UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Um Serviço para Anonimização em Redes Definidas por Software

Dissertação de Mestrado

Leonardo Henrique da Silva Bomfim

São Cristóvão - Sergipe
2017

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Leonardo Henrique da Silva Bomfim

Um Serviço para Anonimização em Redes Definidas por Software

Versão original

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof.ª Dr.ª Edilayne Meneses Salgueiro
Coorientador(a): Prof. Dr. Ricardo José Paiva de Britto Salgueiro

São Cristóvão - Sergipe

2017

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Bomfim, Leonardo Henrique da Silva
B695s Um serviço para anonimização em redes definidas por software / Leonardo Henrique da Silva Bomfim ; orientador Edilayne Meneses Salgueiro. – São Cristóvão, 2017.
101 f. : il.

Dissertação (mestrado em Ciências da computação)– Universidade Federal de Sergipe, 2017.

1. Programas de computador. 2. Redes de computadores. 3. Software - Desenvolvimento. 4. Tecnologia da informação. I. Salgueiro, Edilayne Meneses. II. Título.

CDU 004.4

Leonardo Henrique da Silva Bomfim

Um Serviço para Anonimização em Redes Definidas por Software

Versão original

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Trabalho aprovado. São Cristóvão - Sergipe, 22 de Fevereiro de 2017:

Prof.ª Dr.ª Edilayne Meneses Salgueiro
Orientador - Presidente

Prof. Dr. Ricardo José Paiva de Britto Salgueiro
Coorientador

Professor Dr. Edward David Moreno Odonez
Universidade Federal de Sergipe

Professor Dr. Kelvin Lopes Dias
Universidade Federal de Pernambuco

São Cristóvão - Sergipe
2017

Dedico à Deus, minha esposa, família e amigos.

"Todas as vitórias ocultam uma abdicação"

(Simone de Beauvoir)

RESUMO

Este trabalho tem como objetivo implementar um serviço de anonimização em Redes Definidas por Software (SDN) com o objetivo de realizar a mitigação de tentativas de ataque sofridas por uma rede. Através de um serviço de anonimização é possível realizar a ocultação dos endereços IP dos *hosts* da rede, garantindo maior proteção contra ataques à segurança, permitindo um aumento de sua disponibilidade. Um dos maiores desafios da arquitetura SDN é a segurança. A separação do controle e do plano de dados permite que desafios para garantir a segurança sejam gerados, devido à permissividade da rede a ataques como "Homem no Meio", Negação de Serviço e Saturação. O serviço aqui desenvolvido, denominado de BomIP, utiliza a técnica de anonimização de micro-dados através da randomização dos endereços IP dos *hosts*. O serviço BomIP foi adicionado ao controlador RunOS, que ficou responsável por realizar o gerenciamento dos endereços IP reais e anonimizados. Para validar este serviço foram realizados dois Estudos de Caso em um ambiente simulando um ataque de Negação de Serviço. O primeiro Estudo de Caso realizou a comparação do funcionamento do serviço de anonimização Crypto-Pan com o BomIP. Enquanto que o segundo Estudo de Caso realizou a comparação de uma rede IP tradicional sob ataque de Negação de Serviço e uma SDN utilizando o BomIP. A análise dos resultados mostrou que o serviço desenvolvido tem um tempo de execução 65% mais eficiente que o Crypto-Pan. A análise de complexidade do algoritmo do BomIP demonstrou que é de ordem quadrática. Os resultados também demonstraram que os pacotes anonimizados permitem a rastreabilidade e a mitigação de 80% das tentativas de ataque, dando garantias que os serviços providos pela rede continuem disponíveis.

Palavras-chaves: Redes Definidas por Software. Anonimização. OpenFlow.

ABSTRACT

This work has the goal to make an implementation of an anonymization service on Software-Defined Networks (SDN) with the goal to reduce the number of attacks. With an anonymization service is possible to hide the IP address from the network's hosts, ensuring more protection against security attacks, which allows a more time availability. One of the biggest challenge on SDN architecture is the security issue. The separation of control and data planes allows o generated challenges on security, due to the network's permissiveness to attacks such as " Man in the Middle ", Denial of Service and Saturation. The service developed in this work, named as BomIP, uses the micro-data anonymization technique of randomization of IP address of the hosts. The BomIP was added in the SDN controller RunOS, which was the responsible to make the management of the real and anonymized IP address. To validate this service it was developed two Case Studies with an environment simulating a Denial of Service attack. The first Case Study made a comparison between Crypto-Pan and BomIP. While the second Case Study made a comparison between a traditional network IP and a SDN one using BomIP, both under Denial of Service attack. The analysis of results showed that the service developed has an running time 65% more efficient than Crypto-Pan. The assintotic analysis shows that BomIP is an algorith with running time of quadratic order. The results also showed that the anonymized packets can be tracked and a mitigation of 80% from the attacks trials, ensuring that the services provided by the network remain available.

Keywords: Software-Defined Networking. Anonymization. OpenFlow.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface (Interface de Programação de Aplicativo)
CPU	Central Processing Unit (Unidade Central de Processamento)
DDoS	Distributed Denied of Service (Negação de Serviço Distribuído)
DoS	Denied of Service (Negação de Serviço)
IP	Internet Protocol (Protocolo de Internet)
MAC	Media Access Control (Controle de Acesso ao Meio)
ONF	Open Networking Foundation
SDN	Software-Defined Networking (Rede Definida por Software)

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos os elementos que compõem uma rede têm obtido um melhor desempenho em todos os aspectos, permitindo uma criação de uma sociedade digital, onde a informação está disponível em tempo real em diversos locais. No entanto, a forma tradicional para prover a configuração e administração dos dispositivos estão tornando-se mais custosos e difíceis de realizar (UHLIG, 2015).

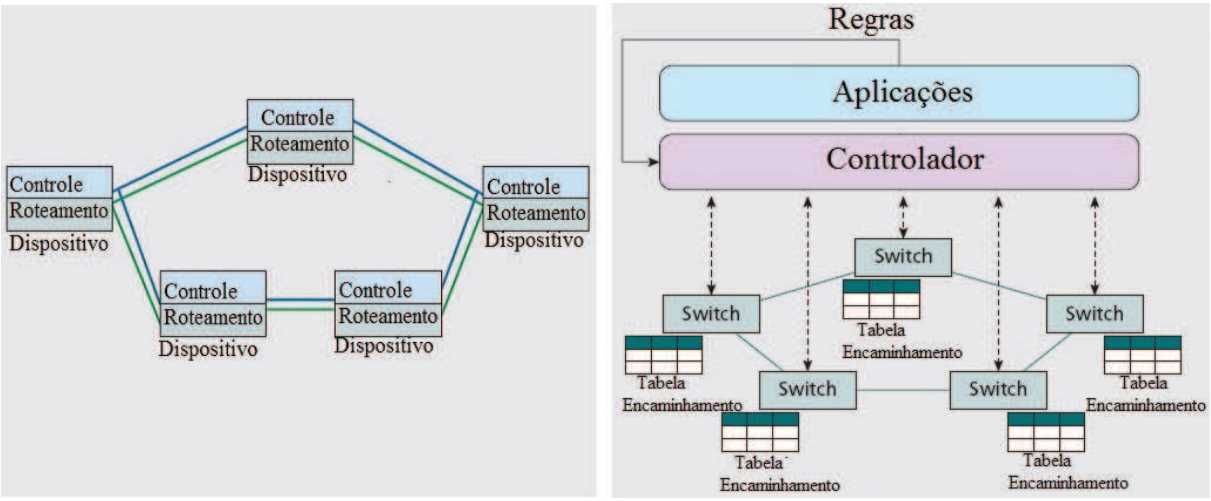
O controle e o transporte de pacotes em uma rede através de roteadores e *switches* são os principais componentes que permitem que a informação, em formato de pacotes digitais, trafeguem pela Internet ao redor do mundo. Apesar deste formato, as redes tradicionais são complexas e difíceis de serem gerenciadas. Para que configurações de alto nível como políticas de segurança e roteamento funcionem, os administradores precisam configurar individualmente cada dispositivo da rede, utilizando, muitas vezes, comandos de baixo nível, e em alguns casos, comandos específicos de cada fabricante. Além da complexidade de configuração, os ambientes de rede têm que suportar o dinamismo de falhas e adaptações de alterações de cargas (UHLIG, 2015). Uma forma de facilitar é através de reconfiguração automática e programação em tempo real dos dispositivos que compõem a rede, porém, estes serviços não estão disponíveis em redes IP tradicionais,

Para resolver este problema surgiram as Redes Definidas por Software (SDN), que consiste na ideia da separação do plano de dados do plano de controle da rede (DABBAGH et al., 2015). O que resulta em transformar a rotina de dispositivos da rede tradicional, em simples *switches* cujo trabalho corresponde apenas a encaminhar a política que é designada por um controlador centralizado e programável. Na Figura 1.1a é apresentada a arquitetura tradicional de uma rede, em que cada dispositivo é responsável por realizar as

funções de controle e encaminhamento de pacotes, enquanto que na Figura 1.1b apresenta a arquitetura SDN, em que os dispositivos da rede correspondem a *switches* que seguem as regras aplicadas pelo controlador central.

Figura 1.1 – Exemplo de Arquitetura SDN e Tradicional

(a) Arquitetura de Rede Tradicional (b) Arquitetura SDN



O plano de controle é responsável por monitorar a rede, tomar as decisões de roteamento e realizar a programação do comportamento físico da rede. O plano de dados é formado por *switches* que são conectados de forma a corresponderem a rede física. A separação entre o plano de dados e o controlador SDN torna-se uma arquitetura fundamental para as redes futuras; no entanto, também cria desafios à segurança. Do ponto de vista de segurança, a preocupação inicial foi do controle centralizado como um ponto de falha na rede (SCOTT-HAYWARD, 2015). Se todas as decisões são realizadas em um local e este é invadido, permitirá o controle da rede pelo usuário invasor.

O papel destes dispositivos é de realizar o encaminhamento de pacotes baseado nas políticas de roteamento indicadas pelo controlador. Para que este objetivo seja atingido, cada *switch* tem uma "Tabela de Encaminhamento", que possui basicamente as regras de encaminhamento indicadas pelo controlador, em que cada uma dessas regras consiste de três campos: padrão, contador e ação.

Uma vez que um pacote é recebido, o *switch* procura uma regra de encaminhamento na tabela cujo campo padrão seja semelhante ao cabeçalho do pacote. Uma vez que a regra é encontrada, o contador é incrementado e a ação da regra é executada. Esta ação tomada pode ser a de encaminhar o pacote ao seu destino, descartar o pacote ou retornar o mesmo para o controlador.

Serviços de segurança em uma SDN são importantes para impedir que a rede possa ser vítima, por exemplo, um ataque de Negação de Serviço, levando a comprometer os serviços que são disponibilizados pela rede (FEGHALI; KILANY; CHAMOUN, 2015). Pacotes do tipo *spoof*, usados em diferentes tipos de mensagens pelo protocolo OpenFlow (*Handshake*, *Hello* etc.) e lógica mal elaborada no controlador podem ser fatores que levem a este tipo de ataque.

Desta forma, serviços voltados para segurança em SDN estão em constante pesquisa e desenvolvimento (AKHUNZADA et al., 2015), para impedir que a identidade dos serviços disponibilizados na rede possam ser descobertas e tornem-se alvos de ataques de usuários maliciosos. Diversos serviços encontram-se em desenvolvimento, dentre as técnicas abordadas no momento, uma de destaque consiste em anonimizar os endereços IP gerenciados pelo controlador (MENDONCA; SEETHARAMAN; OBRACZKA, 2012), de forma a ocultar as estações e servidores, reduzindo riscos de ataques de *spoof* e de Negação de Serviço.

Anonimização consiste em remover ou modificar todos os campos de identificação (nome, identidade, etc.) de um usuário ou entidade, de forma a não permitir a identificação do mesmo, garantindo assim a segurança da informação (KOUKIS et al., 2006). Logo, o objetivo da anonimização é proteger a privacidade dos usuários, não permitindo que informações sejam disponibilizadas de forma pública permitindo a identificação.

Na anonimização de endereços IP (MENDONCA; SEETHARAMAN; OBRACZKA, 2012), algoritmos de anonimizaçao são usados pelo controlador para randomizar os endereços, de forma que a cada novo fluxo de pacote as estações recebam novos identificadores virtuais, impedindo assim, que sejam descobertas e possam ser atacadas.

Atualmente, as pesquisas mais recentes focam em explorar SDN para melhorar a segurança, com uso de sistemas de monitoramento que incluem políticas de segurança para dinamicamente detectar tráfego suspeito durante a operação em tempo real da rede (HU et al., 2015). É esperada que a lista de desafios de segurança em SDN continue a crescer gradativamente, acompanhando o desenvolvimento da tecnologia SDN. Para obter todas as vantagens dos recursos providos pela SDN, estes desafios devem ser destacados para que medidas apropriadas de segurança possam ser tomadas.

1.1 PROBLEMA

A descoberta de estações em uma rede através da identificação de seus endereços e portas, configura-se um problema a ser resolvido no tocante a Redes Definidas por Software. A partir do momento que um usuário malicioso possui a identificação dos serviços fornecidos por uma rede, ele pode realizar ataques para obtenção de informações sigilosas ou mesmo tornar a rede inacessível através de sucessivos ataques, não permitindo que um tráfego legítimo consiga acessar os serviços disponibilizados.

Torna-se necessário o desenvolvimento de técnicas capazes de ocultar as estações da rede, de forma a garantir a disponibilidade dos serviços fornecidos. Técnicas como anonimização permitem preservar a identidade de estações e serviços através do ocultamento dos endereços IP. Porém, estas técnicas de anonimização necessitam de desenvolvimento para SDN, devido às alterações que a rede pode sofrer através de novas configurações que podem ser empregadas no controlador central, dificultando assim, o gerenciamento de identificadores estáticos anonimizados.

1.2 HIPÓTESE

O uso de um serviço de anonimização por fluxo, em uma Rede Definida por Software, pode mitigar o número de ataques sofridos pelos serviços disponibilizados, devido a ocultação dos endereços através do uso de identificadores anonimizados gerenciados dinamicamente pelo controlador.

Com o crescente índice de ataques, ocasionado por fragilidades de segurança que as Redes Definidas por Software vêm sofrendo, através da descoberta de estações, serviços e seus respectivos endereços e portas disponíveis. Assim, torna-se necessário o desenvolvimento de técnicas capazes de ocultar as estações da rede, de forma a garantir a disponibilidade dos serviços fornecidos.

Técnicas como anonimização permitem preservar a identidade de estações e serviços através do ocultamento dos endereços IP em redes tradicionais. Porém, estas técnicas de anonimização necessitam de desenvolvimento para SDN, devido às alterações que a rede pode sofrer através de novas configurações que podem ser empregadas no controlador central, dificultando assim, o gerenciamento de identificadores estáticos anonimizados.

1.3 OBJETIVO

À luz desta hipótese, este trabalho tem o propósito de desenvolver e implementar um serviço de anonimização de endereços IP de *hosts* e serviços em Redes Definidas por Software, que é controlado dinamicamente pelo controlador central, com o objetivo geral de reduzir a quantidade de ataques sofridos pela rede.

Os objetivos específicos deste trabalho são:

- Análise das técnicas existentes de anonimização, para que possa ser possível identificar o melhor algoritmo, em tempo de execução e mitigação de ataques, para implementação do serviço de anonimização de IP em Redes Definidas por Software;
- Realização de um Estudo de Caso de uma Rede Definida por Software com o serviço de anonimização de IP, e uma rede equivalente sem o serviço proposto;
- Análise de Estudos de Caso de Anonimização em Redes Definidas por Software.

1.4 ORGANIZAÇÃO DO TRABALHO

Como solução para o problema levantado, este trabalho apresenta um serviço de anonimização para redes SDN, permitindo o ocultamento das estações e serviços pelo controlador SDN, de forma que os endereços IP da rede não sejam obtidos por um invasor no momento em que ele estiver na rede.

Este trabalho se encontra dividido da seguinte forma: No Capítulo 2 são apresentados os principais conceitos do paradigma SDN, assim como os principais elementos que compõem a sua arquitetura. O Capítulo 3 apresenta os conceitos sobre anonimização, incluindo principais técnicas e algoritmos. Uma revisão sistemática é apresentada no Capítulo 4. No Capítulo 5 é descrito o anonimizador desenvolvido chamado de BomIP, e no Capítulo 6 é apresentado o controlador RunOS e a arquitetura proposta como solução para um serviço de anonimização para SDN. Os principais trabalhos relacionados são apresentados no Capítulo 7. Para validar o serviço de anonimização são apresentados dois Estudos de Caso no Capítulo 8. Por último, o trabalho é concluído no Capítulo 9, com uma reflexão do que foi apresentado.

2 REDES DEFINIDAS POR SOFTWARE

A demanda por serviços de redes tem crescido rapidamente, e desta forma, dispositivos de rede com tráfego de diferentes fontes, como de vídeos, *big data* e dispositivos móveis tornam-se cada vez mais um desafio para os operadores de redes (UHLIG, 2015). Além da maior quantidade de servidores e máquinas virtuais aumentando o tráfego entre servidores.

A grande maioria dos dispositivos de redes tradicionais concentram as funcionalidades de controle e fluxo dos dados (JAMMAL et al., 2014). Porém, os administradores de redes necessitam configurar e gerenciar cada um dos dispositivos separadamente. Para superar este desafio, os administradores necessitam de uma rede eficiente, flexível, ágil e escalável. Com base nestes problemas, surge o conceito de Redes Definidas por Software (SDN - *Software-Defined Network*) que refere-se a uma nova abordagem para as tornar redes programáveis, com capacidade para iniciar, controlar, alterar e gerenciar o comportamento da rede dinamicamente através de uma interface (HALEPLIDIS et al., 2015). A proposta de redes programáveis alavancam a pesquisa de métodos de gerenciamento e configuração de redes automatizadas.

O conceito consiste em "uma arquitetura de rede emergente onde o controle da rede é dissociado e separado do mecanismo de envio, e é diretamente programável" (FOUNDATION, 2015). Desta forma, em SDN há um controlador central lógico, que possui a rede e controla múltiplos encaminhamentos de pacotes entre dispositivos (por exemplo: *switches*) configurados via interfaces. Como exemplos destas interfaces pode-se citar Forces (DORIA et al., 2015) e OpenFlow (MCKEOWN et al., 2008).

2.1 ARQUITETURA SDN

Em uma arquitetura SDN existem três partes distintas separadas por camadas: aplicação, controlador e plano de dados, conforme indicado na Figura 2.1. A camada de aplicação indica a parte que explora a dissociação do controle e do plano de dados para obter metas específicas, como mecanismos de segurança ou soluções de gerenciamento de Internet. Esta é a camada em que aplicações e serviços definem o comportamento da rede.

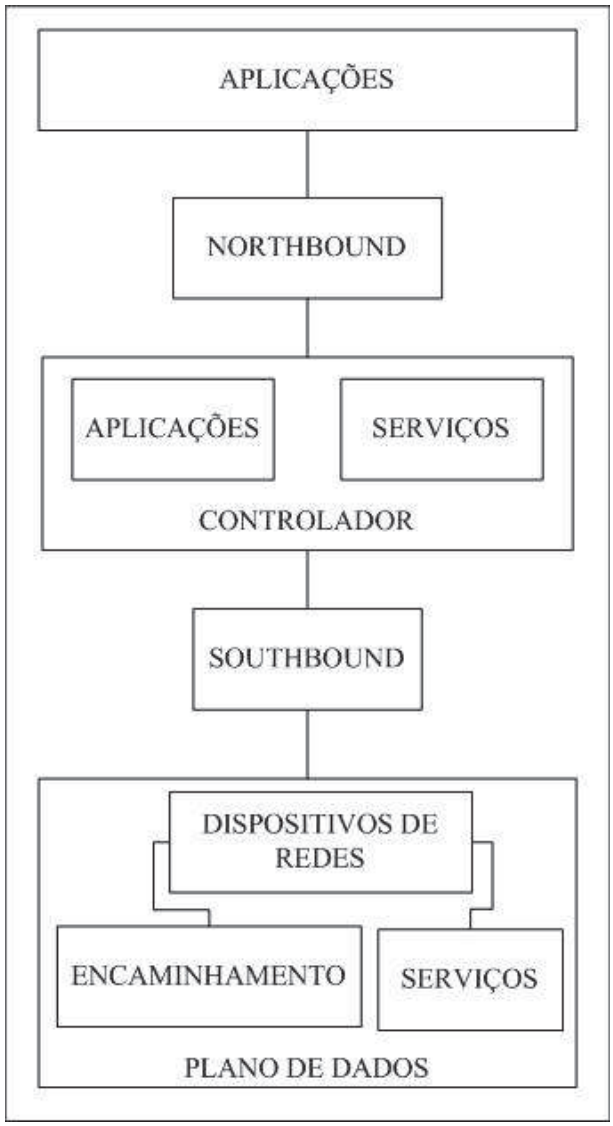
O plano de dados é responsável por lidar com os pacotes baseados nas instruções recebidas do controlador (HALEPLIDIS et al., 2015). Normalmente, o plano de dados é o ponto final dos serviços do controlador e aplicações, não sendo apenas responsável por encaminhar, descartar e alterar pacotes, ele também possui recursos para realizar a classificação dos pacotes.

Em outras palavras, o controlador corresponde a uma abstração completa da infraestrutura de rede, permitindo ao administrador aplicar políticas e protocolos customizados pela rede física. A interface *northbound* representa um ponto de acesso entre as aplicações do controlador e a plataforma SDN. O plano de dados representa o encaminhamento do *hardware* na rede de arquitetura SDN. Devido ao controlador precisar comunicar-se com a infraestrutura de rede, certos protocolos são necessários para realizar este controle e gerenciamento, assim a interface *southbound* corresponde ao ponto de acesso entre o controlador e os dispositivos de rede.

