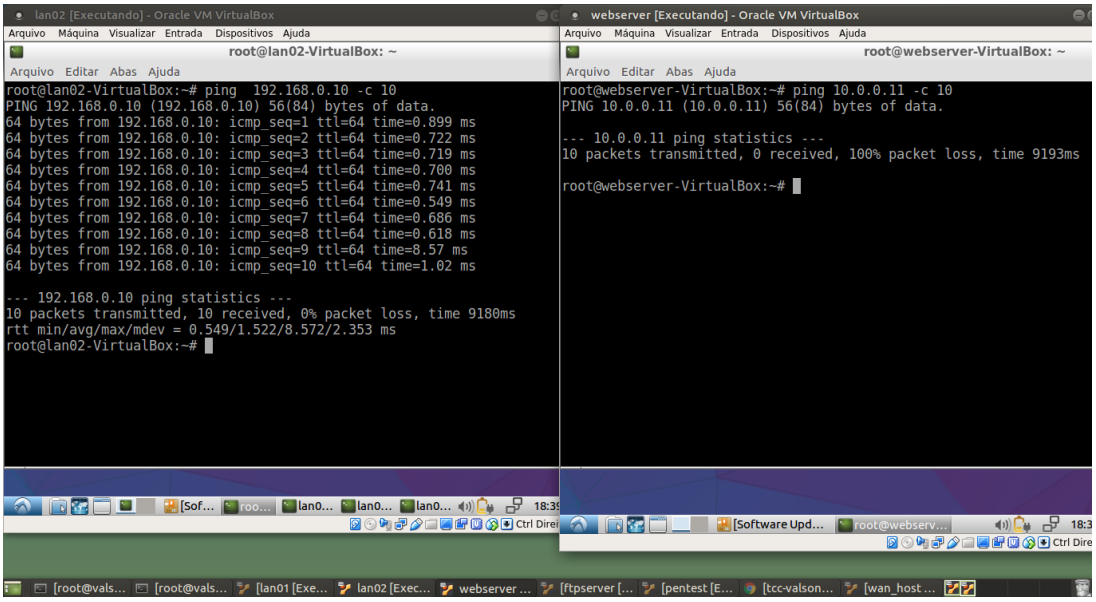


Figura 4.28 – *Host* lan02 obtém sucesso ao pingar *host* na DMZ. O caminho inverso não obtém êxito.



Como as regras da rede DMZ são definidas individualmente para cada *host* desta rede, foram aplicadas regras que permitissem acesso apenas a porta do serviço oferecido por cada servidor, isto é, somente foi liberada a conexão de entrada na porta 80 do servidor web e apenas a porta 22 do servidor SFTP. Em ambos os *hosts* foram aplicadas regras TCP que permitissem a criação de conexões na direção *outbound*.

Diante disso, o hping3 foi executado a partir das outras redes para escanear todas as portas de cada *host* da DMZ. Os resultados obtidos podem ser visualizados nas Figuras 4.29, 4.30, 4.31 e 4.32. Em todos os casos, o *host* externo a DMZ conseguiu visualizar apenas a porta permitida explicitamente pela regra do *firewall*. As demais portas que estavam abertas ficaram inacessíveis fora da DMZ.

Figura 4.29 – Portas abertas no servidor web da DMZ e o escaneamento realizado por wanhost.