No paradigma tradicional de redes, centrado no *hardware*, *switches* são sistemas fechados que possuem controlador, plano de dados e suporte específico de acordo com o fabricante. Assim, novos protocolos e serviços costumam ser um desafio, e os *switches* precisam ser atualizados. Em contraste com SDN, em que os *switches* podem ser reprogramados para suportar novas tecnologias de comunicação.

O plano de controle, chamado de controlador, atua como uma camada intermediária entre as aplicações e o plano de dados, comunicando-se através da interface *northbound* com as aplicações e com a interface *southbound* com os *switches*. Desta forma, torna-se importante que o controlador seja desenvolvido de forma eficiente. Assim, pesquisas estão sendo realizadas para aprimorar as funcionalidades disponibilizadas por esta camada.

Figura 2.1 – Componentes de um SDN



2.2 IMPLEMENTAÇÕES DE SDN

Por ter um controle centralizado, a SDN enfrenta dificuldades para tornar-se escalável, o que leva a trabalhos serem realizados nesta área. Um destes trabalhos corresponde ao DIFANE (YU et al., 2011) que mantém todo o tráfego no plano de dados e seleciona o direcionamento dos pacotes de acordo com os *switches* que armazenam as regras necessárias, evitando um excesso de fluxo no controlador, obtendo melhor performance e escalabilidade. Um modelo híbrido que combina um controle central e distribuído é proposto em (OTHMAN, 2013) para prover maior controle na escalabilidade, através da definição de características, como fluxo proativo, que é ativado a partir de determinadas circunstâncias.

Devido a constante busca de melhorias em controladores SDN, diversas implementações estão em desenvolvimento, utilizando as Linguagens de Programação disponíveis. Nos trabalhos de (FARHADY; LEE; NAKAO, 2015) e (ALEXANDER et al., 2015) são apresentadas algumas características:

- NOX (C++/Python): Primeiro controlador OpenFlow. Escrito em C++ e Python.
- Maestro (Java): maestro aumenta a vazão da rede através da exploração de multiprocessadores e paralelismo.
- MUL (C): suporta infraestrutura *multi-threaded*.
- IRIS (Java): é um controlador OpenFlow recursivo com objetivo de prover escalabilidade e alta disponibilidade.
- OESS (Perl): é um conjunto de softwares para configurar e controlar dinamicamente VLAN usando OpenFlow.
- RouteFlow (C++): provê virtualização de IP sobre serviços usando OpenFlow.
- RunOS (C/C++): controlador OpenFlow leve e que possibilita adição de serviços.

2.3 FERRAMENTAS PARA AVALIAÇÃO DE CONTROLADORES

Para realizar testes com controladores desenvolvidos, há a necessidade de simular o funcionamento de uma SDN em um ambiente. Com este propósito, diversas ferramentas auxiliam na simulação para controladores, como por exemplo, as ferramentas indicadas em (FARHADY; LEE; NAKAO, 2015):

- Mininet: emula múltiplos *switches* OpenFlow virtuais, *hosts* finais e controladores.
- ns-3: suporta OpenFlow v0.89.
- OMNeT++: suporta OpenFlow v1.2.
- EstiNet 8.0 OpenFlow network simulator: pode simular diversos *switches* OpenFlow v1.3.2 e v1.0.0.
- Trema: *framework* para desenvolver controlador OpenFlow em Ruby e C.
- Mirage: um exokernel para construir aplicações de redes através de uma variedade de plataformas que suportam OpenFlow.
- Cbench: realiza testes no controlador OpenFlow através de uma emulação de *switches* e gera pacotes para tráfegos.

- NICE: encontra possíveis problemas de comunicação e confiabilidade em controladores OpenFlow.
- SDN Troubleshooting System (STS): permite simular dispositivos de uma dada rede e programar casos de testes para encontrar erros.
- Oflops: quantifica a performance de um *switch* OpenFlow.
- OFTest: *framework* baseado em Python que realiza testes em *switch* OpenFlow.
- OFRewind: permite gravar o controle e os dados de tráfego e repeti-los, de forma a gerar erros.
- Linking Infrastructure and Applications (OFELIA): permite pesquisadores realizar testes na rede e controlar e estender redes dinamicamente.
- FIBRE: projeto de pesquisa para uma plataforma que realiza experimentos entre pesquisadores do Brasil e Europa.

2.4 PROTOCOLO OPENFLOW

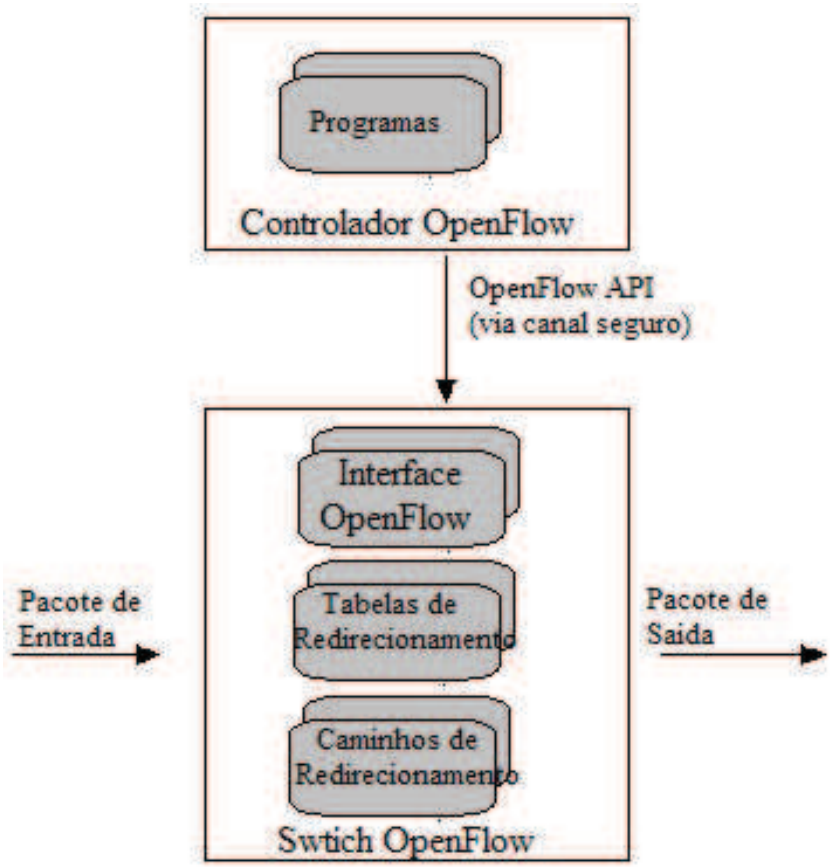
OpenFlow é um protocolo usado para gerenciamento da interface *southbound* da arquitetura SDN. Corresponde a primeira interface padronizada para facilitar a interação entre o controlador e o plano de dados, provendo um acesso por meio de *software* para a tabela de fluxos com instruções para os *switches* e os roteadores sobre como direcionar o tráfego.

Assim, o protocolo OpenFlow define uma API de comunicação entre o controlador e o *switch*, conforme visualizado na Figura 2.2. No entanto, tanto o controlador quanto o *switch* devem implementar o protocolo OpenFlow. O controlador manipula o fluxo de tabelas do *switch* pela adição, atualização e deleção de entrada de fluxos. O *switch* OpenFlow suporta o tráfego do fluxo através da manutenção de uma ou mais tabelas de fluxos.

Um tráfego pode ser definido como um particionamento de fluxos, onde cada fluxo pode ser, por exemplo, uma conexão TCP, de pacotes com os mesmos endereços IP ou MAC ou VLAN, ou pacotes que chegam de uma mesma porta. A arquitetura do OpenFlow consiste, basicamente, de uma quantidade numerosa de *switches* ou outros equipamentos OpenFlow, que são gerenciamentos por controladores OpenFlow.

Cada registro na tabela de fluxo possui três campos (FOUNDATION, 2012): (i) *Header*: que especifica informações como origem e destino, ID, portas, endereços IP e

Figura 2.2 – Componentes de um SDN. **Fonte: (FARHADY; LEE; NAKAO, 2015)**



Ethernet. (ii) Ação: que especifica como o pacote deve ser processado (encaminhado para alguma porta ou perdido ou enviado ao controlador). (iii) Estatística: que inclui informação como número de pacotes, *bytes* e tempo entre pacotes.

Quando um *switch* recebe um pacote que possui uma perda na tabela de encaminhamentos, este pacote pode ser encaminhado ao controlador, passado para a próxima tabela de fluxos ou descartado. Se o pacote for encaminhado para o controlador, o mesmo decide como tratar o pacote, podendo descartar o pacote ou adicionar a um fluxo, o que indica ao *switch* a direção de como encaminhar um pacote semelhante no futuro.

Um *switch* OpenFlow possui uma ou mais tabelas de fluxos que realizam *lookups* e encaminhamentos. O controlador OpenFlow gerencia os *switches* através de um canal com conexão segura, em que este canal corresponde a interface que conecta cada *switch* OpenFlow do plano de dados com o controlador.

Uma função básica do plano de dados consiste em realizar o encaminhamento de pacotes baseado nas políticas de roteamento indicadas pelo controlador. Para que este objetivo seja atingido, cada *switch* tem uma "Tabela de Encaminhamento", que possui

basicamente as regras de encaminhamento indicadas pelo controlador, em que cada uma dessas regras consiste de três campos: padrão, contador e ação. Exemplos de *switches* usados no encaminhamento de pacotes são apresentados na Tabela 2.1.

Tabela 2.1 – Exemplo de Switches.

Fonte: (FARHADY; LEE; NAKAO, 2015)

Tipo Switch	Nome	Linguagem
Open-Source	Open vSwitch	C/Python
	ofsoftswitch13	C/C++
	Indigo	C
	OpenFaucet	Python
	Pantou	C
Comercial	Vendor	Model
	Arista Brocade	7050 series
	Dell Force10	Dell Force10 Z9000 e SSeries S4810
	Extreme	Z9000 e SSeries S4810
	HP	3500, 3800, 5400, 6200, 6600, e 8200 series
	IBM	RackSwitch G8264
	Juniper	MX-series
	NEC	NoviFlow 1248, 1132
	Pica8	P-3290, P-3295, P-3780, e,P-3922
	Larch networks	OpenFlow switch

Controladores e *switches* OpenFlow precisam manter as mesmas políticas para garantir encaminhamento. Com esta finalidade, (PAN et al., 2013) desenvolveu o Flow-Adapter, que permite a consistência necessária para garantir a política de encaminhamento em diferentes tipos de *hardwares* pela conversão do fluxo recebido na tabela de fluxo do *switch* do controlador. Enquanto que (KANIZO; HAY; KESLASSY, 2013) desenvolveram o *framework* Palette que decompõe a tabela SDN em tabelas menores e distribui pela rede.

Pesquisas são realizadas para aprimorar o encaminhamento de pacotes. Em (LUO et al., 2009) é utilizado um acelerador de processamento de redes. Com este trabalho, os autores obtiveram uma redução de 20% no tempo de atraso do encaminhamento dos pacotes. Em (TANYINGYONG; HIDELL; SJöDIN, 2010) os autores descrevem uma abordagem para melhorar a performance das consultas *lookup* no *switch* OpenFlow usando Linux, obtendo um aumento de 25% da vazão na rede comparado com o *switch* OpenFlow padrão.

Outra opção para melhorar o encaminhamento consiste no paralelismo. A aplicação PacketShader (HAN et al., 2010) explora o paralelismo de uma GPU lidando com computação e uso de memória para realizar tarefas e obter objetivos no tráfego de 40Gbps.

2.5 VANTAGENS DE SDN

Conforme mencionado, SDN apresenta vantagens em relação à arquitetura padrão de rede centrada no *hardware*. A separação do controlador e do plano de dados apresenta diversas vantagens, conforme indicado em (UHLIG, 2015):

- Tornou-se fácil programar aplicações devido as abstrações providas pela plataforma do controlador, e as linguagens de programação em redes podem ser compartilhadas.
- Todas aplicações podem ser beneficiadas por terem a mesma informação da rede (visão global da rede), levando a uma maior consistência e eficácia de decisões de políticas.
- Aplicações podem tomar ações (por exemplo, reconfigurar encaminhamento de dispositivos) em qualquer parte da rede. Assim, não há necessidade de elaborar uma estratégia sobre a localização da nova funcionalidade.
- A integração de diferentes aplicações torna-se mais fácil interfaces programáveis. Por exemplo, balanceamento de carga e roteamento podem ser combinados sequencialmente, com decisões sobre balanceamento sendo tomadas a partir de rotinas de policiamento.

2.6 CONCEITOS DE SEGURANÇA EM REDES DE COMPUTADORES

A segurança em Redes de Computadores é um assunto abrangente e inclui diversos problemas, dentre eles preocupar-se em garantir que pessoas mal-intencionadas não leiam ou modifiquem mensagens enviados a outros destinatários. Outra preocupação é evitar que pessoas não autorizadas tenham acesso à serviços na rede, e também em evitar tentativas de inundar a rede com pacotes falsos para não deixar serviços disponíveis para os usuários (TANENBAUM, 2011). Estes problemas são gerados na tentativa de obter informações sigilosos trocadas em uma rede, ou deixar um serviço inacessível por meio de uma ameaça na rede.

De acordo com a RFC 2828 (SHIREY, 2000), uma ameaça é uma potencial violação de segurança, que existe quando uma circunstância ou ação pode gerar brechas prejudiciais na segurança (por exemplo: ataques). Assim, uma ameaça é um perigo que explora uma vulnerabilidade, podendo ser intencional (indivíduo ou organização criminosa) ou acidental

(a possibilidade de um mal funcionamento de um computador, ou um problema causado por uma catástrofe natural, como terremoto).

Um canal de comunicação é considerado seguro, em Redes de Computadores, quando as propriedades de Confidencialidade, Autenticação, Integridade e Disponibilidade das mensagens trocadas no meio são garantidas, assim, permitindo que os usuários consigam se comunicar de forma que as mensagens não sejam obtidas por terceiros. Em (KUROSE, 2013), o autor explica como essas quatro propriedades compreendem a segurança em Redes de Computadores.

Em relação as propriedades citadas temos que a Confidencialidade garante que somente o remetente e o destinatário pretendido devem poder entender o conteúdo da mensagem transmitida, ou seja, a mensagem enviada deve encontrar-se cifrada (com dados disfarçados), e apenas o destinatário deve ser capaz de decifrar (entender). A Autenticação é a propriedade que define que o remetente e o destinatário necessitam confirmar a identidade da outra parte envolvida na comunicação. A propriedade de Integridade garante que o conteúdo da comunicação não seja alterado, através de técnicas de soma de verificação implementadas em protocolos de transporte e de enlace. A propriedade de Disponibilidade surge em meio a crescente ataques de recusa de serviços (como de Negação de Serviço) que inutilizam a rede, estações, ou serviços, não permitindo que usuários legítimos tenham acesso. Assim, um requisito importante para comunicação é garantir que ela possa ocorrer, impedindo que usuários maliciosos não inutilizem a infraestrutura da rede através de ataques.

Invasores utilizam diversas técnicas de escaneamento para descobrir alvos vulneráveis na rede e realizarem um ataque. Conforme a RFC 2828 (SHIREY, 2000), um ataque em um sistema seguro deriva de uma ameaça inteligente, que pode ser deliberada através de uma tentativa para evadir serviços de segurança e violar as políticas do sistema. Um ataque pode ser classificado como ativo (tenta alterar os recursos do sistema ou afetar o funcionamento) e passivo (apenas obter informações do sistema sem afetar os recursos ou funcionamento).

Dentre as formas de ataques, pode-se exemplificar o de Negação de Serviço (DoS - *Denied of Service*), em que o invasor envia uma grande quantidade de requisições de múltiplos computadores para um único computador alvo. Este alvo torna-se indisponível devido a grande quantidade de requisições desnecessárias que necessita processar (WONGKHUENKAEW; BOONMA, 2015), não permitindo assim, que os serviços possam

ser acessados por outros usuários. Outro ataque é o Homem-no-Meio (MITM - *Main-In-The-Middle*) que ocorre quando um usuário forja o endereço MAC do destinatário de uma mensagem e o coloca como *gateway* do remetente (WU et al., 2015), fazendo com que as mensagens sejam enviadas para o usuário malicioso.

Uma forma de defesa para ataques é o uso de IP randômicos usando SDN (SCOTT-HAYWARD, 2013). Esta técnica usa o controlador OpenFlow para gerenciar um conjunto de endereços IP virtuais, que são designados para hosts da rede, escondendo o endereço IP real do mundo externo. Técnicas como criptografia, autenticação, certificação e controle de acesso através de dispositivos físicos, como *firewall* são utilizadas na tentativa de obtenção de um canal seguro de comunicação de redes, de forma a garantir que apenas os usuários remetente e destinatário visualizem as mensagens trocadas entre si.

Técnicas de segurança estão em constante estágio de estudo e desenvolvimento, para que serviços possam ser adicionados à arquiteturas de redes, provendo uma melhor garantia de segurança. Segundo a RFC 2828 (SHIREY, 2000), um serviço de segurança é um processo ou um serviço de comunicação que provê, por um sistema, uma proteção específica para os recursos disponíveis, como exemplos de serviços podem ser citados auditoria, controle de acesso e ocultação dos serviços da rede com uso de anonimização.

Anonimização consiste em manter o anonimato da informação, ou seja, sem assinatura do autor para identificar a fonte de origem (MELO; GUEDES, 2010). Assim, no escopo da disponibilidade de dados através de meios eletrônicos, a informação é anônima quando não se consegue identificar a quem está se referindo.

2.7 SEGURANÇA EM SDN

Pesquisas indicam que SDN encontra-se distante de resolver questões relacionadas a segurança (AKHUNZADA et al., 2015). Desta forma, segurança em SDN torna-se uma questão prioritária a ser resolvida, necessitando de um ambiente simples, eficiente e escalável.

Um aspecto importante em SDN consiste na segurança, para permitir a disponibilidade de interações entre a rede e aplicações para um eficiente controle de acesso, e garantir a proteção dos recursos da rede contra qualquer tipo de ataque. Em particular, as políticas de segurança para SDN devem garantir que os recursos sejam devidamente protegidos contra ações que coloquem em risco a rede ou as aplicações.

Como inicialmente, segurança não foi considerado como parte do desenvolvimento em SDN, cada camada possui suas implicações e requerimentos que necessitam serem avaliadas como questões de segurança. A rede necessita de um *framework* robusto que garanta a direção correta do controlador. Apesar de que segurança deveria ser construída como parte da arquitetura SDN, ela deve ser entregue como um serviço para prover privacidade e integridade de todos os recursos conectados.

Conforme explicado na Seção 2.6, as propriedades básicas para garantir segurança em comunicações de redes são confidencialidade, integridade, disponibilidade da informação e autenticação. Assim, para prevenir ataques maliciosos ou danos, as alterações da arquitetura de rede implementadas por SDN devem ser avaliadas para garantir que a segurança da rede seja mantida (HELLER; SHERWOOD; MCKEOWN, 2014), permitindo que as mensagens trocadas na rede sejam disponíveis apenas para o remetente e o destinatário.

Os serviços providos devem ter procedimentos para assegurar a confiabilidade dos *softwares* embutidos em nós da rede. Tais procedimentos devem incluir o comportamento dos componentes do *software*, detectar vulnerabilidades etc. Estas medidas de segurança devem ser ativadas durante a inicialização da SDN, e também em atividades que implicam em atualizações de sistemas. Contudo, procedimentos de segurança não são especificados em SDN, o que se torna um desafio prático existente para operacionalização da rede.

A especificação do *switch* OpenFlow 1.3.0 descreve o uso de uma camada de transporte segura (TLS) com autenticação mútua entre o controlador e os *switches* (BENTON; CAMP; SMALL, 2013). No entanto, essa configuração é opcional e não padronizada, e por conta disto, a maioria dos desenvolvedores não adota esta camada e possibilita que ataques de Negação de Serviço sejam realizados.

Uma análise geral de segurança em SDN é apresentada em (PAULO, 2013). Nesse trabalho o autor concluiu que, o controlador sendo centralizado e a rede programável, permitem que novas ameaças sejam introduzidas. Análises tem sido realizadas indicando que, alterações nos elementos e nos relacionamentos no *framework* SDN introduziram novas vulnerabilidades, que não estavam presentes antes da SDN.

Sistemas de monitoramento são essenciais para proteger redes de ataques. Em (BRAGA; MOTA; PASSITO, 2010) foi apresentado um método para detectar ataques de Negação de Serviço Distribuído (DDoS), utilizando controlador OpenFlow baseado em C++ (NOX). Outra abordagem é a utilizada pelo OpenSAFE (BALLARD; RAE; AKELLA, 2010), que faz uso da linguagem especificação de fluxo ALARMS (*A Language for*

Arbitrary Route Management for Security). Segundo os autores essa linguagem simplifica o gerenciamento do monitoramento da rede.

SDN têm duas propriedades que se tornaram atrativas para usuários maliciosos. Primeiramente, o controle da rede encontrar-se realizado por *software*, que sempre está sujeito a falhas e a possuir vulnerabilidades. E segundo, a centralização do controlador, que permite que quaisquer usuários que tenham acesso a este componente, poder controlar a rede completamente.

Em (PAULO, 2013) são descritas potenciais ameaças para redes SDN:

1. Vulnerabilidade dos *switches*: de forma que apenas um *switch* pode ser usado para realizar perdas ou desvio de pacotes numa rede, ou mesmo para sobrecarregar o controlador.
2. Ataques ao controlador: que pode gerar ataque de Negação de Serviço ou para captura de informações.
3. Vulnerabilidades no controlador: que consiste em um dos mais perigosos. Uma falha ou um controlador malicioso pode comprometer toda a rede.
4. Falta de mecanismos para garantir segurança entre o controlador e o gerenciamento de aplicações: o que corresponde a uma ameaça semelhante a segunda, ou seja, as falhas de segurança suscetíveis aos *softwares*.
5. Vulnerabilidade de estações em ambientes administrativos: comum em redes tradicionais, a rede é atacada a partir de estações de trabalho em ambientes corporativos. A diferença é que em redes SDN, devido ao controlador ser centralizado, o acesso através de uma estação pode comprometer toda a rede.
6. Falta de confiança em recursos para análises forenses: o que permite compreender a causa de um problema detectado e realizar procedimentos mais rápidos para que a rede volte a operar normalmente.
7. Fluxos com tráfegos falsos: pode ser usado para atacar *switches* e controladores. Esta ameaça pode ser disparada por uma falha de dispositivos ou por um usuário malicioso. O ataque pode utilizar elementos da rede (servidores, computadores etc) para iniciar um ataque de Negação de Serviço contra o *switch* OpenFlow e recursos do controlador.