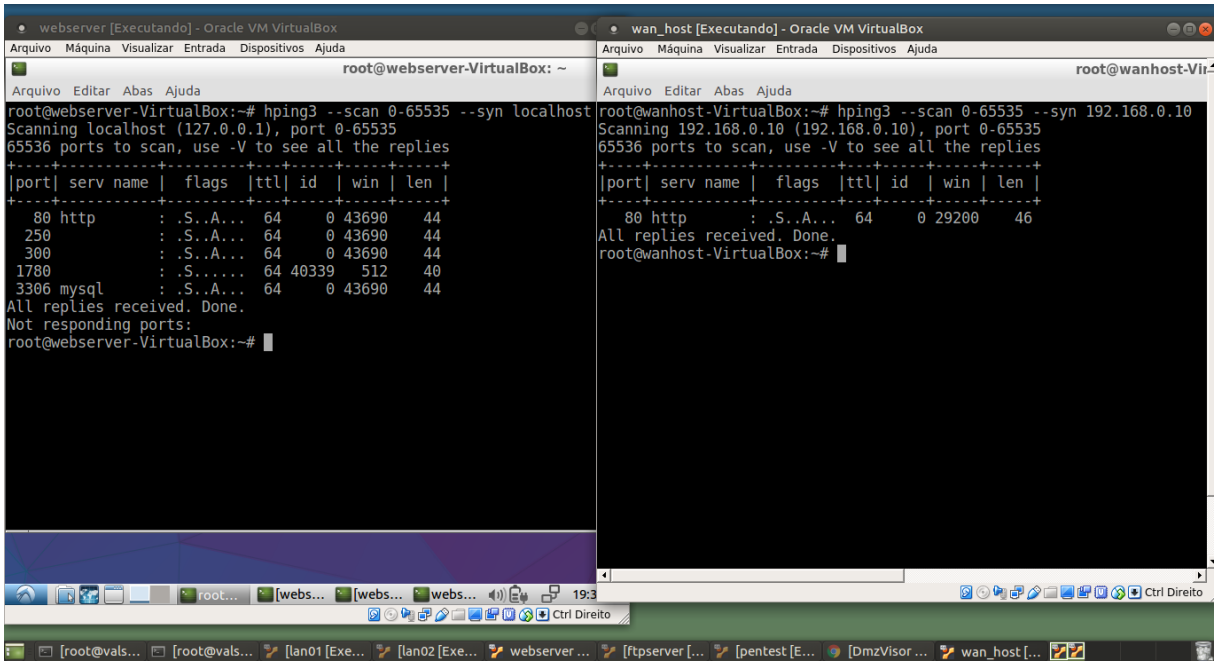


Figura 4.30 – Portas abertas no servidor web da DMZ e o escaneamento realizado por lan01.

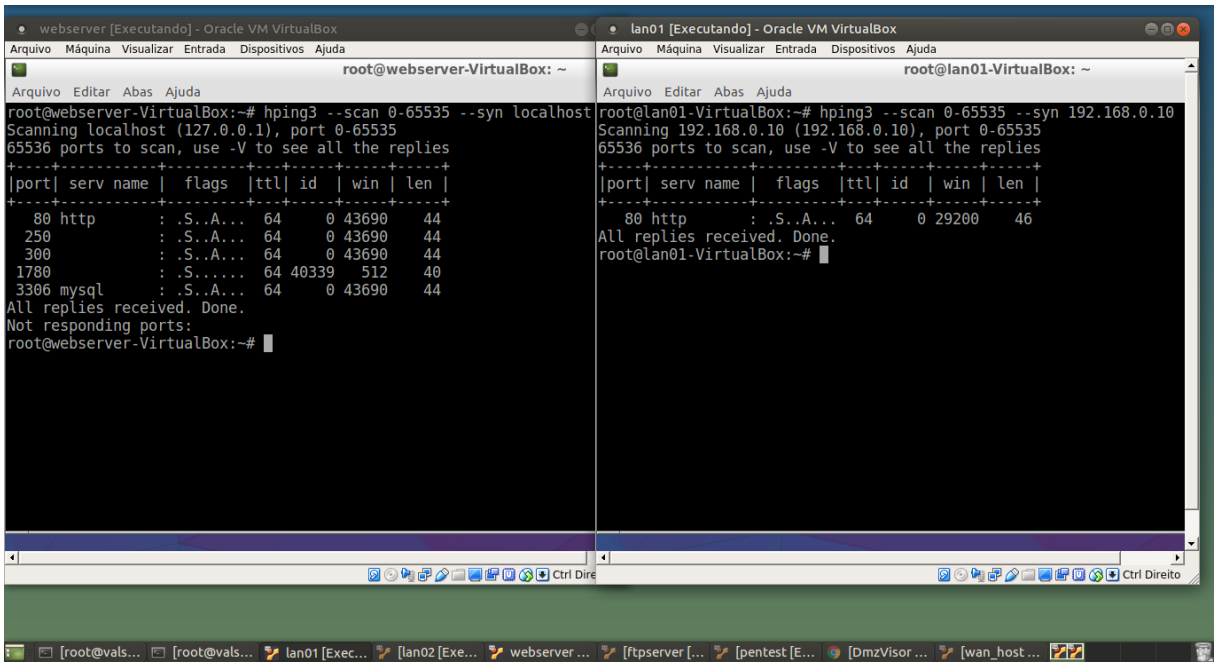


Figura 4.31 – Portas abertas no servidor SFTP da DMZ e o escaneamento realizado por wanhost.

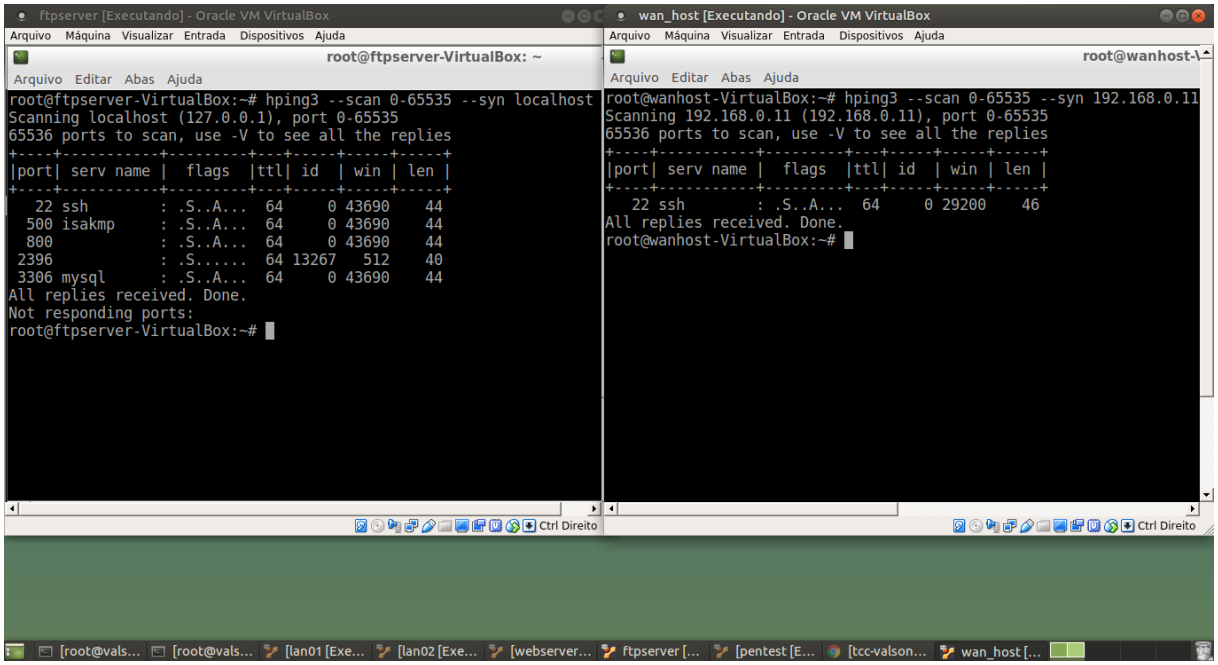
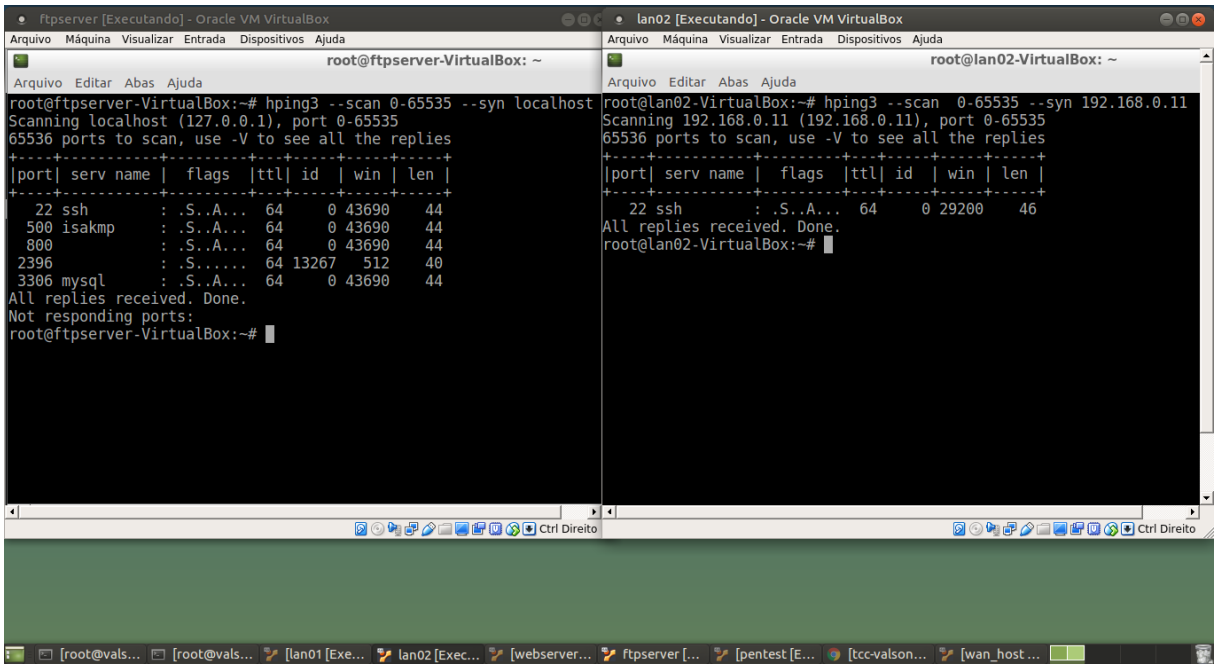
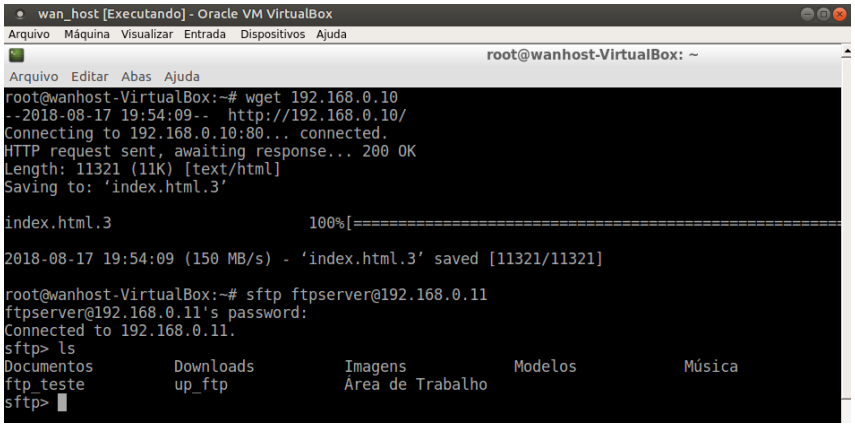