Em (AKHUNZADA et al., 2015) são descritas técnicas de segurança em desenvolvimento para SDN. Um exemplo dessas técnicas é a aplicação FRESCO (SHIN et al., 2013),

que consiste em uma especificação de segurança para desenvolvimento de aplicações para redes OpenFlow. FRESCO facilita a exportação de scripts, o que permite a especialistas em segurança desenvolverem ferramentas para detectar e monitorar a rede.

O trabalho de (SKOWYRA et al., 2013) discute uma ferramenta de infraestrutura para especificar e analisar ambientes reais sem conhecimento prévio de linguagens e lógica. Esta proposta considera a verificação da rede e a especificação do modelo enquanto analisa a escalabilidade da rede OpenFlow.

Em um ambiente dinâmico, como SDN, a execução de políticas de segurança é uma questão séria. Em (SON, 2013) os autores propõem FLOVER, um modelo de sistema que verifica as políticas do fluxo contra as políticas de segurança da rede.

FleXam foi proposto por (SHIRALI-SHAHREZA; GANJALI, 2013) para aprimoramento de segurança, como uma extensão para OpenFlow, que permite acesso por parte do controlador para obter informações do pacote, permitindo obter amostras de forma determinística ou estocasticamente. Outros exemplos de aplicações são propostas por (WANG, 2013), que apresenta uma abordagem sistemática para detectar e resolver conflitos em *firewall* SDN através da análise de autorizações e fluxos.

2.8 PROBLEMAS E QUESTÕES ABERTAS

SDN possui questões pendentes relacionadas a segurança, principalmente por permitir que desenvolvimentos sejam realizados sobre a arquitetura, o que pode permitir que diversos ataques sejam realizados na rede, como sugerem alguns autores. O trabalho (WEN et al., 2013) e de (PORRAS et al., 2012), que apresentam, respectivamente, as ferramentas PermOF e ForNox , permitem o mínimo de privilégios a aplicações através de um resumo de permissões na API. O *framework* FRESCO, apresentado por (SHIN; GU, 2013), permite executar e relacionar segurança aplicada aos programas, contra ataques como de textitDistributed Denied of Service (DDOS), que realiza diversas requisições ao controlador central.

Desta forma, o controlador OpenFlow necessita ser robusto em vários aspectos. Em (KUZNIAR et al., 2013) os autores discutem um tempo para que o sistema se recupere de uma falha, fazendo com que uma nova instância do controlador seja criada, substituindo a anterior que apresentou problema.

A localização do controlador na rede pode impactar na performance do SDN. No trabalho (HELLER; SHERWOOD; MCKEOWN, 2012) os autores tentam responder ao questionamento "dada uma topologia, quantos controladores precisam, e onde eles devem ser colocados?". Para responder a este problema, eles fazem um exame de limites para controlar a propagação da latência. Através dos experimentos, é comprovado que a resposta depende da topologia.

Em (JAMMAL et al., 2014) o autor indica os principais desafios a serem enfrentados pelos pesquisados de SDN:

- Confiança: o controlador SDN deve configurar e validar as topologias de redes para prevenir erros e aumentar a disponibilidade da rede. Em redes legadas, quando um dispositivo de rede falha, o tráfego é roteado alternativamente para nós vizinhos, no entanto, na arquitetura de controlador centralizado (SDN), em caso do controlador falhar, toda rede pode entrar em colapso.
- Escalabilidade/Limitações da CPU: com o aumento da banda e do números de *switches*, mais requisições são enviadas ao controlador, que pode não suportar lidar com todos as requisições.
- Performance: deve-se avaliar a vazão de controladores para lidar com pacotes sem fluxos ainda definidos.
- Localização do controlador: consiste em um problema de *design* da rede, que corresponde a encontrar a melhor localização para colocar um controlador, de forma a minimizar a latência e melhorar a confiança da rede.
- Interfaces de baixo nível: apesar do SDN permitir um gerenciamento das políticas através de interfaces simples com o usuário, o *framework* SDN precisa traduzir essas políticas para as configurações do *switch* de baixo nível.
- Interoperabilidade: o controlador deve ser capaz de comunicar-se com as aplicações executadas sobre as camadas superiores do SDN.
- Segurança: caso o *software* controlador sofra ataque de um *hacker*, isto irá permitir ao invasor causar danos em todos os aspectos da rede. Para aumentar a segurança do controlador deve-se adicionar a habilidade de suportar autenticação e autorização de classes do administrador de redes. Outra solução consiste em adicionar uma Lista de Controle de Acesso (ACL) no controlador para filtrar pacotes e completar o isolamento com a infraestrutura, e o controlador deve ser capaz de alertar o

administrador em caso de tentativa de ataque. Porém, soluções mais robustas de segurança estão em desenvolvimento, devido a esta ser uma questão pendente e de grande importância.

3 ANONIMIZAÇÃO

Pesquisadores na área de privacidade de dados focam em desenvolverem técnicas para que os dados publicados sejam úteis, porém preservando a privacidade de indivíduos. O processo de anonimização refere-se a esconder a identidade de indivíduos. A remoção de identificadores como “Nome” ou “Código” não são suficientes, porque pode-se combinar informações como idade, gênero e CEP para identificar o indivíduo.

Em Redes de Computadores, a anonimização é a modificação dos dados de tráfego em uma rede, para proteger e preservar as identidades dos pontos de origem e destino dos registros compartilhados. A anonimização tende a preservar propriedades de tráfego da rede para análise, mas de forma a não permitir que uma aplicação consiga rastrear a rede analisada (FARAH; TRAJKOVI, 2013), realizando o ocultamento dos serviços e estações de uma rede.

Dentre as técnicas atuais, usadas para SDN, existe a anoninimização (MENDONCA; SEETHARAMAN; OBRACZKA, 2012), que possui pesquisa em atividade com diversas implementações sendo realizadas comercialmente e com código aberto (CHAVEZ; STOUT; PEISERT, 2015). A anonimização tem por característica realizar o ocultamento de estações e serviços disponíveis em uma rede, através da modificação dos dados do tráfego que identificam o endereço de destino do pacote. Este ocultamento tem o objetivo de impedir que um usuário malicioso, ao realizar uma análise do tráfego, consiga obter os endereços de serviços e estações existentes na rede. Desta forma, ele não realiza ataques como Negação de Serviço e Homem no Meio, por não ser capaz de obter o endereço real dos serviços.

3.1 TÉCNICAS DE ANONIMIZAÇÃO

As técnicas de anonimização de dados, em relação aos tipos de dados em um tráfego de redes, são classificadas em Macro-Dados e Micro-Dados (DARA, 2014). Na técnica de Macro-Dados são adicionados dados sintéticos para obter privacidade nos registros, de forma que todos os valores sejam modificados. Enquanto que na técnica de Micro-Dados trabalha-se em nível dos campos da rede, apenas os que contém dados de identificação (endereço MAC, IP, portas) do usuário são modificados (UBIK, 2007). Estes dados são criptografados e podem ser utilizados para análise do tráfego da rede.

Para realização da técnica de Macro-Dados, existem desafios que devem ser analisados. Primeiramente, para obter privacidade nas informações, devem ser adicionados dados do tipo "dummy" no lugar dos dados reais. Outro desafio prático é em relação a adição de dados sintéticos com qualidade para imitar os dados reais, o que pode ser um trabalho complexo.

De maneira geral, na técnica de Macro-Dados, os dados não são realmente anonimizados, apenas são alterados por valores aleatórios, o que impede análise dos registros de forma consistente, permitindo apenas que estatísticas como contagem de pacotes, média de tamanho de pacotes etc.

Já na técnica de Micro-Dados, é utilizada criptografia dos dados sensíveis (ou seja, campos capazes de identificar o usuário), de forma a preservar as características do campo anonimizado, como tamanho do campo, quantidade de *bytes* e grupo de valores possíveis. Por exemplo, supondo a anonimização do tráfego de rede referente ao campo de porta de destino, em que o valor 80 é representado por 110; sempre que a porta de destino no tráfego for 80, o mesmo será substituído por 110. Logo, utilizando a técnica de Micro-Dados é possível, por exemplo, realizar processo de anonimização em um arquivo contendo tráfego de uma rede, e o mesmo permanece disponível para estudos e análises, mas com os dados anônimos.

Outros desafios também devem ser analisados para uso de técnicas de Micro-Dados. Por exemplo, garantir que um campo que acomode a informação de um endereço IPv4 de tamanho de 32 bits, receba um dado anonimizado correspondente a um endereço IPv4.

3.2 K-ANONIMIZAÇÃO

K-Anonimização é uma técnica de Micro-Dados proposta por (SWEENEY, 2002). Nesta técnica, uma estrutura com dados específicos sobre um indivíduo produz novos dados com garantias de que o indivíduo não poderá ser identificado, permitindo que dados continuem utilizáveis.

Os métodos para realizar a K-Anonimização consistem em supressões e generalizações das informações (AYALA-RIVERA et al., 2014). No método de supressão os dados selecionados são substituídos por um símbolo, por exemplo, o asterisco "*", para indicar que este dado não é divulgado. Enquanto que no método de generalizações valores individuais são substituídos por valores categorizados.

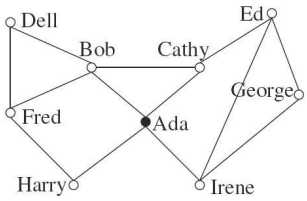
Um exemplo de K-anonimização é apresentado por (ZHOU; PEI, 2011). Supondo que a rede social da Figura 3.1a seja publicada, uma anonimização apenas ocultando os nomes dos participantes da rede é apresentada em 3.1b, mas isto não é suficiente para garantir a privacidade da rede, uma vez que se um usuário souber os vizinhos de um indivíduo, ele pode obter toda a rede.

Por exemplo, se um usuário souber que Ada possui dois amigos que se conhecem, e tem outros dois amigos que não se conhecem, conforme o grafo de Ada da Figura 3.1c. Assim, o vértice que representa Ada no grafo pode ser identificado unicamente na rede porque os demais não possuem as mesmas características. Similarmente, Bob também pode ser identificado em 3.1b.

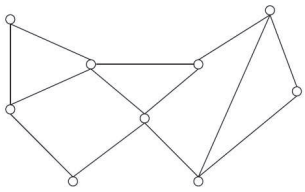
De acordo com (SWEENEY, 2002), para proteger a privacidade, uma alternativa é garantir que nenhum indivíduo possa ser identificado corretamente na rede anonimizada com probabilidade maior que $\frac{1}{k}$, onde k é um usuário específico. Logo, através da adição de uma aresta entre Harry e Irena, conforme a Figura 3.1d, nenhum usuário com conhecimento de 1 vizinho do grafo consegue identificar os integrantes da rede, devido ao grafo encontrar-se com grau de confiança maior ou igual a $\frac{1}{2}$. Ou seja, indicando que para cada vértice selecionado do grafo há pelo menos 2 vizinhos ligados a ele, não existindo mais vizinhos ligados a apenas um outro vizinho.

Figura 3.1 – Exemplo de Anonimização de uma Rede. Fonte: (ZHOU; PEI, 2011)

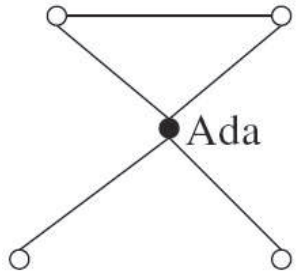
(a) Exemplo de Rede Social



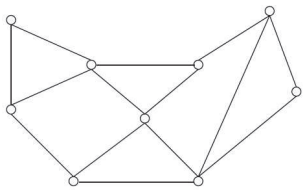
(b) Exemplo de Rede Social com nós anonimizados



(c) Grafo do nó Ada da Rede Social



(d) Grafo K-anonimizado da Rede Social



3.3 ALGORITMOS DE ANONIMIZAÇÃO

Para que seja possível anonimizar informações, conforme apresentado no exemplo anterior, é necessário adotar uma técnica de anonimização. Os autores (FARAH; TRAJKOVI, 2013) apresentam as principais técnicas para se realizar anonimização:

- Substituição por uma Constante: apaga ou substitui a informação completa de um campo por um valor fixo, podendo ser aplicado em qualquer campo do fluxo de uma rede. Apesar desta técnica proteger os dados do campo, também reduz a utilidade das informações.
- Enumeração: o processo inicia ordenando os dados, para após este processo escolher um valor maior do que o primeiro valor ordenado, e então, este valor é adicionado a

todos os dados apontados no conjunto. Este algoritmo não pode ser utilizado em todos os campos.

- Hash: substitui os dados por uma *string* de tamanho fixo usando técnicas de funções de criptografia *hash*, fazendo com que qualquer alteração nos dados resulte em uma mudança de valor. No entanto, o algoritmo *hash* é fácil de ser decodificado, isto ocorre porque em alguns casos, a função *hash* pode resultar em um valor menor do que a informação real. Por exemplo, se a função *hash* de um endereço IPv4 for menor do que 32 bits, o ataque pode tornar-se possível.
- Particionamento: divide um conjunto de possíveis valores em sub-conjuntos por uma relação equivalente. Assim, a função de anonimização substitui cada valor por um correspondente do sub-conjunto.
- Degradação: remove o componente mais preciso do campo. É muito utilizado para anonimização de datas, podendo transformar várias informações de datas em apenas uma.
- Permutação: muito utilizado para anonimização de endereços IP e MAC. Aplica-se uma permutação randômica usando um conjunto de possíveis endereços. Este algoritmo utiliza duas tabelas *hash*: uma para mapear o IP não anonimizado para o anonimizado e outra para armazenar todos os endereços anonimizados.
- Preservação de Prefixo: similar ao algoritmo Permutação, compreendendo um sistema de substituição direta, preservando a estrutura dos valores. Este algoritmo verifica se dois endereços IP possuem os primeiros n bits, então os endereços IP anonimizados também vão possuir os primeiros n bits iguais. O conjunto de endereços anonimizados são gerados a partir de um bloco de endereços criptografados, que consiste em um método que usa uma chave criptográfica para gerar endereços anonimizados.
- Deslocamento aleatório: cria um conjunto de possíveis valores para serem alterados. E para cada campo, escolhe um valor do conjunto e o altera.
- Arredondamento: este algoritmo é usado para anonimizar endereços IP e MAC, em que uma parte dos dados são mantidos inalterados, enquanto que outra porção é apagada. Esta técnica é eficaz para tornar os *end-points* não identificáveis.
- Arredondamento Reverso: este algoritmo é usado para anonimizar endereços IP e MAC, removendo os n mais significantes bits de um endereço e substituindo por zeros. Esta técnica é eficiente para tornar uma rede não identificável.

- Anulação de Unidades de Tempo: algoritmo de particionamento usado para anonimizar datas. Este algoritmo anula alguns valores dos campos de data substituindo por zeros.

Este capítulo apresentou uma abordagem sobre os conceitos de anonimização e as principais técnicas existentes. Dentre estas técnicas, as que são utilizadas em anonimização de dados em SDN correspondem as técnicas de Deslocamento Aleatório e Preservação de Prefixo. Este trabalho realiza uma comparação entre as ferramentas desenvolvidas no Capítulo 7, referente aos Trabalhos Relacionados na área.

4 REVISÃO SISTEMÁTICA

Este capítulo apresenta uma revisão sistemática sobre técnicas de anonimização em Redes Definidas por Software. Conforme é apresentado nesta revisão, anonimização em SDN corresponde a um tema com pesquisa e publicações recentes, dos últimos 5 anos, e desta forma, ainda não se encontra com o estado da técnica em andamento, por isso, a ausência de uma revisão tecnológica.

Para realizar um estudo e mapeamento do estado da arte acerca de técnicas utilizadas para anonimização de dados de tráfego em Redes Definidas por Software, foi adotado um método de mapeamento sistemático, que de acordo com a abordagem de (MATTSON, 2008), consiste em definir questões de pesquisa e realizar busca de artigos científicos sobre a área de interesse. A partir dos artigos obtidos, foi realizada uma triagem que possibilitasse a seleção dos estudos relevantes, e assim, realizar a extração dos dados e mapear os resultados.

Desta forma, com o objetivo de seguir a metodologia de mapeamento sistemático, é descrito neste capítulo como foi realizado o processo de busca e seleção dos estudos. Para isto, foi necessário, inicialmente, definir as questões de pesquisa, a estratégia de busca e seleção e os critérios para triagem dos estudos.

4.1 QUESTÕES DE PESQUISA

Com o propósito de atingir o objetivo proposto por esse mapeamento, as seguintes questões foram elaboradas para pesquisa:

- *Q1*) Em quais anos há publicações sobre anonimização em redes SDN?
- *Q2*) Quais autores estão publicando na área?
- *Q3*) Quais países estão publicando na área?
- *Q4*) Quais controladores SDN foram utilizados para anonimizar os dados do tráfego da rede?
- *Q5*) Quais técnicas de anonimização são utilizadas em tráfego para redes SDN?
- *Q6*) Quais dados (IP, MAC, portas) do tráfego foram anonimizados em redes SDN?

4.2 ESTRATÉGIA DE BUSCA E DE SELEÇÃO

Para a execução da busca foram selecionadas as bases de dados na área de computação: *IEEE Xplore (IEEE)*, *Digital Library (ACM)*, *Science Direct (SD)* e *Springer* e a Biblioteca Digital Brasileira de Computação (BDBComp). Com exceção da última base mencionada, as demais foram acessadas a partir do portal Scopus, que realiza uma busca nas bases mencionadas, agrupando todos os resultados em uma única consulta.

Durante o processo de execução das consultas foram adotadas as ferramentas de filtragem disponibilizadas por cada uma, visando considerar apenas título, resumo e palavras-chave dos artigos, para assim, reduzir a quantidade de resultados retornados para anonimização em SDN. Porém, devido a base BDBComp não apresentar ferramentas para filtragens dos resultados, foi realizada uma busca simples.

Com o propósito de realizar a pesquisa dos artigos nas bases supracitadas, foram definidas as seguintes palavras chaves:

- Inglês: Inicialmente foram adotadas três palavras-chave “software-defined networking”, “anonymization” e “sdn”. Porém, durante as pesquisas constatou-se que alguns resultados não era retornados por dois motivos: (1) alguns trabalhos trazem o termo “software defined networking” e (2) também há trabalhos que utilizam as variações de *anonymization*, como “anonymous” e “anonymity”.
- Português: Inicialmente foram adotadas cinco palavras-chave “anonimização”, “sdn”, “software-defined network”, “software defined network”, “redes definidas por software”. Os termos em inglês foram adicionados porque a maioria dos trabalhos apresenta SDN escrito em língua inglesa. Para expandir a pesquisa em português também foram adicionados os termos “anônimo” e “anonimato”.

Com isso, os termos de busca ficaram definidos da seguinte forma:

- Inglês: ((anonymization OR anonymous OR anonymity) AND (sdn OR "software-defined network" OR "software defined network"))
- Português: ((anonimização OR anônimo OR anonimato) AND (sdn OR "software-defined network" OR "software defined network" OR "redes definidas por software"))

A partir destes termos de busca foram realizadas, em maio de 2016, as consultas nas bases indicadas, retornando um total de 67 artigos, conforme apresentado na Tabela 4.1, em que é possível verificar a quantidade de artigos obtidos com a consulta com termos em Inglês e Português. Importante destacar que a consulta também foi realizada na base brasileira BDBComp, não retornando resultados.

Tabela 4.1 – Resultados das buscas nas bases de dados utilizando os termos de busca

Bases de dados	Inglês	Português
ACM	8	0
BioMed	1	0
IEEE	21	0
Institute for Computer Sciences	1	0
Institute of Agricultural	1	0
Pol J Radiol	1	0
RSNA	1	0
ScienceDirect	12	0
SERSC	1	0
South Atlantic Quartely	1	0
Springer	17	0
Susan Leong	1	0
Universiti Putra Malaysia Press	1	0
BDBComp	0	0

4.3 CRITÉRIOS DE SELEÇÃO

Com o objetivo de filtrar apenas os artigos relevantes para a área de pesquisa deste mapeamento sistemático, foram definidos critérios para inclusão e critérios para exclusão desses artigos. Os seguintes critérios foram utilizados para inclusão:

1. Foram incluídos artigos que apresentam técnicas de anonimização em Redes Definidas por Software.
2. Foram incluídos artigos que abordam anonimização como uma forma de prover segurança em um controlador para Redes Definidas por Software. Nestes artigos há uma abordagem sobre outras formas de segurança, o que permite uma comparação dos resultados obtidos e uma análise das técnicas.

A confirmação dos critérios de inclusão foi dada a partir da análise do resumo e conclusão de cada um dos artigos encontrados. Em casos inconclusivos, a leitura da introdução também foi realizada para validar os critérios de inclusão.

Também foi realizada a análise dos artigos quanto aos critérios de exclusão, foram aplicados os critérios de exclusão listados a seguir:

1. Foram excluídos os artigos duplicados.
2. Foram desconsiderados os artigos que não disponibilizavam acesso à íntegra do trabalho.
3. Foram desconsiderados os trabalhos que não abordavam anonimização como uma forma de segurança em Redes de Computadores.

Terminada a aplicação dos critérios de inclusão e exclusão dos artigos encontrados nas bases de pesquisa, foi verificado que do total de 67 artigos, foram mantidos apenas 18 para compor os estudos primários, o que significa 26,9% do total.

De acordo com os critérios de exclusão aplicados, os seguintes trabalhos foram descartados conforme demonstrado na Tabela 4.2

Na Tabela 4.3 é apresentado o quantitativo final de trabalhos selecionados por base de pesquisa após a aplicação dos critérios de inclusão.