Figura 4.32 – Portas abertas no servidor SFTP da DMZ e o escaneamento realizado por lan02.



Conclui-se que as regras de *firewall* da DMZ são tão eficazes quanto às regras da LAN, fazendo a filtragem correta dos pacotes conforme seu propósito de segurança, pois apesar de existirem várias portas abertas, o hping3 executado a partir das redes LAN e WAN só identifica a porta na qual a regra de *firewall* libera o acesso, sendo assim, filtrando o acesso às demais portas dos servidores. Por fim, é importante destacar que é permitido tanto o acesso da LAN, quanto da internet, à serviços da DMZ, de acordo o que demonstra a Figura 4.33.

Figura 4.33 – O *host* wanhost acessando serviços dos dois servidores da DMZ.



4.3 Resumo do Capítulo

Ao longo deste capítulo foi descrito a metodologia de avaliação do protótipo proposto por essa monografia. Primeiramente foi descrito o ambiente de avaliação que foi realizando por meio da virtualização de máquinas utilizando o VirtualBox. A avaliação foi composta por 02 cenários, no primeiro cenário buscou-se avaliar a segurança da rede LAN tanto internamente como externamente em face das redes DMZ e Internet. Em contrapartida, no cenário 02 foi avaliada a segurança da rede DMZ de forma semelhante ao do primeiro cenário.

A avaliação levou em consideração importantes aspectos considerados durante este trabalho, tais como: conectividade dentro da mesma rede, segurança ARP e DHCP, isolamento das redes e a validação das regras de *firewall* imposta em cada uma das redes.

Os resultados obtidos foram satisfatórios, uma vez que a aplicação mostrou ser capaz de manter as redes da organização seguras em relação às ameaças externas que possam surgir. O *DmzVisor* demonstrou que tanto suas regras de segurança ARP e DHCP, quanto às regras de *firewall* são eficazes nas duas redes que são protegidas pelo protótipo proposto nesse trabalho, conseguindo evitar ataques do homem-do-meio, falsificação de endereços IP e MAC, bem como realizar a filtragem e encaminhamento de pacotes da camada de rede e transporte de forma segura.

5 Conclusão e Trabalhos Futuros

Este capítulo apresenta a conclusão e os trabalhos futuros desta monografia. Na seção 5.1 é mostrada a conclusão do trabalho e a seção 5.2 expõe os trabalhos futuros.

5.1 Conclusão

Esta monografia discutiu a importância para a segurança dos ativos da organização por meio da utilização de uma zona desmilitarizada, que separa a rede interna da organização dos serviços acessados pela a rede externa, no caso a internet.

Juntamente com a discussão sobre a importância no uso de redes DMZ, foi apresentado o *DmzVisor* que tem como proposta principal oferecer um *firewall* de baixo custo e amigável, totalmente implementado fazendo uso de tecnologias gratuitas *open source* e utilizando o emergente paradigma de redes definidas por software, por meio do protocolo *OpenFlow*.

O protótipo implementado neste trabalho, demonstrou pela a avaliação realizada e pelos resultados obtidos que fornece segurança para as redes na qual faz uso do mesmo, pois mitiga o risco de ataques a protocolos importantes, como DHCP e ARP, além de manter o isolamento entre redes e empregar regras de *firewall* auxiliando a organização a manter suas informações seguras em relação à rede externa.

5.2 Trabalhos Futuros

A partir desta monografia almeja-se realizar novos trabalhos complementares e importantes futuramente. Pretende-se concretizar os seguintes trabalhos futuros:

- **Avaliação de desempenho em ambiente real**: avaliar a presente proposta a partir de métricas de desempenho de redes, como por exemplo QoS, em ambiente real que forneça um escopo maior;
- **oferecer suporte à IPv6**: tendo em vista o esgotamento de endereços IPv4 e com relação à

internet do futuro, por meio da internet das coisas, o *DmzVisor* também pode ser adaptado para oferecer suporte ao protocolo IPv6 que já é uma realidade no mundo inteiro;

- **reforçar o isolamento das redes**: para complementar a segurança de isolamento pode ser utilizado mecanismos que envolvam virtualização das redes, que consiste numa área que tem evoluído nos últimos anos em conjunto com a utilização de redes SDN;
- **gerenciamento de várias redes**: estender a proposta para que seja possível adicionar e gerenciar várias redes LAN e DMZ;
- **segurança inteligente**: pode-se aplicar algoritmos de aprendizagem de máquina no Dmz-Visor para que possa aprender automaticamente sobre comportamentos suspeitos em suas redes, com objetivo de emitir alertas para o gestor;
- **monitoramento em tempo real**: uma vez que redes definidas por software possuem uma visão global das suas redes, poderá ser implementado no *DmzVisor*, um monitoramento em tempo real do tráfego. Proporcionando uma interface rica em que o administrador poderá visualizar como a rede se comporta sem a necessidade de ferramentas de inspeção de pacotes, como o Wireshark.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT, N. Nbr iso/iec 27002 – tecnologia da informação – técnicas de segurança – código de prática para a gestão da segurança da informação. *Rio de Janeiro: Associação Brasileira de Normas Técnicas*, 2013.

ADMINLTE. Adminlte control panel template. 2017. Disponível em: <https://adminlte.io>.

ALEXANDER, S.; GRAPHICS, S.; DROMS, R. Rfc 2132: Dhcp options and bootp vendor extensions. IETF, 1997. Disponível em: <https://tools.ietf.org/html/rfc2132>.