A Figura 4.1 apresenta o resumo da contribuição de cada base para o total de 19 estudos primários selecionados. Pode ser observada uma grande percentagem de representatividade da base *IEEE Xplore*, com os artigos desta base correspondendo a 53% do total de selecionados, enquanto *ACM* representa 10% e *ScienceDirect* tem 37% dos trabalhos.

Após a seleção, os estudos primários foram encaminhados para leitura aprofundada e análise, os resultados dessa etapa podem ser encontrados na seção seguinte.

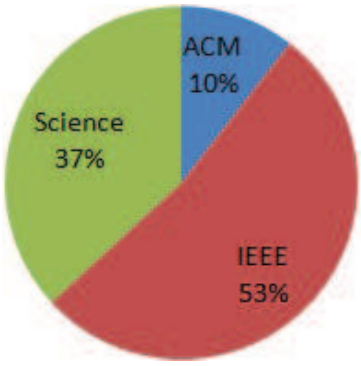
Tabela 4.2 – Artigos descartados de acordo com critérios de exclusão

Critérios	Trabalhos Excluídos
Duplicados	(KONSTANTINOV et al., 2015)
Indisponíveis	(W et al., 2015), (SOO; SAMSUDIN; GOH, 2002), (SHUANGYU et al., 2014), (KONSTANTINOV et al., 2015), (GOLLASCH; ROSENTHAL, 2006)
Fora do tema	(TAN et al., 2011), (NIA et al., 2015), (CHAKRABARTY; ENGELS; THATHAPUDI, 2015), (ROSSITER, 2015), (ROMEO et al., 2015), (BREMLER-BARR et al., 2014), (SUTIKNO; STIAWAN; SUBROTO, 2014), (SIEK; VACHHARAJANI, 2008), (KAYDAN; KOZAR; ERKILIC, 2014), (LEONG, 2013), (ZHANG et al., 2016), (SAMSUDIN; SURYANA, 2009), (CUSTERS, 2007), (YANBING et al., 2016), (WANG; CHEN; XING, 2015), (MIET, 2010), (GARG; GARG, 2016), (GABOR; SZABO, 2013), (SAHANI et al., 2012), (ZHANG et al., 2015), (GERSON, 2010), (DESHPANDE et al., 2015), (JOSHI; CHATURVEDI, 2013), (SCHWEITZER; HANNAN; COREN, 2012), (DU et al., 2014), (CARNIELLI; AIASH, 2015), (SANATINIA; NARAIN; NOUBIR, 2013), (BAIARDI et al., 2004), (BAIARDI et al., 2005)

Tabela 4.3 – Resultados das buscas nas bases de dados utilizando os termos de busca e critérios de inclusão

Bases de dados	Inglês	Português
ACM	2	0
IEEE	10	0
ScienceDirect	7	0

Figura 4.1 – Contribuição de trabalhos primários por base de dados



4.4 ANÁLISE DOS RESULTADOS

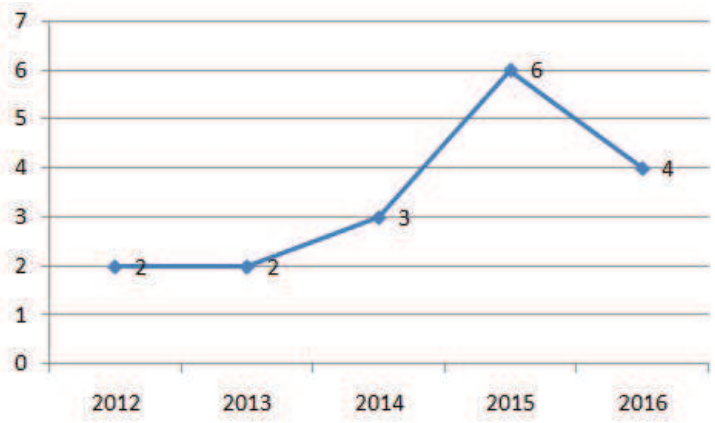
Nesta seção, são apresentados os resultados da análise dos estudos primários, respondendo assim às questões de pesquisa deste mapeamento.

4.4.1 EM QUAIS ANOS HÁ PUBLICAÇÕES SOBRE ANONIMIZAÇÃO EM REDES SDN

O objetivo dessa questão é verificar o crescimento anual de publicações sobre o assunto e vislumbrar perspectivas futuras.

Na Figura 4.2 é apresentado o crescimento de publicações ao longo dos anos. É verificado que a abordagem de serviço de segurança em SDN através de anonimização está em crescimento nos últimos anos, sendo iniciado em 2012 com apenas duas publicações, tendo em 2015 um total de 6 publicações, e até a metade de 2016 já há um total de 4 publicações.

Figura 4.2 – Publicações por ano sobre anonimização em SDN



De acordo com o ano, os trabalhos indicados no gráfico correspondem a:

- 2012: (JAFARIAN; AL-SHAER; DUAN, 2012), (MENDONCA; SEETHARAMAN; OBRACZKA, 2012)
- 2013: (CHON et al., 2013), (HASEGAWA, 2013)
- 2014: (BALASHOV et al., 2014), (HOFSTEDE et al., 2014), (REBOLLO-MONEDERO et al., 2014)
- 2015: (IRFAN; TAJ; MAHMUD, 2015), (ALI et al., 2015), (CHEN et al., 2015), (ALSMADI; XU, 2015), (CHAVEZ; STOUT; PEISERT, 2016), (ACCETTURA; NEGLIA; GRIECO, 2015)
- 2016: (LI; DOH; CHAE, 2016), (LI; MENG; KWOK, 2016), (KHAN et al., 2016), (CHEN et al., 2016)

4.4.2 QUAIS AUTORES ESTÃO PUBLICANDO NA ÁREA?

Esta pergunta tem o objetivo de verificar se há autores com mais de uma publicação em relação ao tema, e assim, identificar a abordagem utilizada para indicar um serviço de anonimização para SDN.

Com a leitura dos artigos selecionados com base nos critérios de inclusão verificou-se que, devido a ser um tema com publicações recentes, em que as pesquisas para aprimoramento dos serviços estão em fase de consolidação, não foram encontrados autores com mais de um artigo publicado.

4.4.3 QUAIS PAÍSES ESTÃO PUBLICANDO NA ÁREA?

Com esta pergunta tem por objetivo analisar as regiões que estão realizando pesquisas sobre este tipo de serviço de segurança para SDN.

De acordo com a Figura 4.3 é possível verificar que o país que mais concentra pesquisas sobre o assunto é os Estados Unidos, correspondendo a aproximadamente 33% das publicações. Analisando de forma continental, constata-se que América, Ásia e Europa possuem quantitativo semelhante de publicações, conforme pode ser visualizado no gráfico da Figura 4.4, em que Ásia possui 7 publicações, América e Europa possuem 6 publicações cada continente.

Figura 4.3 – Publicações por países sobre anonimização em SDN

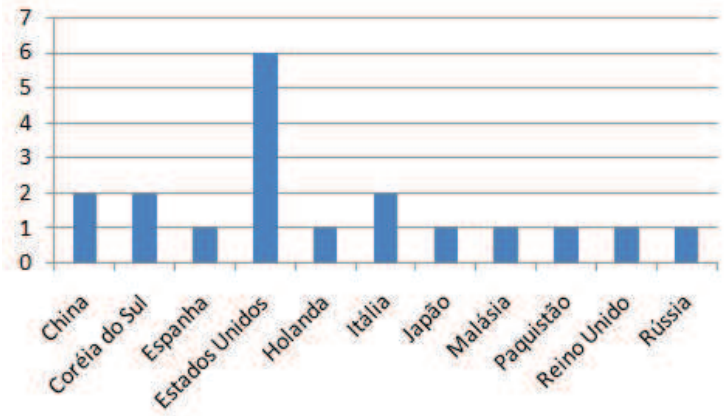
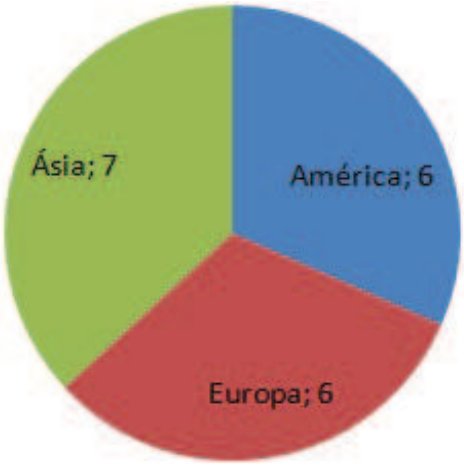


Figura 4.4 – Publicações por continentes sobre anonimização em SDN



4.4.4 QUAIS CONTROLADORES SDN FORAM UTILIZADOS PARA ANONIMIZAR OS DADOS DO TRÁFEGO DA REDE?

O objetivo desta pergunta é verificar quais controladores SDN são utilizados nas pesquisas para desenvolvimento de serviços de anonimização.

Dentre os 19 trabalhos selecionados através dos critérios de inclusão, 11 possuem serviços de anonimização para SDN. Os trabalhos que possuem serviços de anonimização são: (LI; DOH; CHAE, 2016), (ALI et al., 2015), (LI; MENG; KWOK, 2016), (JAFARIAN; AL-SHAER; DUAN, 2012), (MENDONCA; SEETHARAMAN; OBRACZKA, 2012), (HOFSTEDE et al., 2014), (REBOLLO-MONEDERO et al., 2014), (CHEN et al., 2016), (BAIARDI et al., 2005), (BAIARDI et al., 2004), (CHAVEZ; STOUT; PEISERT, 2016).

Dos trabalhos publicados sobre serviços de anonimização para SDN, apenas 2 indicam o controlador utilizado para validar o serviço. Os trabalhos de (JAFARIAN; AL-SHAER; DUAN, 2012) e (MENDONCA; SEETHARAMAN; OBRACZKA, 2012) utilizam o controlador Nox (NATASHA et al., 2008).

A não divulgação dos controladores utilizados nos demais trabalhos pode-se considerar um obstáculo para continuidade de pesquisas, devido a dificuldade de replicação dos trabalhos para análise e validação.

4.4.5 QUAIS TÉCNICAS DE ANONIMIZAÇÃO SÃO UTILIZADAS EM TRÁFEGO PARA REDES SDN?

Para que seja possível anonimizar informações é necessário adotar uma técnica de anonimização.

Nos trabalhos selecionados neste artigo para estudo primário, 8 trabalhos usam a técnica de anonimização por "Deslocamento Aleatório", conforme apresentado em (LI; DOH; CHAE, 2016), (JAFARIAN; AL-SHAER; DUAN, 2012), (MENDONCA; SEETHARAMAN; OBRACZKA, 2012), (REBOLLO-MONEDERO et al., 2014), (CHEN et al., 2016), (BAIARDI et al., 2005), (BAIARDI et al., 2004), (CHAVEZ; STOUT; PEISERT, 2016).

A remoção ou modificação randômica dos endereços IP de um pacote da rede, apesar de anonimizá-lo, deixa-o inútil em relação a permitir que estudos possam ser realizados em relação ao fluxo (HOFSTEDE et al., 2014). Isto ocorre devido a perda das informações referentes a sub-rede, impedindo a análise do fluxo do pacote na rede.

Assim, importante verificar que através deste questionamento apresenta-se outra linha de pesquisa a ser investigada em SDN, que corresponde a criar um serviço que utilize uma técnica de anonimização que preserve o prefixo dos endereços, permitindo que os dados compartilhados do fluxo da rede não sejam inúteis.

4.4.6 QUAIS DADOS DO TRÁFEGO FORAM ANONIMIZADOS EM REDES SDN?

Com base nos trabalhos selecionados na pesquisa primária, verificou-se que os dados anonimizados do tráfego da rede foram IP, portas e tráfego. Importante destacar que nenhum dos trabalhos anonimizou os endereços MAC das estações.

A Tabela 4.4 apresenta quais dados da rede foram anonimizados por trabalho selecionado na pesquisa primária. É possível identificar que a maioria dos trabalhos anonimiza o endereço IP; e em relação a portas, apenas 1 trabalho a anonimiza.

Este capítulo apresentou uma revisão sistemática sobre serviços de anonimização para SDN, realizada através de consultas à Base de Pesquisas de períodicos. Esta pesquisa respondeu a questionamentos sobre SDN com o objetivo de obter o Estado da Arte sobre os

Tabela 4.4 – Campos anonimizados pelos serviços em SDN

Dados Anonimizados	Trabalhos Primários
IP	(LI; DOH; CHAE, 2016), (ALI et al., 2015), (LI; MENG; KWOK, 2016), (JAFARIAN; AL-SHAER; DUAN, 2012), (MENDONCA; SEETHARAMAN; OBRACZKA, 2012), (HOFSTEDE et al., 2014), (CHAVEZ; STOUT; PEISERT, 2016)
Portas	(CHAVEZ; STOUT; PEISERT, 2016)
Tráfego	(REBOLLO-MONEDERO et al., 2014), (CHEN et al., 2016), (BAIARDI et al., 2005), (BAIARDI et al., 2004), (CHAVEZ; STOUT; PEISERT, 2016)

serviços desenvolvidos e as principais técnicas utilizadas, para desta forma, poder comparar com o serviço desenvolvido e implementado neste trabalho.

5 SERVIÇO DE ANONIMIZAÇÃO

Neste capítulo é apresentado o anonimizador BomIP, desenvolvido neste trabalho, responsável por realizar a anonimização de endereços IP em um controlador SDN, mantendo a rastreabilidade dos endereços, e assim, permitindo posterioranálise da rede.

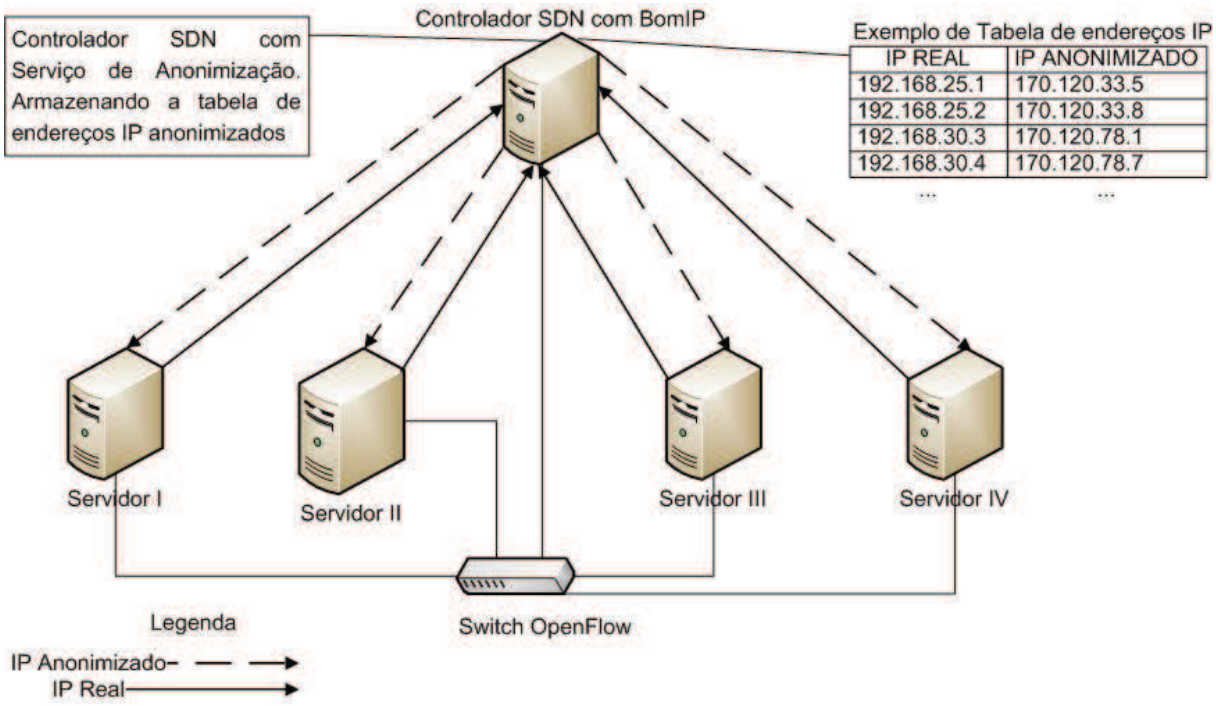
5.1 ARQUITETURA DO SERVIÇO

O anonimizador BomIP, desenvolvido e implementado neste trabalho, consiste em uma classe que realiza a anonimização dos endereços IP dos *hosts* (estações e serviços) em uma SDN. A arquitetura é apresentada na Figura 5.1. Importante destacar que neste exemplo é apresentado apenas um *switch* OpenFlow, mas no Capítulo 8 é apresentado um Estudo de Caso com mais de um *switch* OpenFlow.

Na arquitetura de funcionamento, um *host* contendo um controlador SDN, como por exemplo, RunOS (ALEXANDER et al., 2015), gerencia a rede formada por outras estações e serviços conectados a um *switch* que funciona com protocolo OpenFlow. As estações, enviam pacotes contendo seus endereços IP para o controlador, através do *switch* OpenFlow, para que o controlador estabeleça a rota que deve ser realizada no roteamento dos pacotes para os respectivos endereços de destino.

Quando o endereço destes dispositivos é recebido pelo controlador, através do processamento do pacote, ele anonimiza o endereço IP, armazenando o valor anonimizado na Tabela de Endereços, garantindo assim a segurança da informação em caso de invasão ao controlador, uma vez que as informações na Tabela de Endereços encontram-se ano-

Figura 5.1 – Arquitetura de uma SDN com Serviço BomIP



nimizadas. Este processo de anonimização é realizado de forma a permitir a criação de novas sub-redes, e assim, manter a rastreabilidade e análise dos pacotes.

5.2 ANONIMIZADOR BOMIP

O anonimizador BomIP é uma classe que implementa o serviço de anonimização proposto neste trabalho, foi desenvolvida utilizando a Linguagem de Programação C (BRIAN; DENNIS, 1989). Esta classe, pode ser utilizada como um *plugin* em aplicações de análise de pacotes em uma rede, permitindo a anonimização dos endereços IP. Os pacotes podem ser fornecidos através do fluxo em tempo real da rede, ou por meio de um arquivo contendo os *traces* da rede.

A Linguagem C foi escolhida para prover melhor desempenho durante o processamento dos dados recebidos na rede, para desta forma, minimizar possíveis perdas de pacotes durante o fluxo de dados. Para facilitar o acesso aos dados de um pacote, o anonimizador BomIP utiliza a biblioteca Libpcap (VAN; MCCANNE, 2009).

Libpcap é uma biblioteca *opensource* que provê uma interface de alto nível para trabalhar com captura de pacotes em rede, tendo sido desenvolvida em 1994 na Universidade de Berkeley e posteriormente aprimorada (MARTIN, 2008). O principal objetivo era criar

uma API independente que eliminasse a necessidade de um sistema de captura. Esta biblioteca foi desenvolvida para trabalhar com C e C++, no entanto, há mecanismos que permitem utilizá-la com *Perl*, *Python*, *Java*, *CSharp* e *Ruby*.

A biblioteca Libpcap funciona na maioria dos sistemas operacionais baseados em UNIX, também existindo uma versão para Windows denominada Winpcap. Atualmente, a biblioteca Libpcap é gerenciada pelo grupo Tcpdump.

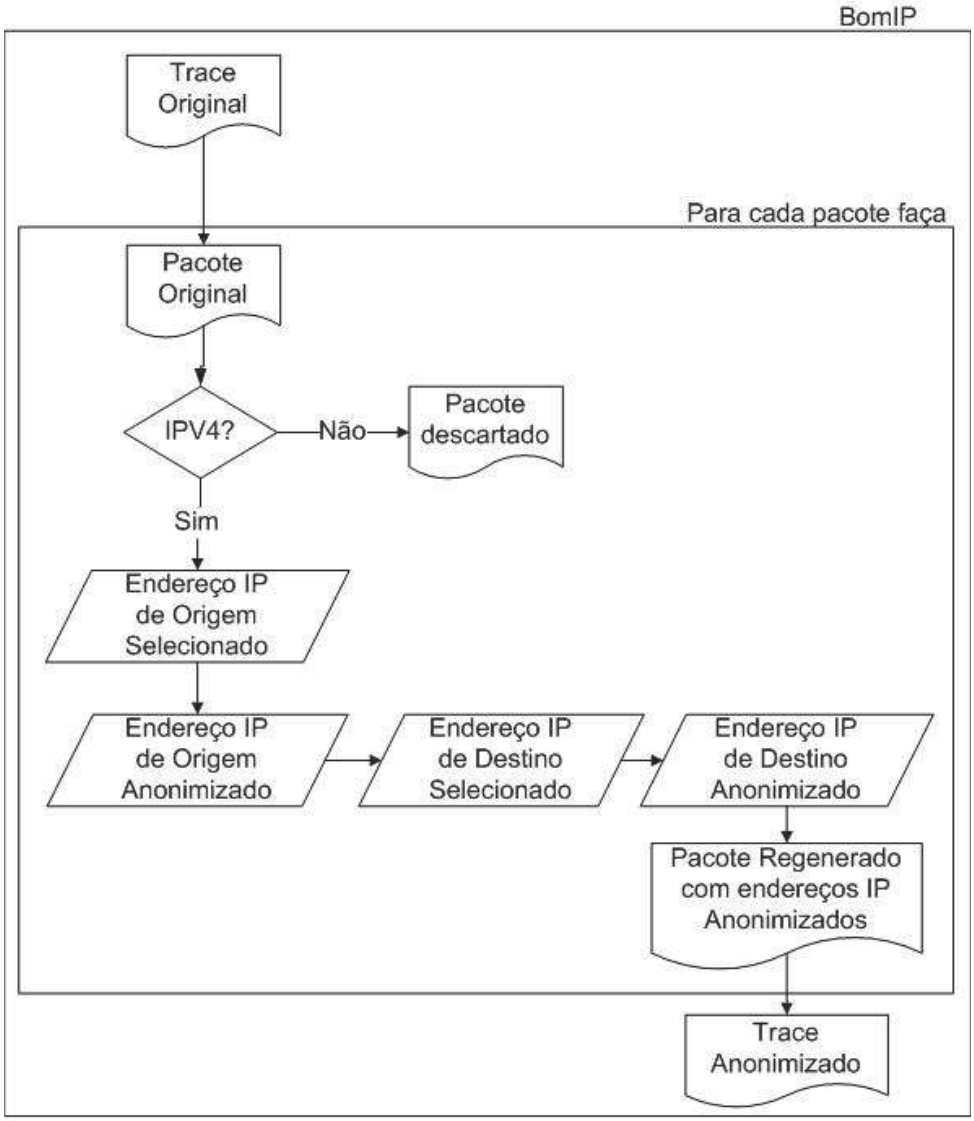
Durante o processo de anonimização de endereços IP, o BomIP utiliza a técnica de preservação de prefixo (JINLIANG et al., 2004). Esta técnica diz que dois endereços IP $a = a_1a_2...a_n$ e $b = b_1b_2...b_n$ possuem o mesmo $k - bit$ prefixo quando $0 \leq k \leq n$, se $a_1a_2...a_k = b_1b_2...b_k$ e $a_{k+1} \neq b_{k+1}$, indicando assim, que os endereços IP possuem o mesmo prefixo. Isto permite que, ao realizar a anonimização dos endereços IP em um *trace offline* ou em tempo real, novas sub-redes sejam criadas de forma anonimizadas, o que permite uma posterior análise do tráfego da rede devido à manutenção da rastreabilidade, garantindo verificar a existência de possíveis ataques à rede, como ataque de Negação de Serviço, dentre outros.