ALVES, A. R. et al. Homenetrescue: An sdn service for troubleshooting home networks. In: IEEE. *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, 2018.

AZODOLMOLKY, S. *Software defined networking with OpenFlow*. Birmingham: Packt Publishing Ltd, 2013. ISBN 978-1-84969-872-6.

BADOTRA, S.; SINGH, J. Creating firewall in transport layer and application layer using software defined networking. In: SAINI, H. S. et al. (Ed.). *Innovations in Computer Science and Engineering*. Singapore: Springer Singapore, 2018. p. 95–103. ISBN 978-981-10-8201-6.

BASTA, A.; BASTA, N.; BROWN, M. *Segurança de Computadores e Teste de Invasão*. 2. ed. São Paulo: Cengage Learning, 2014. ISBN 978-85-221-1799-4.

BERDE, P. et al. Onos: Towards an open, distributed sdn os. In: *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. New York, NY, USA: ACM, 2014. (HotSDN '14), p. 1–6. ISBN 978-1-4503-2989-7.

BRAUN, W.; MENTH, M. Software-defined networking using openflow: Protocols, applications and architectural design choices. *Future Internet*, v. 6, n. 2, p. 302–336, 2014. ISSN 1999-5903.

CASADO, M. et al. Ethane: Taking control of the enterprise. In: *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. New York, NY, USA: ACM, 2007. (SIGCOMM '07), p. 1–12. ISBN 978-1-59593-713-1.

CASADO, M. et al. Fabric: a retrospective on evolving sdn. In: ACM. *Proceedings of the first workshop on Hot topics in software defined networks*. Helsinki, 2012b. p. 85–90. ISBN 978-1-4503-1477-0.

CHAPMAN, D. B.; ZWICKY, E. D.; RUSSELL, D. *Building internet firewalls*. 2. ed. Sebastopol: O'Reilly & Associates, Inc., 2001.

COMER, D. E. *Redes de Computadores e Internet.* Tradução de José Valdeni de Lima e Valter Roesler. 6. ed. Porto Alegre: Bookman, 2016.

COMER, D. E. *Interligação de Redes com TCP/IP–Vol. 1: Princípios, Protocolos e Arquitetura*. Rio de Janeiro: Elsevier Brasil, 2017. v. 6.

CONTERATO, M. da S. *Estratégias para redução do consumo de energia em redes de Data Center*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, jan. 2016. Disponível em: <http://tede2.pucrs.br/tede2/handle/tede/6831>.

DART, E. et al. The science dmz: A network design pattern for data-intensive science. In: IEEE. *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. Denver, 2013. p. 1–10.

DROMS, R. Rfc 2131: Dynamic host configuration protocol. IETF, 1997. Disponível em: <https://tools.ietf.org/html/rfc2131>.

DROMS, R.; LEMON, T. *The DHCP Handbook*. 2. ed. Indianapolis, Indiana, USA: Sams Publishing, 2003. ISBN 0-672-32327-3.

DUANGPHASUK, S.; KUNGPISDAN, S.; HANKLA, S. Design and implementation of improved security protocols for dhcp using digital certificates. In: IEEE. *Networks (ICON), 2011 17th IEEE International Conference on*. Singapore, 2011. p. 287–292.

DUMKA, A. *Innovations in Software-Defined Networking and Network Functions Virtualization*. Hershey, PA, USA: IGI Global, 2018.

ERICKSON, D. The beacon openflow controller. In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. New York, NY, USA: ACM, 2013. (HotSDN '13), p. 13–18. ISBN 978-1-4503-2178-5.

ETTERCAP. Ettercap. 2018. Disponível em: <https://www.ettercap-project.org/>.

FARIAS, F. N. et al. Pesquisa experimental para a internet do futuro: Uma proposta utilizando virtualização e o framework openflow. *XXIX Simpósio de Redes de Computadores e Sistemas Distribuídos – SBRC*, Campo Grande, p. 18, 2011.

FERNANDES, E. L. *Implementação de um comutador OpenFlow para experimentação em Redes Definidas por Software*. Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, abril 2015.

FLASK. The python micro framework for building web applications. 2018. Disponível em: <https://github.com/pallets/flask>.

FOROUZAN, B. A. *Protocolo TCP/IP*. 3. ed. Porto Alegre: AMGH Editora Ltda., 2010. ISBN ISBN 978-85-63308-68-9.

GOODRICH, M. T.; TAMASSIA, R. *Introdução à segurança de computadores*. Porto Alegre: Bookman, 2013.

GORANSSON, P.; BLACK, C.; CULVER, T. *Software defined networks: a comprehensive approach*. 2. ed. Cambridge: Elsevier, 2017.

GUDE, N. et al. Nox: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 3, p. 105–110, jul. 2008. ISSN 0146-4833.

GUEDES, D. et al. Redes definidas por software: uma abordagem sistêmica para o desenvolvimento de pesquisas em redes de computadores. *Minicursos do Simpósio Brasileiro de Redes de Computadores – SBRC 2012*, v. 30, n. 4, p. 160 – 210, 2012.

GURGEL, P. et al. *Redes de computadores: da teoria à prática com netkit*. 1. ed. Rio de Janeiro: Elsevier, 2015.

HELLER, B. et al. Elastictree: Saving energy in data center networks. In: *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2010. (NSDI'10), p. 17–17.

HPING. Hping network tool. 2014. Disponível em: <https://github.com/antirez/hping>.

HU, F. *Network Innovation through OpenFlow and SDN: Principles and Design*. Boca Raton, FL, USA: CRC Press, 2014. ISBN 978-1-4665-7210-2.

JAIN, S. et al. B4: Experience with a globally-deployed software defined wan. In: *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. New York, NY, USA: ACM, 2013. (SIGCOMM '13), p. 3–14. ISBN 978-1-4503-2056-6.

JIANG, N. et al. Research of paired industrial firewalls in defense-in-depth architecture of integrated manufacturing or production system. In: IEEE. *Information and Automation (ICIA), 2017 IEEE International Conference on*. [S.l.], 2017. p. 523–526.

JUNIOR, E. C. de A. *Wi-Flow: Uma Arquitetura baseada em SDN para o Gerenciamento e Mobilidade em redes Wi-Fi com Suporte à Autenticação 802.1x*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco, Recife, ago. 2016.

KAUR, K. et al. Programmable firewall using software defined networking. In: IEEE. *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*. New Delhi: IEEE, 2015. p. 2125–2129.

KHONDOKER, R. et al. Feature-based comparison and selection of software defined networking (sdn) controllers. In: IEEE. *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*. Hammamet: IEEE, 2014. p. 1–7. ISBN 978-1-4799-3350-1.

KOMAR, B.; BEEKELAAR, R.; WETTERN, J. *Firewalls for Dummies*. 2. ed. New York: John Wiley & Sons, Inc., 2003.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, IEEE, v. 103, n. 1, p. 14–76, 2015.

KUMAR, A.; SRINATH, N. Implementing a firewall functionality for mesh networks using sdn controller. In: IEEE. *Computation System and Information Technology for Sustainable Solutions (CSITSS), International Conference on*. Bangalore: IEEE, 2016. p. 168–173.

KUROSE, J. F.; ROSS, K. W. *Redes de computadores e a Internet: uma abordagem top-down*. Tradução de Daniel Vieira. Revisão técnica Wagner Luiz Zucchi. 6. ed. São Paulo: Pearson Education do Brasil, 2013.

LARA, A.; KOLASANI, A.; RAMAMURTHY, B. Network innovation using openflow: A survey. *IEEE communications surveys & tutorials*, IEEE, v. 16, n. 1, p. 493–512, 2014. ISSN 1553-877X.

MARIADB. Mariadb: drop-in replacement for mysql. 2018. Disponível em: <https://github.com/MariaDB/server>.

MCKEOWN, N. et al. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833.

MITEFF, S.; HAZELHURST, S. Nfshunt: a linux firewall with openflow-enabled hardware bypass. In: IEEE. *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*. San Francisco: IEEE, 2015. p. 100–106.

NADEAU, T. D.; GRAY, K. *SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies*. Sebastopol: O'Reilly Media, Inc, 2013.

NAKAMURA, E. T.; GEUS, P. L. de. *Segurança de redes em ambientes cooperativos*. São Paulo: Novatec Editora, 2007.