Assim, o anonimizador BomIP utiliza a técnica de preservação de prefixo, através de um algoritmo de randomização dos valores gerados para substituir as informações reais. Por utilizar duas técnicas para atingir o objetivo de anonimizar a informação, o BomIP é classificado como um anonimizador híbrido.

O funcionamento geral do BomIP é apresentado na Figura 5.2. Um arquivo de *trace* contendo o tráfego ou a captura em tempo real da rede é fornecido como parâmetro de entrada para o anonimizador. Neste momento, o BomIP analisa, separadamente, cada um dos pacotes, de acordo com a ordem que estiverem no arquivo fornecido ou chegarem à rede em tempo real. Para cada um dos pacotes, o BomIP extrai os endereços IP de origem e destino, anonimizando-os. Se o parâmetro de entrada for um arquivo, ao término do processo, é gerado um novo arquivo com todos os endereços IP anonimizados. No caso de captura em tempo real, os endereços IP dos serviços e *hosts* são anonimizados. Em ambas as situações, o parâmetro de entrada sendo um arquivo ou uma captura em tempo real, a rastreabilidade das conexões é mantida, permitindo o compartilhamento do *trace* para estudos e outras atividades.

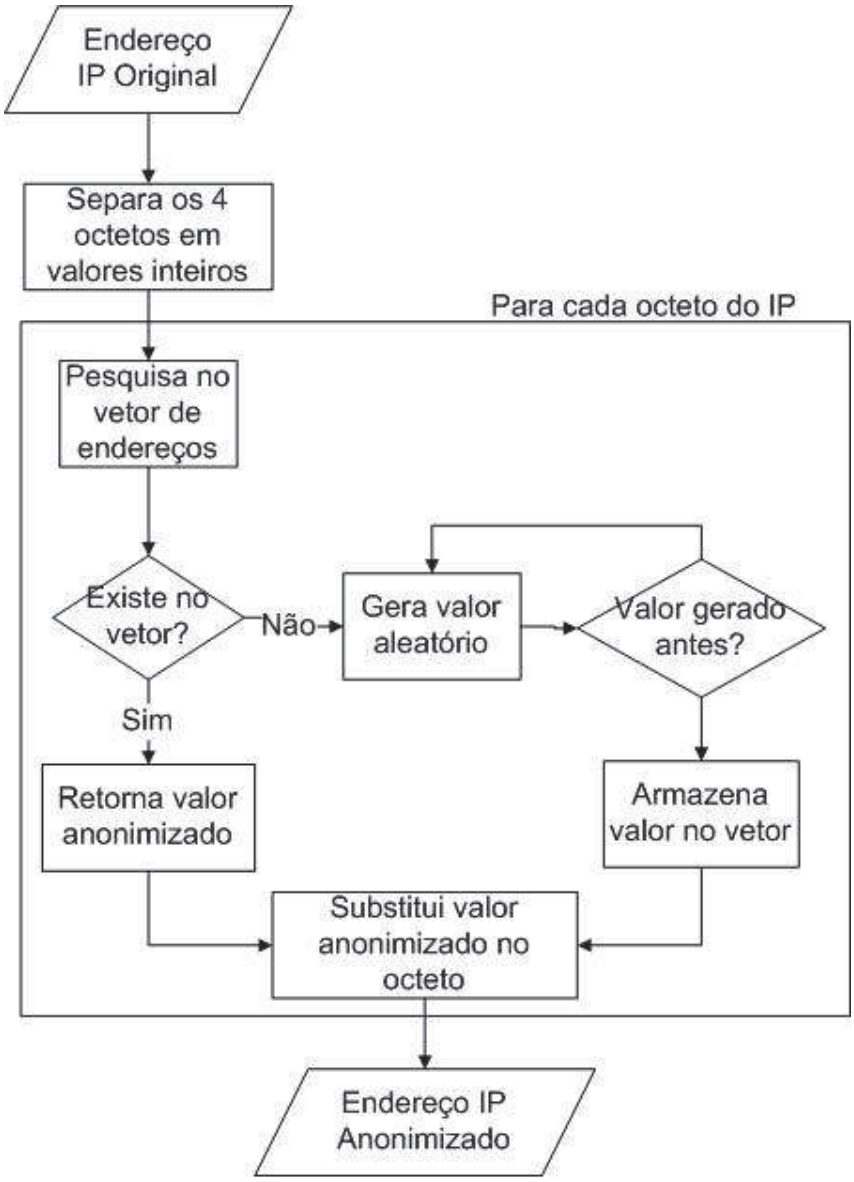
O processo de anonimização do BomIP é apresentado na Figura 5.3, este processo é realizado para os endereços IP de origem e destino, sendo análogo para ambos os tipos de endereço. Em um primeiro momento, o endereço IP tem os quatro octetos separados,

Figura 5.2 – Fluxo de funcionamento do anonimizador BomIP



porque o processo de anonimização é realizado separadamente para cada um dos octetos. Em seguida a anonimização é iniciada para o primeiro octeto x; assim, no vetor que armazena os valores anonimizados, consulta-se na posição x se já existe um valor gerado. Se não existir um valor, na posição x do vetor consta o *flag* de valor -1, e então um valor y é gerado randomicamente. Com este valor y é realizada uma consulta no vetor para verificar se o mesmo já existe, se o mesmo já existir, um novo valor randomicamente é gerado, e este processo é interrompido apenas quando um valor gerado ainda não existir no vetor. Quando um y não existente no vetor é gerado para posição x, este valor é retornado para formar o novo endereço IP anonimizado, e sempre que este valor x aparecer em qualquer octeto de qualquer endereço IP (origem ou destino), será substituído pelo valor y.

Figura 5.3 – Processo de anonimização de endereços IP do BomIP



Importante observar que os valores são armazenados em um vetor, e como são armazenadas as possibilidades de valores em um octeto, o espaço de armazenamento corresponde a um vetor de 256 posições (valores de 0 a 255), isto porque não são armazenados os endereços IP, o que ocasiona um ganho de espaço na memória para o processamento. Este vetor é inicializado com todas as posições contendo o *flag* de valor -1, o que identifica que para uma dada posição, ainda não existe um valor randomizado para ser utilizado na anonimização.

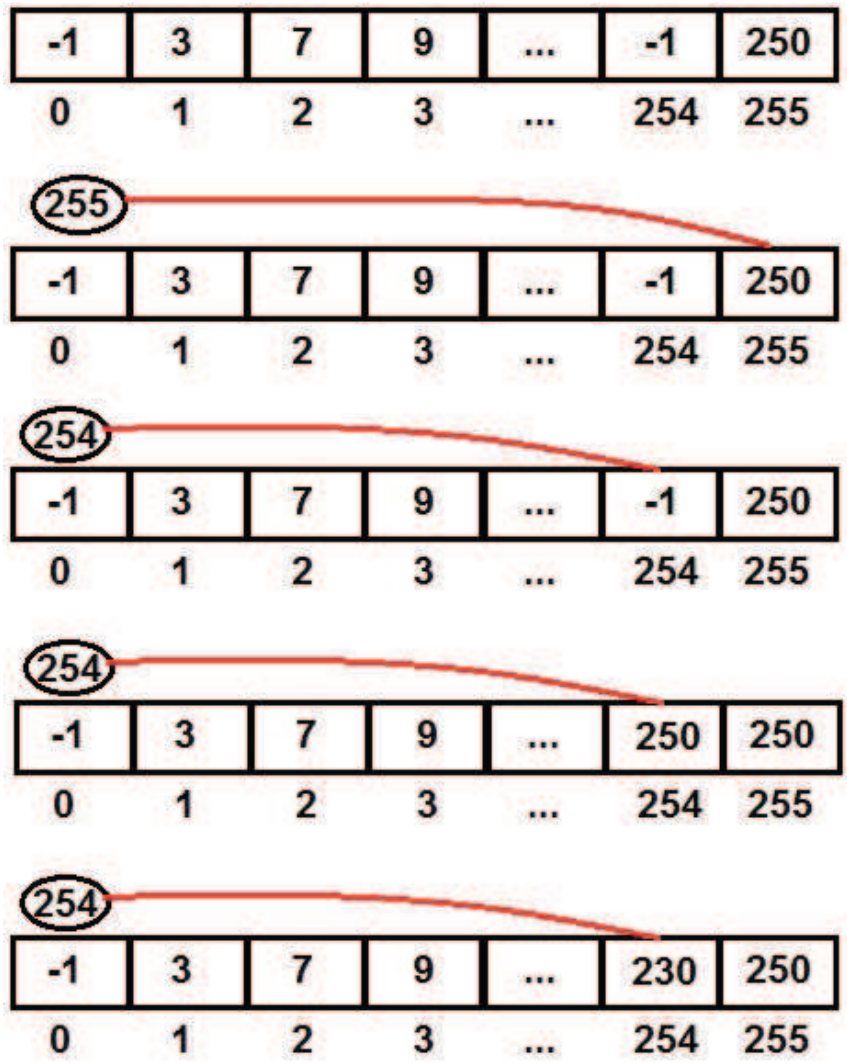
A Figura 5.4 ilustra este processo, em que as posições 0 e 254 ainda não foram utilizadas para anonimização, ou seja, nenhum endereço IP, até o momento do processamento não conteve um octeto que fossem os valores 0 ou 254, enquanto que as demais posições já

foram utilizadas e por isso possuem valores para anonimização. Por exemplo, sempre que um octeto tiver o valor 3, será anonimizado pelo valor 9.

Ainda sobre o exemplo da Figura 5.4, quando um octeto de valor 255 tiver que ser anonimizado pela primeira vez, um valor aleatório é gerado, neste caso foi gerado o valor 250. O mesmo foi adicionado ao vetor de posições, e assim, sempre que existir um octeto de valor 255, ele será anonimizado por 250.

Mesmo processo vale para um octeto de valor 254, no exemplo da Figura 5.4. Primeiramente, ele recebeu o valor 250, porém, o valor 250 já é relacionado com o valor real 255, e desta forma, um novo valor aleatório é gerado, resultado em 230. Como este valor ainda não existe no vetor, ele é adicionado, e sempre que um octeto tiver um valor 254, será anonimizado por 230.

Figura 5.4 – Exemplo de Vetor Anonimizado



Assim, de acordo com os diagramas apresentados na Figura 5.2 e na Figura 5.3, é possível descrever o pseudocódigo do BomIP. O procedimento *InicializaVetor* é responsável por iniciar o vetor de anonimização com todas as 256 posições contendo o *flag* de valor -1, conforme apresentado no Algoritmo 5.1, assim, quando o aplicativo necessitar consultar se já existe um valor para anonimizar determinado octeto, basta verificar se na posição correspondente consta o *flag* de valor -1, em caso afirmativo, deve-se gerar um valor. A função *ExisteValor*, apresentada no Algoritmo 5.2 impede que dois ou mais possíveis valores de octetos sejam anonimizados com a mesma referência, por exemplo, impedir que 50 e 70 sejam anonimizados para serem substituídos pelo valor 90. Caso isso acontecesse, o arquivo contendo os pacotes iria perder a rastreabilidade, visto que as sub-redes geradas não seriam consistentes. A função *SeparaOctetos*, Algoritmo 5.3 é responsável por separar os quatro octetos do endereço IP. E por fim, o procedimento *AnonimizaIP*, apresentado no Algoritmo 5.4 é quem realiza a anonimização do endereço IP e gera um novo valor para ser armazenado. O Algoritmo 5.5 é apenas o método para iniciar todo o processo de anonimização a partir do fornecimento de um arquivo contendo os pacotes de uma captura em rede.

Algoritmo 5.1 Algoritmo BomIP - Procedimento de inicialização de vetor

```
1: procedure InicializaVetor(int vetor[256])
2:     for i <= 255 do
3:         vetor[i] = -1
```

Algoritmo 5.2 Algoritmo BomIP - Função para verificar existência de valor

```
1: function ExisteValor(int vetor[256], int x):boolean
2:     booleanExiste = False
3:     inti = 0
4:     while (i <= 255)and(Existe == False) do
5:         if vetor[i] == x then
6:             Existe = True
7:         i = i + 1
8:     return = Existe
```

Algoritmo 5.3 Algoritmo BomIP - Função para separar os quatro octetos do endereço IP

```
1: function SeparaOctetos(string IP, int posicaoOcteto):int
2:     intocteto = IP[posicaoOcteto]
3:     return = octeto
```

Algoritmo 5.4 Algoritmo BomIP - Função para anonimizar o endereço IP

```
1:  procedure AnonimizaIP(pacote)
2:      stringIpOrigem = IP(pacote)
3:      intIP1normal = SeparaOctetos(IpOrigem, 1)
4:      intIP2normal = SeparaOctetos(IpOrigem, 2)
5:      intIP3normal = SeparaOctetos(IpOrigem, 3)
6:      intIP4normal = SeparaOctetos(IpOrigem, 4)
7:      if vetor(IP1normal) ≠ −1 then
8:          IP1anon = vetor(IP1normal)
9:      else
10:         IP1anon = randomiza()
11:         while ExisteValor(IP1anon) == True do
12:             IP1anon = randomiza()
13:         vetor(IP1normal) = IP1anon
14:     if vetor(IP2normal) ≠ −1 then
15:         IP2anon = vetor(IP2normal)
16:     else
17:         IP2anon = randomiza()
18:         while ExisteValor(IP2anon) == True do
19:             IP2anon = randomiza()
20:         vetor(IP2normal) = IP2anon
21:     if vetor(IP3normal) ≠ −1 then
22:         IP3anon = vetor(IP3normal)
23:     else
24:         IP3anon = randomiza()
25:         while ExisteValor(IP3anon) == True do
26:             IP3anon = randomiza()
27:         vetor(IP3normal) = IP3anon
28:     if vetor(IP4normal) ≠ −1 then
29:         IP4anon = vetor(IP4normal)
30:     else
31:         IP4anon = randomiza()
32:         while ExisteValor(IP4anon) == True do
33:             IP4anon = randomiza()
34:         vetor(IP4normal) = IP4anon
35:     IpOrigemAnonimizado = concatenacao(IP1anon, "‘.”′)
36:     IpOrigemAnonimizado = concatenacao(IP2anon, "‘.”′)
37:     IpOrigemAnonimizado = concatenacao(IP3anon, "‘.”′)
38:     IpOrigemAnonimizado = concatenacao(IP4anon, "‘.”′)
```

Este capítulo apresentou o serviço de anonimização BomIP, desenvolvido e implementado neste trabalho, com o objetivo de realizar a mitigação dos ataques de uma rede SDN através da anonimização dos endereços IP. No capítulo seguinte é apresentado o

Algoritmo 5.5 Algoritmo BomIP - Procedimento para ler arquivo e iniciar processo de anonimização

```
1: procedure MAIN(ArquivoTrace)
2:     InicializaVetor
3:     Realizaleituradearquivofornecidocomtraces
4:     while existirpacotesnoarquivo do
5:         AnonimizaIP(pacote)
```

controlador SDN RunOS, utilizado nos Estudos de Caso deste trabalho, para mostrar o funcionamento do serviço BomIP em tempo real em uma rede.

6 CONTROLADOR RUNOS

Neste capítulo é apresentado o controlador OpenFlow RunOS, usado para gerenciar uma Rede Definida por Software, desenvolvido em Linguagem de Programação C++. Baseado no controlador RunOS foi desenvolvida a arquitetura proposta de um controlador com serviço de anonimização da rede, para desta forma, mitigar as tentativas de ataques.

6.1 RUNOS

RunOS (ALEXANDER et al., 2015) é um controlador OpenFlow, desenvolvido em C++ e em 2014, com o objetivo de ter uma alta performance, combinando técnicas de programação para atingir este objetivo, permitindo que o controlador tenha qualidade, programabilidade e usabilidade.

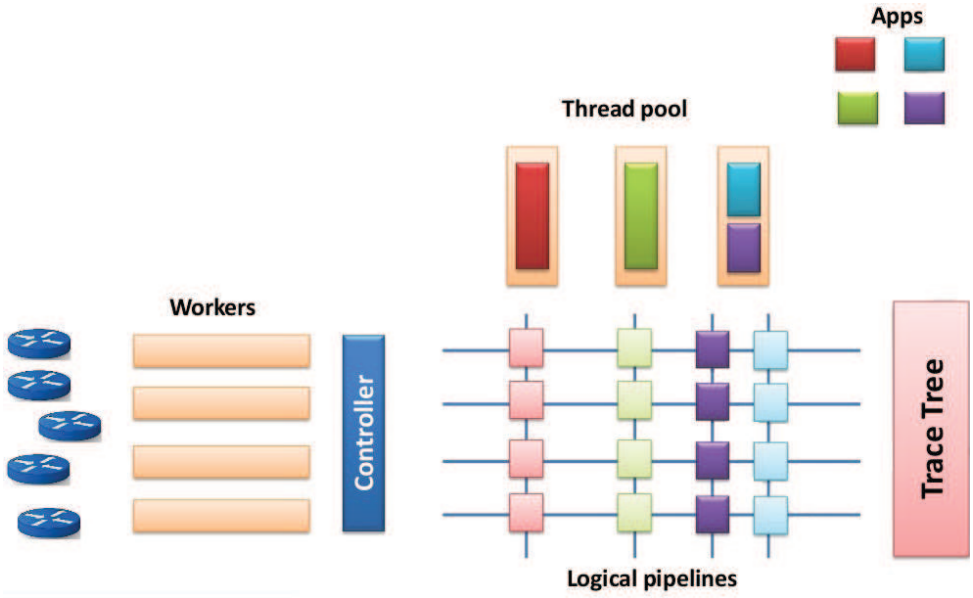
Para prover alta performance, RunOS é desenvolvido em Linguagem C++, por apresentar melhor desempenho que outras, como Java, por produzir um código completamente compilado, e por permitir trabalhar com baixo nível de programação voltado para redes e por permitir trabalhar com *threads*.

O controlador RunOS utiliza *threads* dedicadas para realizar a comunicação com os *switches* da rede e as aplicações. Esta comunicação com os *switches* é realizada através do uso da biblioteca Libfluid (ALLAN; CHRISTIAN; FABIO, 2014), utilizada para prover aos controladores SDN uma interface, ou seja, fornecer comandos básicos para comunicação entre os controladores e os equipamentos da rede.

A arquitetura do RunOS é apresentada na Figura 6.1. O controlador é inicializado com o número desejado de *threads* informada pelo usuário, em seguida os serviços são

iniciados. As aplicações são iniciadas e distribuídas entre as *threads*, e por fim, a ordem dos eventos é determinada e processada pelas aplicações.

Figura 6.1 – Arquitetura do RunOS. Fonte: Adaptado de (ALEXANDER et al., 2015)



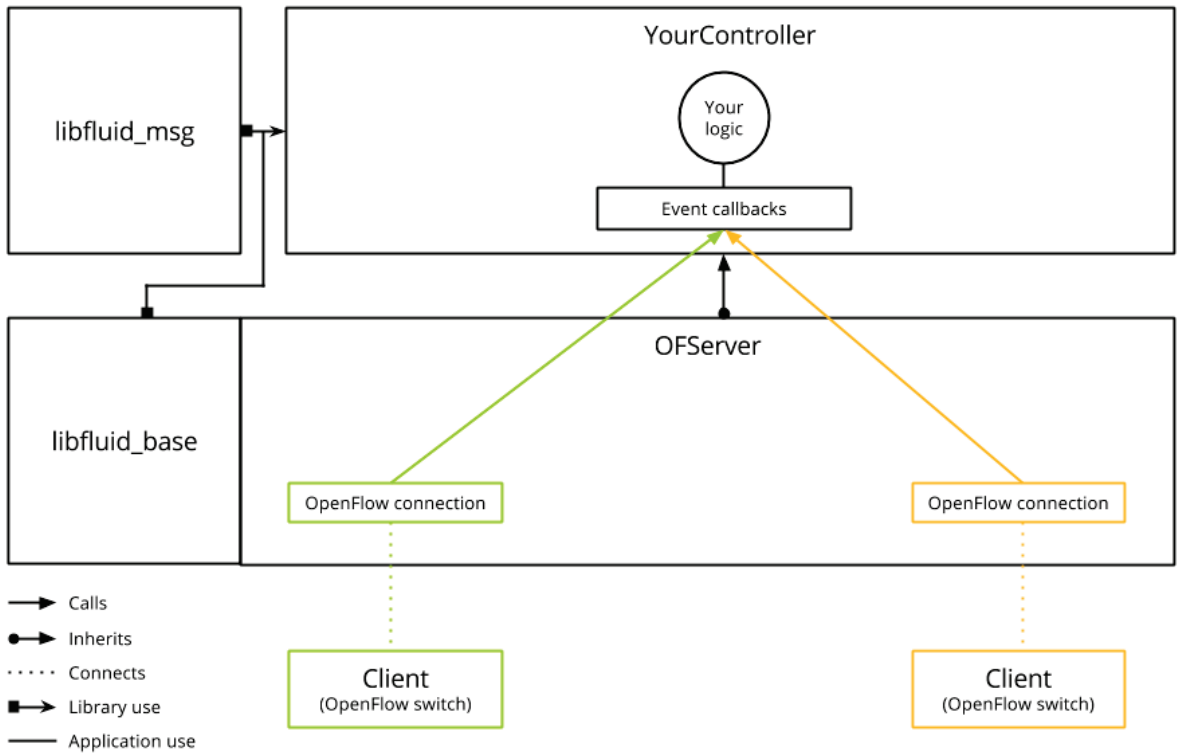
6.1.1 LIBFLUID

Libfluid (ALLAN; CHRISTIAN; FABIO, 2014) foi desenvolvido em resposta a competição realizada pela *Open Networking Foundation* (ONF), e foi selecionada como a implementação vencedora do prêmio. A competição consistia em desenvolver um *driver* OpenFlow leve e em Linguagem C/C++, que permitisse ser adicionada a aplicações e compiladas.

A arquitetura do Libfluid gerencia a conectividade necessária para implementar o protocolo OpenFlow, e pode ser associado com qualquer mensagem da biblioteca OpenFlow. A arquitetura da biblioteca Libfluid é um cliente-servidor, em que o controlador é o servidor e o *switch* um cliente.

A Figura 6.2 apresenta a arquitetura geral de um sistema que utiliza a biblioteca Libfluid para implementar o controlador (*YourController*). Os clientes, representados pelos *switches* OpenFlow, são designados para múltiplas *threads* gerenciados pelo *libfluid_base*, em que cada uma dessas *threads* será responsável por ler e escrever dados para os *sockets*. Uma vez que os dados são lidos, uma chamada *callback*, definida pela usuário, é ativada. Para lidar com os protocolos específicos de cada mensagem é utilizado o *libfluid_msg*.

Figura 6.2 – Arquitetura de um controlador implementado com Libfluid. Fonte: (ALLAN; CHRISTIAN; FABIO, 2014)



Desta forma, Libfluid corresponde ao Estado da Arte referente a performance, e traz benefícios em tormos de portabilidade para diferentes plataformas e linguagens de programação para diferentes casos de uso, como controladores e *switches*, permitindo integração com aplicações SDN existentes.

6.2 ARQUITETURA PROPOSTA

A arquitetura proposta consiste em um serviço de anonimização para uma Rede Definida por Software, com o propósito de realizar a mitigação da possibilidade de ataques à rede através da ocultação das estações. Este processo de ocultação das estações é realizado com a anonimização dos endereços IP da rede, inclusive da tabela mantida pelo controlador, e desta forma, impedindo que toda a rede seja descoberta em um ataque.

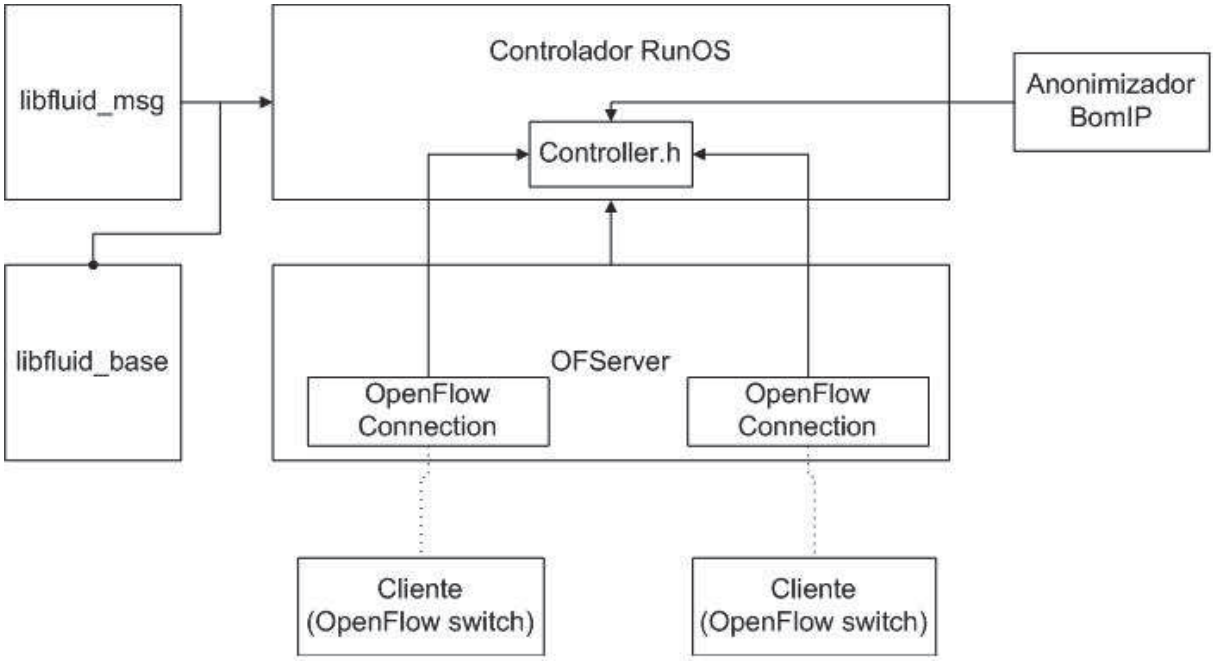
Inicialmente, realizou-se uma pesquisa com os controladores existentes de forma a encontrar um que possuísse melhor desempenho. E assim, permitisse a agregação de serviços desenvolvidos, para desta forma, poder se adicionar um serviço de anonimização dos dados da rede (ALEXANDER et al., 2015). Com base nessa análise, o controlador

selecionado foi o RunOS, por apresentar essas características e ter o código em C/C++ disponível para modificação e compilação, o que permite compilar o controlador já com o serviço desejado, otimizando o funcionamento do mesmo.

Em relação ao anonimizador de dados, neste caso, de endereços IP (origem e destino), foi utilizada a classe desenvolvida, nomeada de BomIP. Esta classe, também desenvolvida em C++, permite a integração por meio de compilação junto ao RunOS, garantindo um melhor desempenho, visto que aplicações compiladas apresentam melhores índices de funcionamento comparadas com as interpretadas (ALEXANDER et al., 2015).

Desta forma, a arquitetura proposta consiste na apresentada na Figura 6.3. Como é possível visualizar, corresponde a arquitetura do RunOS com a adição do BomIP, que é consumida na própria classe do controlador do RunOS, no momento em que um pacote é processado para ser enviado na rede.

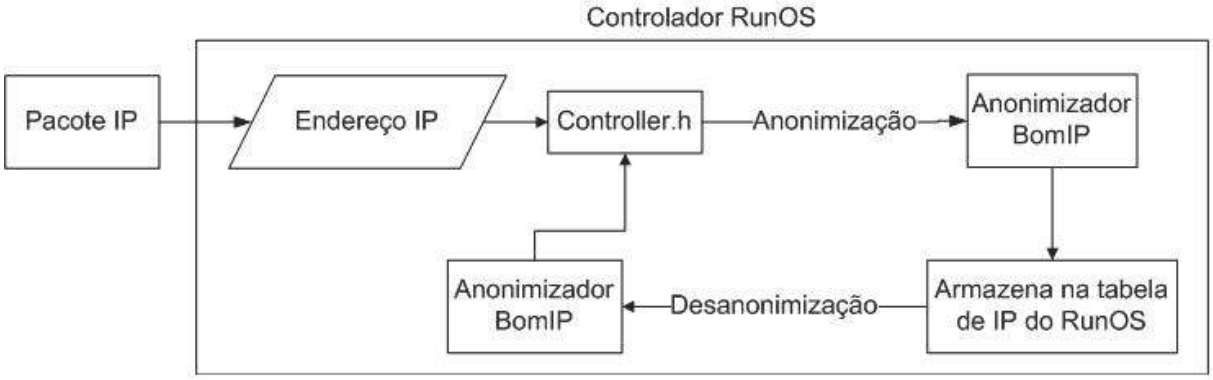
Figura 6.3 – Arquitetura Proposta de um Serviço de Anonimização em SDN



No funcionamento da arquitetura, todo pacote tem seus endereços IP de origem e destino analisados no controlador através do anonimizador BomIP. Na tabela de endereços IP da rede, que fica armazenada no controlador para indicar as rotas que os pacotes devem seguir, os endereços são armazenados anonimizados. Ou seja, quando um novo endereço é adicionado, o mesmo é anonimizado utilizando o BomIP e armazenado.

A Figura 6.4 apresenta o diagrama com o processo de anonimização realizada pelo controlador RunOS com a classe de anonimização BomIP. No momento que um pacote precisa ser roteado através da tabela de endereços IP do controlador, o mesmo faz a desanonimização, ou seja, faz o processo inverso, busca o IP anonimizado armazenado e o converte para o IP real. Este procedimento pode ser realizado graças ao vetor, explicado na seção 5.2, que armazena os dados reais e anonimizados com os possíveis valores de octetos (0 até 255).

Figura 6.4 – Fluxo de anonimização realizada pelo controlador RunOS



A partir desta arquitetura foi desenvolvido uma Rede Definida por Software com um serviço de anonimização de seus endereços IP para prover uma maior segurança, de forma a reduzir as tentativas de ataque sofridas através da descoberta dos endereços IP das estações, reduzindo assim, a quantidade de ataques como Negação de Serviço (DoS e DDoS).

7 TRABALHOS RELACIONADOS

Este capítulo tem o objetivo de apresentar os trabalhos mais relevantes já publicados acerca de anonimização em Redes Definidas por Software. Um comparativo entre os *frameworks* desenvolvidos por cada autor é apresentada ao final do capítulo, demonstrando assim, o que cada serviço tem a oferecer como técnica para ocultação das estações em uma SDN.

7.1 ANONIMIZAÇÃO COM SISTEMA DE ALVO EM MOVIMENTO

Para realizar a reconfiguração de uma rede em tempo real, Chavez, Stout e Peisert (2015) propõem o sistema "Moving Target Defense" (MTD), ou Sistema de Alvo em Movimento, que garante a não interrupção da conectividade entre os nós da rede. O sistema MTD consiste de três técnicas: a randomização de portas TCP/UDP, a randomização de endereços IP, e a randomização dos caminhos de uma rede.

Randomizar os números das portas aumenta a dificuldade para que um usuário malicioso consiga descobrir os serviços que estão em execução na rede. As portas de serviços conhecidos estão, tipicamente em valores abaixo de 1024, e são definidas em arquivos do sistema UNIX. Um dos primeiros passos, para causar uma adversidade na fase de reconhecimento de um ataque, é dificultar a descoberta dos sistema que estão em execução. Porque com conhecimento dos sistemas em execução, o usuário malicioso consegue atacar as vulnerabilidades conhecidas. Porém, a randomização da porta força o usuário a tentar realizar a engenharia reversa, para aprender as portas reais que estão sendo utilizadas.

Em relação a randomização de endereços IP, (CHAVEZ; STOUT; PEISERT, 2015) utilizam um controlador SDN que gerencia a comunicação através dos dispositivos pelo endereço IP. Quando o tráfego ingressa na camada de rede do dispositivo, o par remetente-destino é validado e as regras instaladas para codificar/decodificar os endereços IP randomizados. O objetivo de randomizar os endereços IP consiste em enganar o usuário malicioso, mitigando-o a atacar endereços que não correspondem aos reais da rede.

A randomização de endereços IP ainda permite que a rede esteja suscetível a análise de tráfego através de ferramentas do tipo *sniffer*. Para impedir o monitoramento da comunicação dos pacotes trafegados na rede, os autores propõem a randomização dos caminhos. Para cada fluxo da rede, o controlador designa um caminho randomizado entre os nós da rede que estão se comunicando.

Ainda no mesmo trabalho, os resultados dos experimentos indicam que a randomização de porta foi o que menos impactou no desempenho da comunicação da rede. O processo de anonimização mantém a conectividade da sessão, com eficiência ao frustrar ataques à rede, porém com perda da informação dos tipos de protocolo que estão trafegando na rede. Em relação a randomização de endereços IP, o processo de anonimização também manteve a conectividade da sessão, porém com um impacto no desempenho da rede maior que a randomização da porta. E, a randomização dos caminhos conseguiu evitar a rastreabilidade da rede, porém, com um alto custo de desempenho, devendo ter atenção para uso em sistemas críticos, que necessitam de resultados em tempo hábil.

O trabalho de (ZHAO; GUO; LIU, 2015) concentra-se no campo referente a variação do endereço IP, de forma a modificá-lo antes que ele entre em uma rede pública, prevenindo que o mesmo possa ser identificado. Para esta finalidade, os autores desenvolveram três módulos para realizar a randomização de endereços IP, gerenciados pelo controlador SDN. Os módulos consistem em: um módulo de conversão entre endereços IP virtuais e reais, um módulo de autenticação e um gerador de endereços IP virtuais.

O módulo principal é chamado de vIP, neste módulo o controlador é encarregado de gerenciar todos os *switches* de um domínio, e cada *switch* está conectado ao controlador através de um canal seguro. Assim, o controlador cria uma tabela contendo a relação entre os endereços IP virtuais e os reais da rede para conversão. O módulo de autenticação é desenvolvido para prover controle de acesso aos dispositivos da rede que acessarem a Internet, e o módulo de geração de endereços IP virtuais é responsável por gerar endereços IP randômicos, que serão utilizados no mapeamento entre os IP virtuais e reais.

Os resultados obtidos com experimentos indicaram que o trabalho de (ZHAO; GUO; LIU, 2015) possui uma taxa de 95% de eficiência, impedindo assim, que a maioria dos endereços IP reais fossem descobertos durante as tentativas simuladas de descoberta de endereços.

OF-RHM (OpenFlow Random Host Mutation) (JAFARIAN; AL-SHAER; DUAN, 2012) é um mecanismo para realizar "mutação" no endereço IP de uma rede randomicamente e frequentemente de forma transparente. Neste caso, o controlador designa a cada estação um IP virtual temporário, que é traduzido "de" e "para" em relação aos endereços IP reais das estações. Os resultados teóricos indicaram que OF-RHM invalidou 99% da tentativa de obtenção de informação através de escaneamento remoto.

7.2 ANONIMIZAÇÃO POR RANDOMIZAÇÃO

Com o objetivo de realizar uma randomização para aumentar a detecção de ataques enquanto os mesmos ocorrem, em (JAFARIAN; AL-SHAER, 2015) é introduzido o conceito de "Defesa Proativa contra Adversário Consciente". Esse mecanismo tem como alvo estabelecer, estrategicamente, um dinamismo em sistemas estáticos para caracterizar o comportamento do adversário. A abordagem realiza a anonimização do endereço IP das estações frequentemente, através da geração de novos identificadores aleatórios, de forma a não permitir a rastreabilidade do reconhecimento da rede para ataques. A distribuição baseada em quais endereços são designados para cada estação da rede, bem como a taxa de randomização é adaptativa e determinada pela verificação do endereço que está sob ataque.

De acordo com os autores, a adaptação para detectar um ataque é rápida e apurada, assim, a abordagem é capaz de reagir e evacuar as estações da rede mediante uma ameaça de ataque. No entanto, alterar o endereço IP de uma estação não pode ser realizado com frequência, devido a interrupção das sessões ativas e afetar a performance. Para manter o endereço IP das estações finais, Jafarian e Al Shaer (2015) utilizavam um endereço IP efêmero, de vida curta, que é escolhido a partir de um conjunto de endereços internos não utilizados da rede. Estes endereços IP efêmeros não são utilizados no roteamento, e são automaticamente traduzidos para os reais e vice versa. Assim, esta transparência permite randomizar os endereços IP da rede sem afetar as sessões que estão estabelecidas.

A eficiência do conceito é verificada através de várias tentativas de escaneamento da rede. Experimentos de simulação realizados mostraram que o reconhecimento adaptativo, configurado como uma estratégia defensiva, conseguiu conter completamente a propagação de *worms* na rede. Estes *worms* são utilizados com o objetivo de espalharem-se na rede, obterem e divulgarem vulnerabilidades para que ataques possam ser realizados.

AnonyFlow (MENDONCA; SEETHARAMAN; OBRACZKA, 2012) é uma solução em que usuários da Internet utilizam um endereço de IP temporário e descartado por fluxo. Esta abordagem é similar ao conceito de *Network Address Translation* (NAT): terceiros não tem visibilidade sobre a rede interna, e assim, não conseguem correlacionar o tráfego com um endereço IP. Os autores apresentam essa característica como um serviço adicional que pode ser escolhido, similar ao ocultamento de números em chamadas telefônicas.

Tentativas anteriores de anonimização necessitavam de *gateways* especiais para rastrear o estado do usuário e do fluxo do tráfego. Com o paradigma SDN isto fica simplificado, porque tudo é gerenciado pelo controlador, que implementa a customização de políticas de roteamento entre múltiplos *switches*. Assim, a função de anonimização é realizada pelos *switches* com grande velocidade. Testes realizados pelos autores em *switches* OpenFlow comerciais revelaram que AnonyFlow causa uma perda insignificante de vazão.

Os trabalhos analisados nesta seção podem ser classificados de acordo com a técnica de anonimização adotada conforme apresentado na Tabela 7.1.

As colunas Portas, IP e Fluxos da Tabela 7.1, destacam quais itens foram anonimizados em cada trabalho. Devido a todos os trabalhos manterem a conexão ativa dos fluxos quando a anonimização é realizada, a informação não foi apresentada. Em relação a análise de desempenho da rede com o uso dos serviços desenvolvidos, apenas dois trabalhos realizam a verificação: (CHAVEZ; STOUT; PEISERT, 2015) e (MENDONCA; SEETHARAMAN; OBRACZKA, 2012).

Nestes trabalhos, a anonimização dos identificadores ocorre de forma randômica, sem preservação do prefixo dos endereços IP, o que não permite que uma análise de tráfego seja realizada na rede, uma vez que os identificadores não mantém características como a máscara da rede. Por isto, é possível visualizar também na Tabela 7.1 a lacuna em relação a permitir uma análise do tráfego após a anonimização ter sido realizada na rede.

Diferente de outros modelos propostos, o BomIP oferece anonimização em tempo real dos endereços IP dos pacotes trafegados na rede, realizando uma anonimização por meio de randomização, porém, agregando a técnica de preservação de prefixo, para oferecer

um serviço de segurança provendo privacidade dos *hosts* e outros serviços da rede, e permitindo a rastreabilidade do tráfego para análise e verificação de possíveis ataques.

Tabela 7.1 – Comparação dos Trabalhos Relacionados

Autores	Técnica	Portas	IP	Fluxo	Análise Desempenho	Permite Análise Tráfego	Novo IP por Fluxo
(JAFARIAN; AL-SHAER, 2015)	Anonimização gerando valores aleatórios		X				
(CHAVEZ; STOUT; PEISERT, 2015)	Anonimização com Sistema de Alvo em Movimento	X	X	X	X		
(ZHAO; GUO; LIU, 2015)	Anonimização com Sistema de Alvo em Movimento		X				
(MENDONCA; SEETHARAMAN; OBRACZKA, 2012)	Anonimização gerando valores aleatórios		X		X		X
(JAFARIAN; AL-SHAER; DUAN, 2012)	Anonimização com Sistema de Alvo em Movimento		X				X
Serviço BomIP	Anonimização Aleatória com preservação de prefixo		X			X	

8 ESTUDO DE CASO

Neste capítulo são apresentados os experimentos realizados para validação da arquitetura proposta. Para cada experimento, há a respectiva análise dos resultados.

O Estudo de Caso I consiste em apresentar a rastreabilidade do tráfego dos dados anonimizados, o que só é possível devido a utilização da técnica de preservação de prefixo. Para apresentar este resultado, os *traces* anonimizados com BomIP são comparados com *traces* anonimizados por outra ferramenta e com os *traces* originais.

O Estudo de Caso II apresenta a mitigação de ataques ocorrida em uma SDN utilizando o serviço BomIP, comparando com um mesmo cenário que não utiliza o serviço.