NEGUS, C.; BRESNAHAN, C. *Linux – A Bíblia*. Tradução de Edson Furmankiewicz. Revisão técnica Allan Trabuco. 8. ed. Rio de Janeiro: Alta Books, 2014. ISBN 978-85-7608-774-8.

NLANR/DAST. Iperf. 2018. Disponível em: <https://iperf.fr/>.

NTT. Ryu component-based software defined networking framework. 2018. Disponível em: <https://github.com/osrg/ryu>.

NUNES, B. A. A. et al. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 3, p. 1617–1634, 2014. ISSN 1553-877X.

OFSOFTSWITCH13; CPQD. Openflow 1.3 for openwrt. dec 2017. Disponível em: <https://github.com/CPqD/ofsoftswitch13/wiki/OpenFlow-1.3-for-OpenWRT>. Acesso em: 17 jul. 2018.

ONF. Openflow switch specification – version 1.0.0 (wire protocol 0x01). *Open Networking Foundation*, 2009. Disponível em: <https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf>.

ONF. Software-defined networking: The new norm for networks. *Open Networking Foundation White Paper*, v. 2, p. 2–6, 2012a. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf>.

ONF. Openflow switch specification – version 1.3.0 (wire protocol 0x04). *Open Networking Foundation*, 2012b. Disponível em: <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf>.

OPENVSWITCH. Open vswitch release 2.9.2. jun 2018. Disponível em: <https://media.readthedocs.org/pdf/openvswitch/stable/openvswitch.pdf>. Acesso em: 19 jul. 2018.

ORACLE. Virtualbox. 2018. Disponível em: <https://www.virtualbox.org/>.

ORTEGA, A. L. Gnu mac changer. 2018. Disponível em: <https://github.com/alobbs/macchanger>.

PLUMMER, D. C. Rfc 826: An ethernet address resolution protocol – or – converting network protocol addresses. IETF, 1982. Disponível em: <https://tools.ietf.org/html/rfc826>.

ROTHENBERG, C. E. et al. Openflow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. *Cad. CPqD Tecnologia, Campinas*, v. 7, n. 1, p. 65–76, 2010.

SCHLINKER, B. et al. Engineering egress with edge fabric: Steering oceans of content to the world. In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. New York, NY, USA: ACM, 2017. (SIGCOMM '17), p. 418–431. ISBN 978-1-4503-4653-5.

SENA, Y. A. B. L. de. *Abordagem para Gerenciamento de Mobilidade com Suporte a Multi-homing e Handover Transparente em Redes Definidas por Software*. Dissertação (Mestrado) — Centro de Informática, Universidade Federal de Pernambuco, Recife, ago. 2014.

SHALIMOV, A. et al. Advanced study of sdn/openflow controllers. In: *Proceedings of the 9th Central and Eastern European Software Engineering Conference in Russia*. New York, NY, USA: ACM, 2013. (CEE-SECR '13), p. 1:1–1:6. ISBN 978-1-4503-2641-4.

SONG, H. Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In: ACM. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. Hong Kong: ACM, 2013. p. 127–132.

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

STALLINGS, W. *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud*. Indiana: Addison-Wesley Professional, 2015.

TANENBAUM, A. S.; WETHERALL, D. *Redes de computadores*. Tradução de Daniel Vieira. Revisão técnica Isaías Lima. 5. ed. São Paulo: Pearson Prentice Hall, 2011.

TORRES, G. *Redes de Computadores: Versão Revisada e Atualizada*. 2. ed. Rio de Janeiro: Novaterra, 2014.

UNDERDAHL, B.; KINGHORN, G. *Software Defined Networking for Dummies – Cisco Special Edition*. New Jersey: John Wiley & Sons, Inc., 2015.

WIRESHARK. Wireshark. 2018. Disponível em: <https://www.wireshark.org/>.

YAP, K.-K. et al. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In: ACM. *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. New York, NY, USA, 2017. p. 432–445.

ZERKANE, S. et al. Software defined networking reactive stateful firewall. In: HOEPMAN, J.-H.; KATZENBEISSER, S. (Ed.). *ICT Systems Security and Privacy Protection*. Cham: Springer International Publishing, 2016. p. 119–132. ISBN 978-3-319-33630-5.