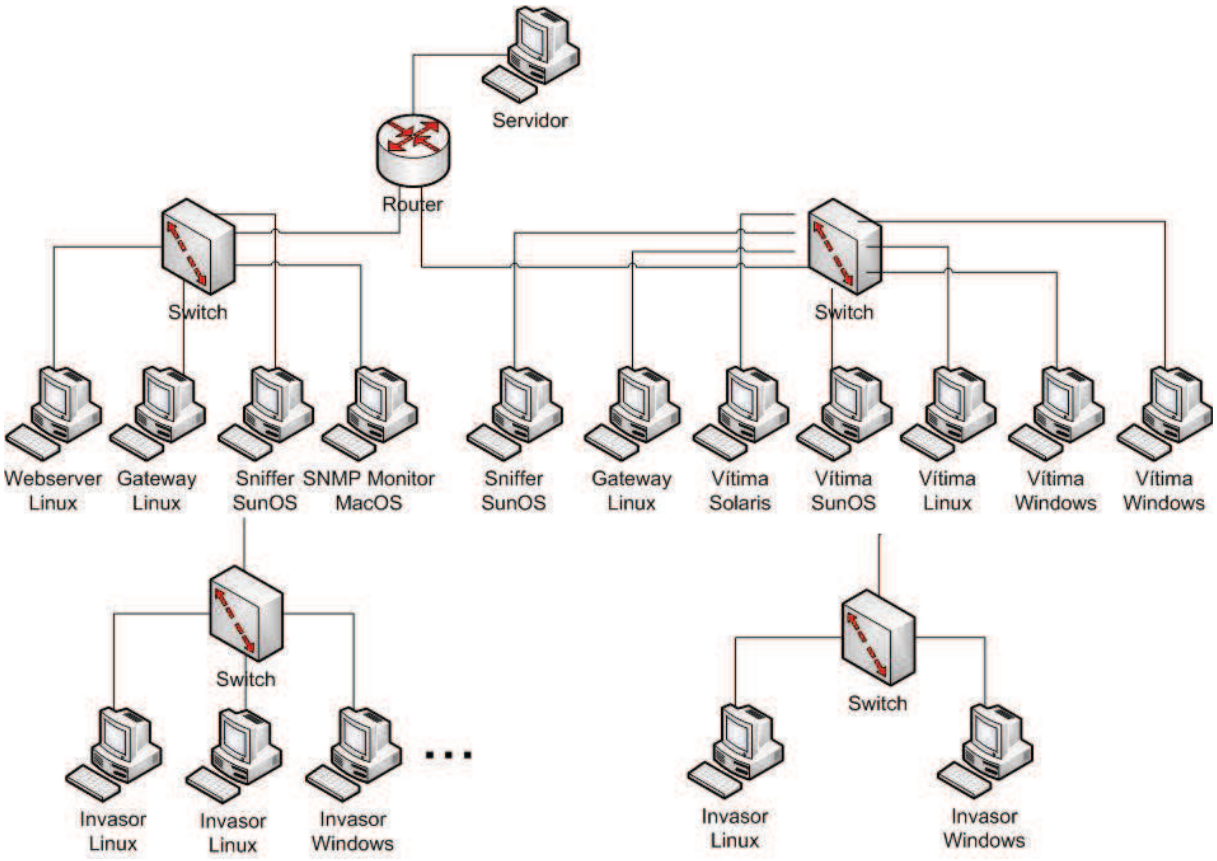
8.1 CENÁRIO

Para realizar a Análise do Algoritmo e o Estudo de Caso I foram utilizados os *traces* disponibilizados pelo *Lincoln Laboratory do Massachusetts Institute of Technology* (MIT). Estes *traces* foram criados em laboratório pelo grupo de pesquisa *Intrusion Detection Evaluation Group*, denominado DARPA (RICHARD et al., 2000).

A Figura 8.1 apresenta a arquitetura da rede desenvolvida para simulação dos ataques, e desta forma, geração dos *traces* para torná-los disponíveis publicamente para pesquisas na área de segurança de redes.

Baseado no cenário apresentado na Figura 8.1, foram realizadas simulações de ataques a rede como, *rootkits*, *Denied of Service* (DoS), *buffer overflow*, tentativas de descoberta de senha via Telnet, tentativa de mapeamento da rede usando NMAP, e a

Figura 8.1 – Rede de Simulação utilizada para geração dos traces pelo MIT.



simulação da invasão de um *host* para que o administrador possa verificar o tráfego da rede.

A simulação coletou os dados durante 22 horas. O programa TCPSlice (SONY, 2000) foi usado para examinar os dados e gerar os arquivos referente a cada um dos dias de simulação.

Cada um dos *hosts* disponibilizados possuem o seu endereço IP publicado, que servirão para avaliar a anonimização realizada pela ferramenta BomIP. As Tabelas 8.1 e 8.2 apresentam os endereços IP de cada um dos *hosts* e equipamentos usados na simulação.

Estas simulações ocorreram durante a segunda semana de março do ano de 1999, sendo gerado um arquivo para cada dia da semana entre segunda e sexta-feira. A Tabela 8.3 apresenta os horários, de cada um dos arquivos gerados, com a indicação do primeiro e último pacote capturado.

O Estudo de Caso II demonstra a mitigação dos ataques de Negação de Serviços, sofridos por uma SDN, em um ambiente simulado. Os resultados deste ataque são analisados com uma mesma rede sem o uso de serviço de anonimização, e desta forma, é possível

Tabela 8.1 – Endereços IP dos hosts na rede local. Fonte: (RICHARD et al., 2000).

Endereço IP	*Nome do Host*	*Sistema Operacional*
172.16.12.10	plato.eyrie.af.mil	Solaris
172.16.112.10	locke.eyrie.af.mil	SunOS
172.16.112.20	hobbes.eyrie.af.mil	Redhat 5.0
172.16.112.50	pascal.eyrie.af.mil	Solaris 2.5.1
172.16.112.100	hume.eyrie.af.mil	Windows NT 4.0
172.16.112.149	eagle.eyrie.af.mil	Redhat 5.0
172.16.112.194	falcon.eyrie.af.mil	Solaris 2.5.1
172.16.112.207	robin.eyrie.af.mil	SunOS 4.1.4
172.16.113.50	zeno.eyrie.af.mil	SunOS 4.1.4
172.16.113.84	duck.eyrie.af.mil	SunOS 4.1.4
172.16.113.105	swallow.eyrie.af.mil	Redhat 5.0
172.16.114.168	finch.eyrie.af.mil	SunOS 4.1.4
172.16.114.169	swan.eyrie.af.mil	Solaris 2.5.1
172.16.114.207	pigeon.eyrie.af.mil	Redhat 5.0
172.16.115.5	pc1.eyrie.af.mil	Windows 95
172.16.115.87	pc2.eyrie.af.mil	Windows 95
172.16.115.234	pc0.eyrie.af.mil	Window NT 4.0
172.16.116.44	pc5.eyrie.af.mil	Windows 3.1
172.16.116.194	pc3.eyrie.af.mil	Windows 95
172.16.116.201	pc4.eyrie.af.mil	Windows 95
172.16.117.52	pc7.eyrie.af.mil	Windows 3.1
172.16.117.103	pc9.eyrie.af.mil	MacOS
172.16.117.111	pc8.eyrie.af.mil	MacOS
172.16.117.132	pc6.eyrie.af.mil	Windows 3.1
172.16.118.10	linux1.eyrie.af.mil	Redhat 5.0
172.16.118.20	linux2.eyrie.af.mil	Redhat 5.0
172.16.118.30	linux3.eyrie.af.mil	Redhat 5.0
172.16.118.40	linux4.eyrie.af.mil	Redhat 5.0
172.16.118.50	linux5.eyrie.af.mil	Redhat 5.0
172.16.118.60	linux6.eyrie.af.mil	Redhat 5.0
172.16.118.70	linux7.eyrie.af.mil	Redhat 5.0
172.16.118.80	linux8.eyrie.af.mil	Redhat 5.0
172.16.118.90	linux9.eyrie.af.mil	Redhat 5.0
172.16.118.100	linux10.eyrie.af.mil	Redhat 5.0

comparar o percentual de estações atacadas em cada uma das redes durante um período de tempo, sendo possível verificar a mitigação em relação à rede com uso do serviço de anonimização.

Tabela 8.2 – Endereços IP dos hosts externos à rede local. Fonte: (RICHARD et al., 2000).

Endereço IP	*Nome do Host*	*Sistema Operacional*
135.13.216.191	alpha.apple.edu	Redhat 5.0
135.8.60.182	beta.banana.edu	Solaris 2.5.1
194.27.251.21	gamma.grape.mil	SunOS 4.1.4
194.7.248.153	delta.peach.mil	Redhat 5.0
195.115.218.108	epsilon.pear.com	Solaris 2.5.1
195.73.151.50	lambda.orange.com	SunOS 4.1.4
196.37.75.158	jupiter.cherry.org	Redhat 5.0
196.227.33.189	saturn.kiwi.org	Solaris 2.5.1
197.182.91.233	mars.avocado.net	SunOS 4.1.4
197.218.177.69	pluto.plum.net	Redhat 5.0
192.168.1.30	monitor	MacOS
192.168.1.10	calvin.world.net	
192.168.1.20	aesop.world.net	
192.168.1.90	solomon.world.net	

Tabela 8.3 – Horários de coleta de pacotes durante a simulação

Primeiro Pacote			**Último Pacote**		
Segunda	08/03/1999	08:00:01	Terça	09/03/1999	06:00:49
Terça	09/03/1999	08:00:01	Quarta	10/03/1999	02:59:59
Quarta	10/03/1999	08:00:03	Quinta	11/03/1999	06:00:01
Quinta	11/03/1999	08:00:03	Sexta	12/03/1999	06:00:00
Sexta	12/03/1999	08:00:02	Sábado	13/03/1999	06:00:00

8.2 ANÁLISE DO ALGORITMO DO BOMIP

A análise do algoritmo consiste em comparar o tempo de execução dos algoritmos de anonimização do BomIP e do Crypto-Pan (JINLIANG; JUN; MOSTAFA, 2004). Esta comparação é realizada através de análise assintótica utilizando a notação Big-O, com expansão de somatório através de indução matemática (MANBER, 1989).

Para realizar a anonimização, o Crypto-Pan utiliza o algoritmo de cifra de Rijndael (JOAN; VINCENT, 1998). Este corresponde a um algoritmo de cifra de chave simétrica de 128 ou 256 bits, em que a única forma conhecida para quebrar o segredo é através de algoritmos de força bruta (HARSH; RAVINDRA, 2012), o que garante uma maior segurança e confiabilidade a uma informação anonimizada pelo Crypto-Pan.

Desta forma, o Crypto-Pan foi selecionado para ser comparado com o BomIP devido ao fato de ambos utilizarem a técnica de preservação de prefixo. Porém, o Crypto-Pan baseia-se no uso de criptografia para realizar a anonimização, enquanto que o BomIP utiliza a técnica de randomização.

A partir da análise realizada por (CRISTOPHER et al., 2008) é constatado que o algoritmo de Rijndael, utilizado no Crypto-Pan, tem uma complexidade algorítmica de $O(n^4)$. Isto ocorre devido ao processo de criptografia dos valores fornecidos, que é representado conforme Algoritmo 8.1.

Algoritmo 8.1 Pseudocódigo do algoritmo de cifra de Rijndael

```
1: for i = numBlocks; i > 0; i − − do
2:     for k = 0; k < 128; k + + do
3:         encrypt(block, block)
```

De acordo com o trecho do pseudocódigo de cifra de Rijndael, é possível verificar a presença de dois laços de repetição aninhados, em que é chamada a função de criptografia. Esta função de criptografia faz um caminhamento em uma matriz.

Com base em (MANBER, 1989), sabe-se que a análise assintótica de um algoritmo com a presença de dois laços de repetição aninhados é representado através do somatório $\sum_{i=1}^{n} i$, enquanto que percorrer uma matriz também é representado por $\sum_{i=1}^{n} i$.

Assim, a equação para resolução da complexidade assintótica do algoritmo de Rijndael, no pior caso, fica $\sum_{i=1}^{n} i * \sum_{i=1}^{n} i$.

Esta equação pode ser resolvida através de indução matemática. Sabendo que o somatório de i, em que os valores são somados de 1 até n, corresponde a soma de uma sequência de uma Progressão Matemática (PA) de razão 1, ficando conforme representado por $\sum_{i=1}^{n} i = 1 + 2 + 3 + 4 + ... + n = \frac{n(n+1)}{2}$.

Logo, $\sum_{i=1}^{n} i * \sum_{i=1}^{n} i = \frac{n(n+1)}{2} * \frac{n(n+1)}{2} = \frac{n^4}{4} + \frac{2n^2}{4} + \frac{n^2}{4}$.

O que resulta em um algoritmo de complexidade $O(n^4)$.

O algoritmo do BomIP, realiza a anonimização dos valores através das operações apresentadas no Algoritmo 8.2.

Algoritmo 8.2 Bloco de anonimização do BomIP

```
1: for todosospacotes do
2:     ipOrigem = separaIP(ipOrigemPacote)
3:     ipDestino = separaIP(ipOrigemPacote)
4:     ipFinalOrigem = anonimizaIP(ipOrigem)
5:     ipFinalDestino = anonimizaIP(ipOrigem)
```

De acordo com o Algoritmo 8.2 é possível visualizar que a etapa de anonimização do BomIP, no pior caso, analisa cada pacote para processar os endereços IP de origem e destino. Porém, é necessário validar se o valor anonimizado já não existe, e com isto percorrer

o vetor, o que, no pior caso, pode ter que percorrer as n posições existentes. Segundo (MANBER, 1989), os processos de atribuição de valor, atribuição de valor aleatório e acesso direto a posição, são etapas de tempo constante, porém, percorrer um vetor de tamanho n, no pior caso, custa tempo n.

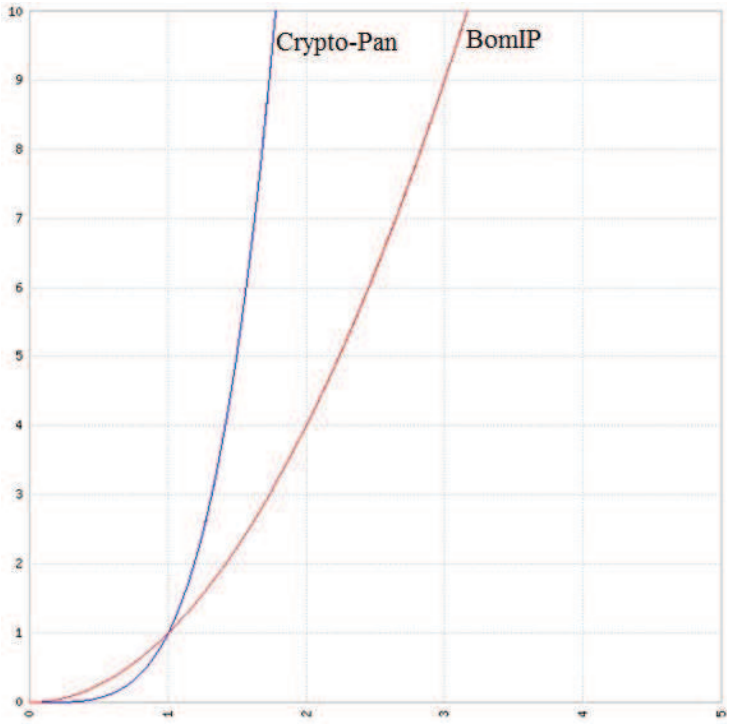
Assim, a análise assintótica é expressa pelo seguinte somatório $\sum_{i=1}^{n} i$, com o resultado da expansão deste somatório sendo o próprio n^2, conforme a expansão $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$.

Logo, o algoritmo é executado em tempo O(n^2).

8.2.1 DISCUSSÃO DA ANÁLISE DO ALGORITMO DO BOMIP

A Figura 8.2 apresenta a comparação entre as funções assintóticas dos dois algoritmos, em que é possível identificar o crescimento acelerado do Crypto-Pan (curva em azul) em relação ao BomIP (curva em vermelho). O eixo horizontal representa os valores da variável das funções de quarta ordem do Crypto-Pan e de segunda ordem do BomIP, enquanto que o eixo vertical corresponde aos resultados das funções.

Figura 8.2 – Gráfico com complexidade assintótica do Crypto-Pan e BomIP



Em relação ao espaço utilizado, BomIP armazena os dados anonimizados, ao realizar a divisão dos octetos de um endereço IP, em um vetor de 256 posições. Enquanto que o

algoritmo de Rijndael possui uma complexidade espacial de uma matriz de 256 linhas por 4 colunas (CRISTOPHER et al., 2008). Desta forma, em comparação, o BomIP apresenta um ganho de espaço no processamento, por não trabalhar com blocos de cifras igual ao Crypto-Pan.

Utilizando a base de dados do MIT, com os resultados da segunda semana de simulação do ano de 1999, os endereços IP dos pacotes capturados nesta simulação foram anonimizados em ambas as ferramentas, BomIP e Crypto-Pan, de forma a validar os resultados de complexidade obtidos através de um processamento real.

Os resultados obtidos no processamento estão apresentados nas Figuras 8.3, 8.4, 8.5, 8.6 e 8.7, em que a curva em azul de cada gráfico representa o Crypto-pan, enquanto que a curva em vermelho corresponde ao tempo do BomIP. É possível visualizar um maior tempo de processamento no processo de anonimização realizado pelo Crypto-Pan em relação ao BomIP, o que confirma o melhor desempenho na anonimização. Nestas figuras, o eixo X representa a quantidade de repetições de anonimizações dos *traces* na respectiva ferramenta, enquanto que o eixo Y representa o tempo, em segundos, gasto para realizar o processamento pela ferramenta.

Figura 8.3 – Tempo de processamento em segundos de amostras da segunda-feira

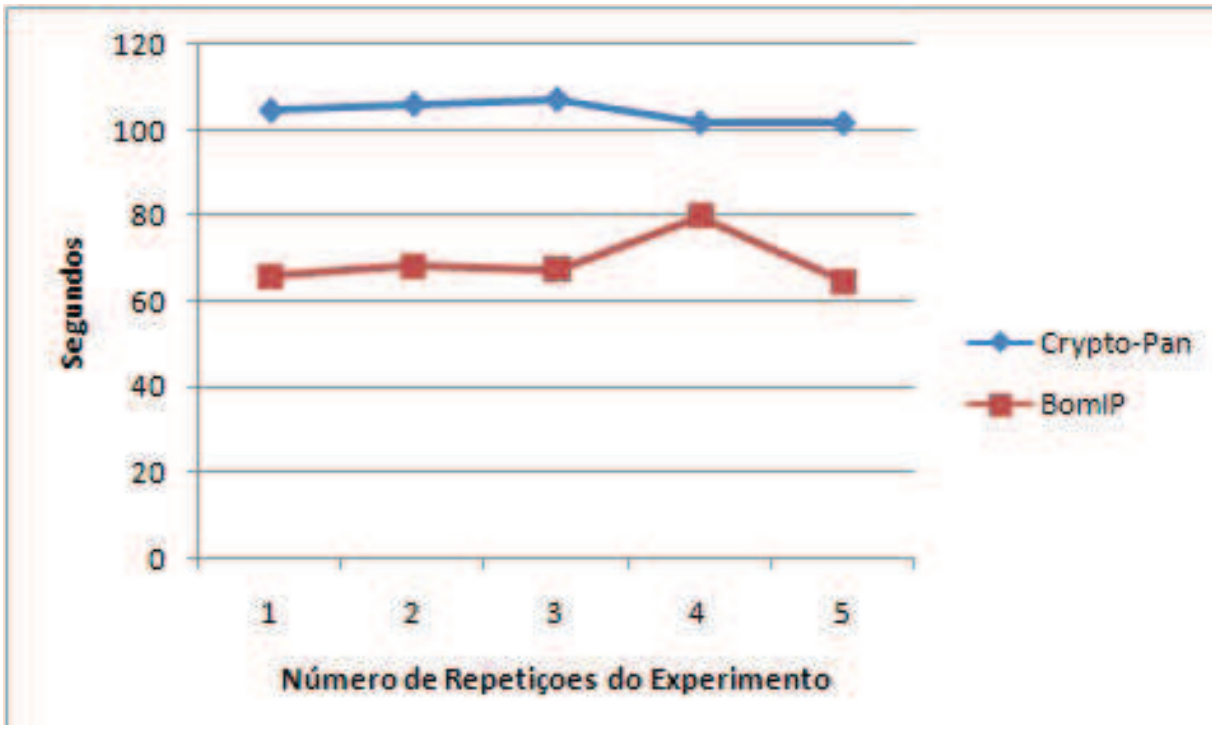


Figura 8.4 – Tempo de processamento em segundos de amostras da terça-feira

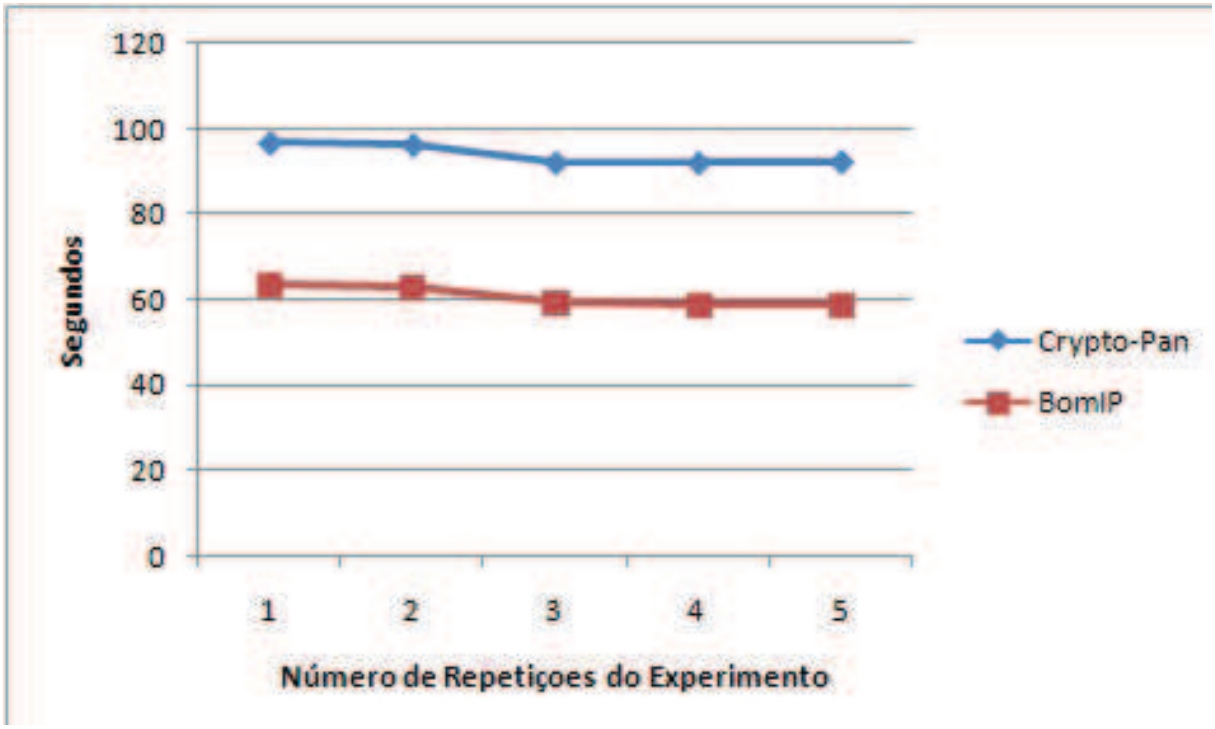
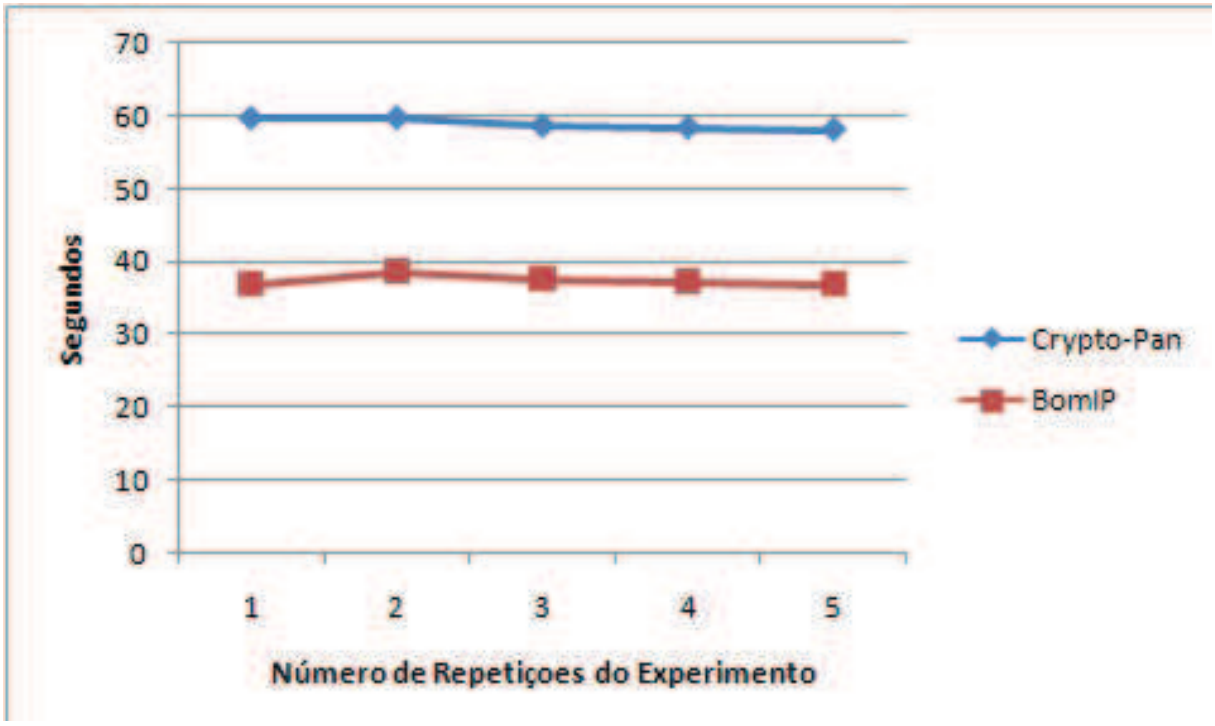


Figura 8.5 – Tempo de processamento em segundos de amostras da quarta-feira



Com base na análise dos gráficos, é possível verificar que o ganho de processamento do BomIP em relação ao Crypto-Pan corresponde a 64,62% (segunda-feira: 66,27%, terça-feira: 64,56%, quarta-feira: 63,49%, quinta-feira: 64,97%, sexta-feira: 63,81%), o que

Figura 8.6 – Tempo de processamento em segundos de amostras da quinta-feira

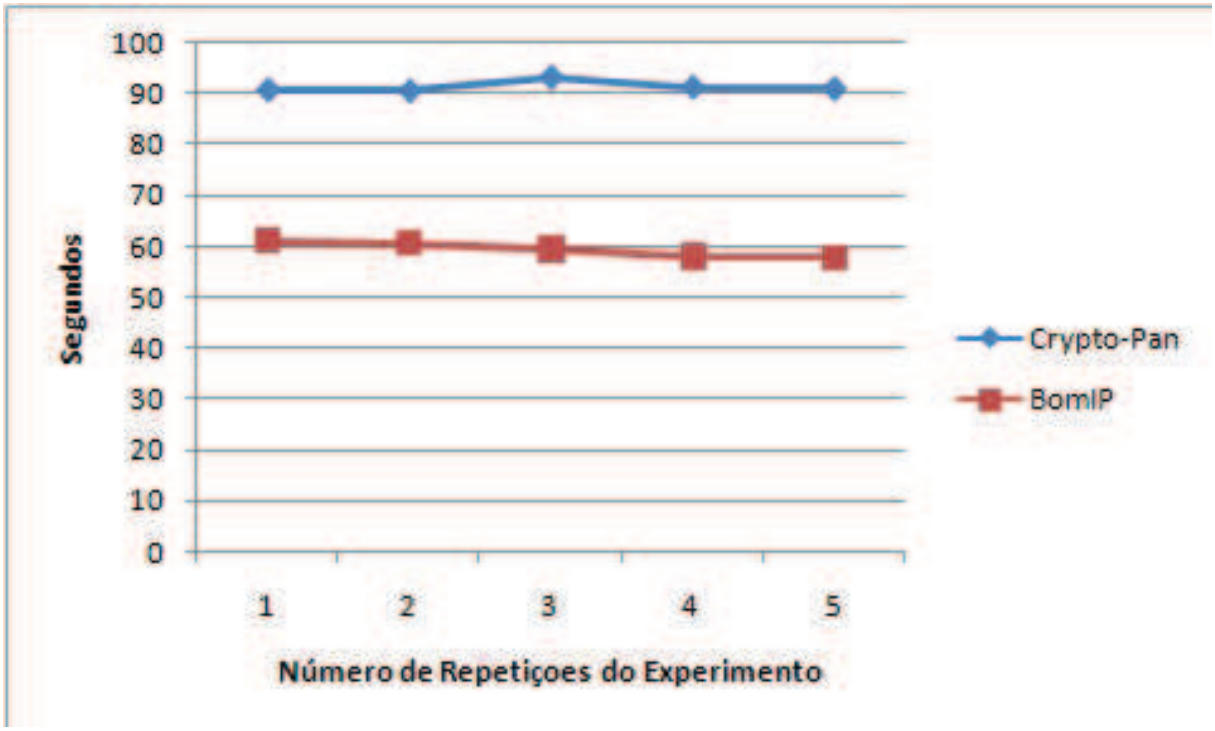
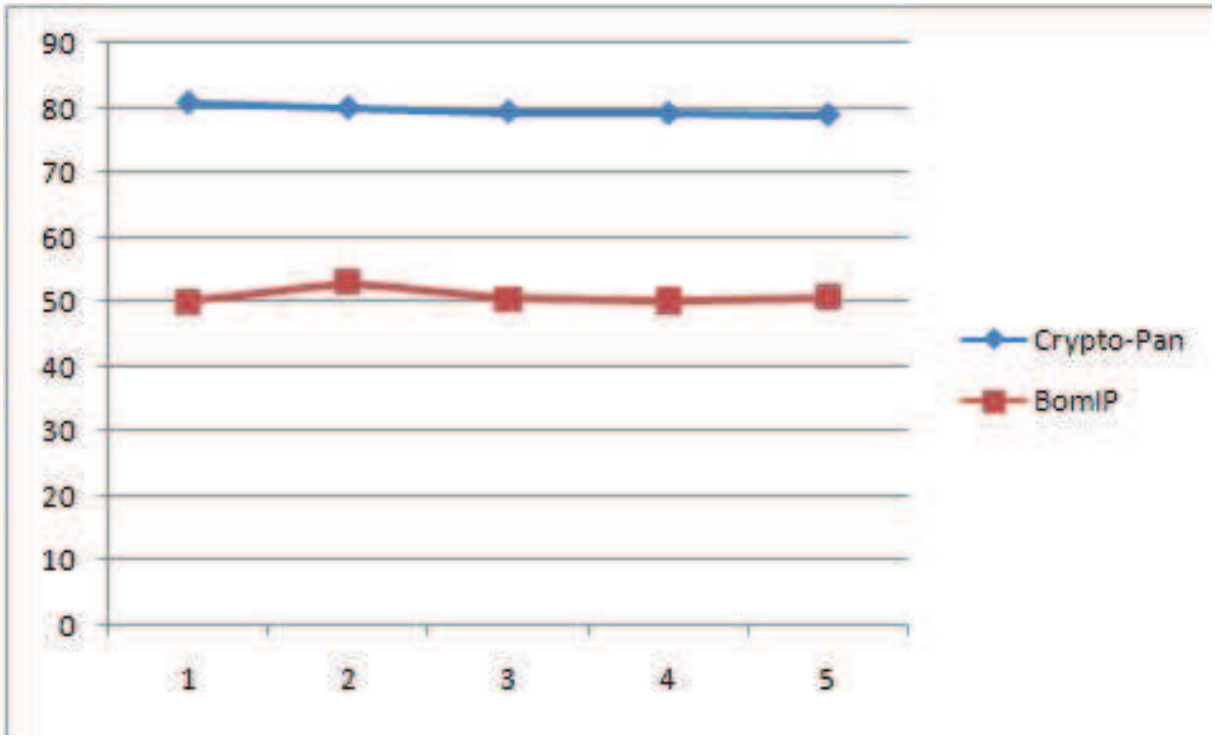


Figura 8.7 – Tempo de processamento em segundos de amostras da sexta-feira



corrobora com a análise assintótica realizada de acordo com os algoritmos de anonimização do Crypro-Pan e BomIP.

8.3 ESTUDO DE CASO I

O primeiro Estudo de Caso, baseado no trabalho de (KIRAN; ADAM, 2008), consiste em anonimizar o *trace* original, e realizar a análise na ferramenta de *Intrusion Detection System* (IDS) SNORT (MARTIN; CHRIS, 2003), e verificar desta forma a rastreabilidade dos *traces* e identificação de falsos-positivos e falsos-negativos, que são apresentados quando comparados os dados anonimizados em relação com os originais.

Em (KIRAN; ADAM, 2008) é apenas comparado o *trace* original com a ferramenta desenvolvida. Para validar o BomIP, foram comparados os resultados do *trace* original, com os *traces* originais anonimizados com o Crypto-Pan (JINLIANG; JUN; MOSTAFA, 2004), e os *traces* originais anonimizados com o BomIP.

A Figura 8.8 apresenta o fluxograma do experimento realizado. Inicialmente foram obtidos os *traces* correspondentes a segunda semana de medição disponibilizada pelo grupo DARPA (RICHARD et al., 2000) do MIT.

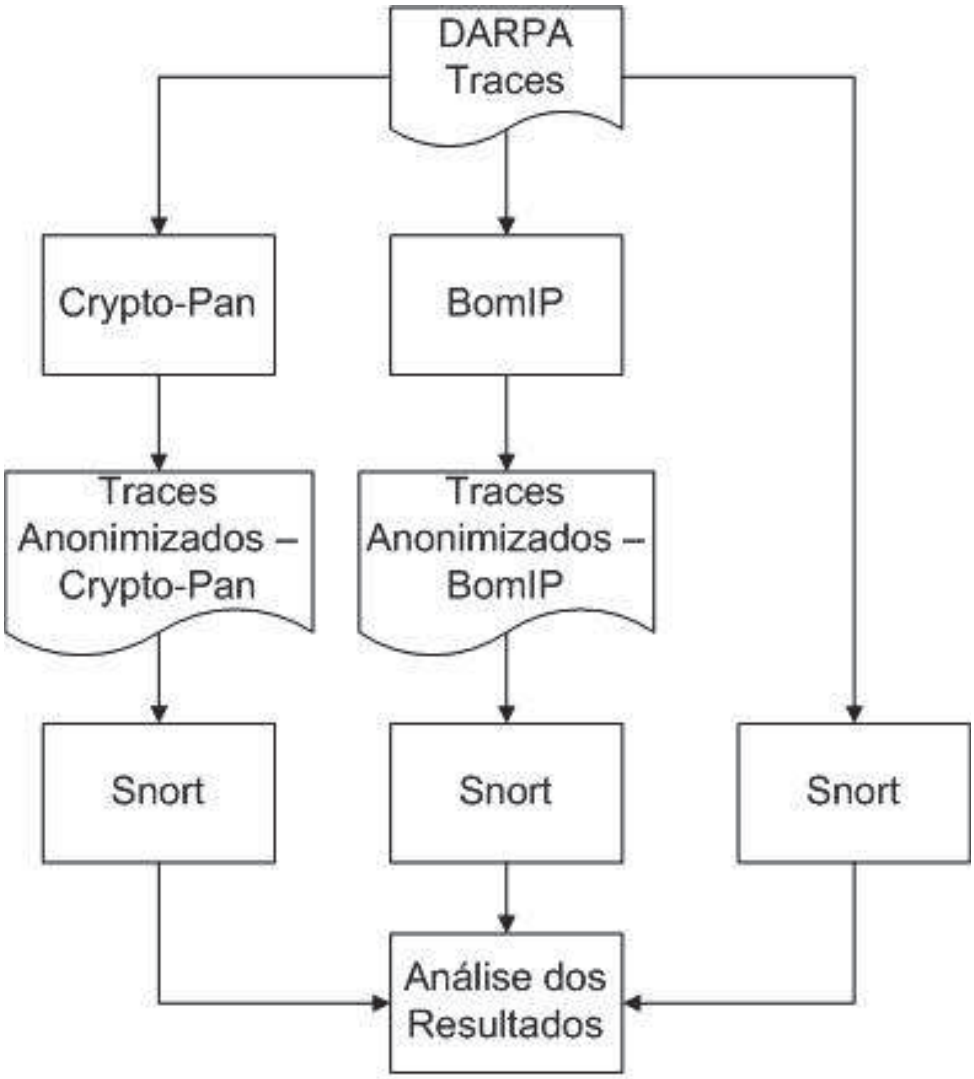
Em seguida, estes traces originais foram anonimizados pelo Crypto-Pan, gerando um novo conjunto de traces; e também os traces originais foram anonimizados com o BomIP, ocasionando um novo conjunto de traces anonimizados.

Por fim, os três conjuntos de traces foram submetidos ao SNORT e avaliados os resultados da seguinte forma:

- Positivos: número correto de alertas gerados.
- Falso-positivos: número de alertas gerados nos *traces* anonimizados, mas não gerados no trace original.
- Falso-negativos: número de alertas que não foram gerados nos traces anonimizados.
- Quantidade de pacotes anonimizados.
- Número de conexões TCP, UDP, SMTP.
- Número de alertas gerados.

Estes alertas, gerados pelo SNORT, são comparados com os valores originais não anonimizados, para desta forma, validar a rastreabilidade dos *traces* anonimizados. Neste caso, o SNORT é capaz de identificar as tentativas de ataque de um endereço IP anonimizado pela manutenção das rastreabilidade, ou seja, pelo quantitativo de vezes que o endereço aparece em um dado intervalo de tempo, o que é aferido através das regras padrões da ferramenta SNORT para identificação de ataques.

Figura 8.8 – Fluxograma do experimento realizado.



8.3.1 ANÁLISE DO ESTUDO DE CASO I

Os resultados estão apresentados nas Tabelas 8.4, 8.5, 8.6, 8.7 e 8.8, contendo os valores obtidos a partir da análise de cada um dos LOGs gerados pelo SNORT para cada *trace* original e anonimizado.

De acordo com os dados extraídos dos LOGs, é possível visualizar que a quantidade de Falso-Positivos gerados são semelhantes em ambas as ferramentas, com uma variação aproximada de 1%. Os Falso-Negativos, quando ocorreram, tiveram a mesma incidência em ambas as ferramentas, e foram causadas por ataques UDP em tentativas de invasão ao servidor de E-Mail, e também em envios de pacotes ICMP sucessivos, constantes em tentativas de ataques DoS.

Tabela 8.4 – Dados estatísticos referentes aos *traces* DARPA de segunda-feira

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.753.377	1.738.745	1.738.745
Positivos	64.651	64.649	64.649
Falso-Positivos	–	2.329	2.330
Falso-Negativos	–	2	2
Nº Pacotes Anonimizados	–	1.738.745	1.738.745
Nº conexões TCP	44.491	44.491	44.491
Nº conexões UDP	95.136	95.236	95.029
Nº conexões SMTP	2.684	2.684	2.684
Nº total de sessões	139.627	139.727	139.520
Nº de sessões TCP rastreadas	1.336.660	1.364.559	1.364.559
Nº de sessões UDP rastreadas	95.136	95.236	95.029

Tabela 8.5 – Dados estatísticos referentes aos *traces* DARPA de terça-feira

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.585.120	1.572.881	1.572.881
Positivos	52.903	52.903	52.903
Falso-Positivos	–	2.503	2.508
Falso-Negativos	–	0	0
Nº Pacotes Anonimizados	–	1.572.881	1.572.881
Nº conexões TCP	55.503	55.503	55.503
Nº conexões UDP	22.821	22.927	22.927
Nº conexões SMTP	3.371	3.371	3.371
Nº total de sessões	78.324	78.430	78.430
Nº de sessões TCP rastreadas	1.456.213	1.482.408	1.482.408
Nº de sessões UDP rastreadas	22.821	22.927	22.927

Tabela 8.6 – Dados estatísticos referentes aos *traces* DARPA de quarta-feira

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.011.149	996.547	996.547
Positivos	32.347	32.347	32.347
Falso-Positivos	–	1.732	1.750
Falso-Negativos	–	0	0
Nº Pacotes Anonimizados	–	996.547	996.547
Nº conexões TCP	26.219	26.219	26.219
Nº conexões UDP	23.202	23.200	23.286
Nº conexões SMTP	2.890	2.890	2.890
Nº total de sessões	49.421	49.419	49.505
Nº de sessões TCP rastreadas	898.632	922.164	922.164
Nº de sessões UDP rastreadas	23.202	23.200	23.286

A quantidade menor de pacotes anonimizados em relação ao número total de pacotes originais ocorre porque a anonimização é realizada apenas em pacotes de protocolo IPv4, descartando-se os que utilizam IPv6. Estes descartes geram algumas divergências

Tabela 8.7 – Dados estatísticos referentes aos *traces* DARPA de quinta-feira

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.563.069	1.548.993	1.548.993
Positivos	48.719	48.715	48.715
Falso-Positivos	–	1.977	1.962
Falso-Negativos	–	4	4
Nº Pacotes Anonimizados	–	1.548.993	1.548.993
Nº conexões TCP	68.121	68.121	68.121
Nº conexões UDP	17.027	17.027	16.914
Nº conexões SMTP	3.229	3.229	3.229
Nº total de sessões	85.148	85.148	85.035
Nº de sessões TCP rastreadas	1.456.303	1.474.669	1.474.669
Nº de sessões UDP rastreadas	17.027	17.027	16.914

Tabela 8.8 – Dados estatísticos referentes aos *traces* DARPA de sexta-feira

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.362.422	1.348.705	1.348.705
Positivos	57.987	57.987	57.987
Falso-Positivos	–	1.534	1.534
Falso-Negativos	–	0	0
Nº Pacotes Anonimizados	–	1.348.705	1.348.705
Nº conexões TCP	50.005	50.005	50.005
Nº conexões UDP	12.570	12.570	12.570
Nº conexões SMTP	3.118	3.118	3.118
Nº total de sessões	62.575	62.575	62.575
Nº de sessões TCP rastreadas	1.263.543	1.280.621	1.280.621
Nº de sessões UDP rastreadas	12.570	12.570	12.570

nos resultados de rastreamento de protocolos TCP e UDP, de acordo com a amostra apresentada, com uma margem de diferença em relação ao original inferior a 1%.

Estes resultados apontam uma eficiência em rastreabilidade do bomIP semelhante ao Crypto-Pan, porém, com um tempo de processamento inferior, conforme verificado através da análise de tempo de processamento entre as ferramentas, o que permite que ele possa ser utilizado em tempo real em uma rede para anonimizar os *hosts* adicionados dinamicamente à uma rede SDN.

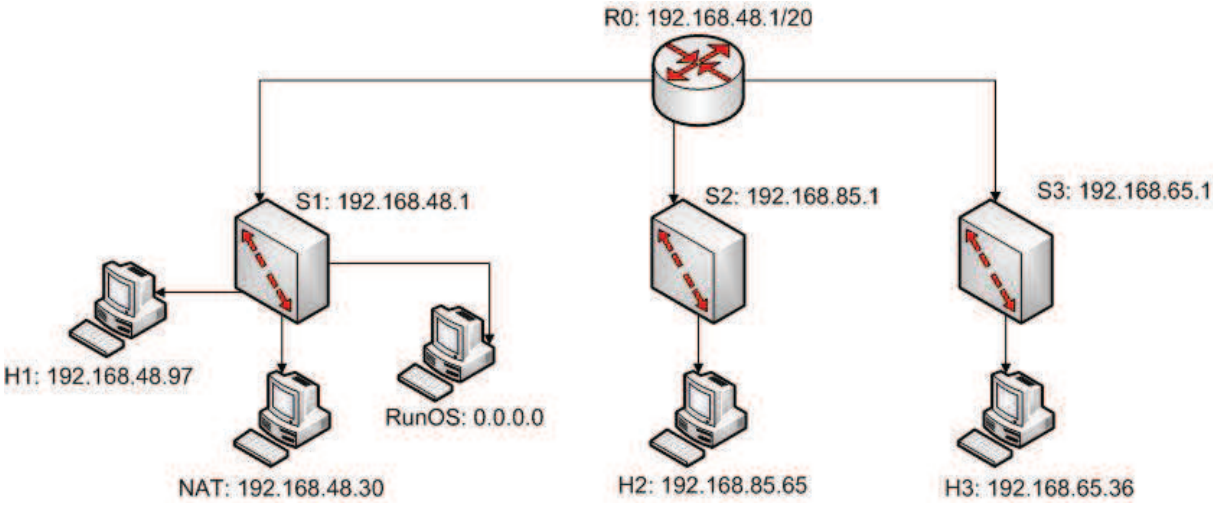
8.4 ESTUDO DE CASO II

O segundo experimento, baseado nos trabalhos de (NI et al., 2015) e (ZHAO; GUO; LIU, 2015), é realizado através de uma rede simulada, criada utilizando um *script* desenvolvido em *Python* para *Mininet* (ROGERIO et al., 2014).

O objetivo deste experimento é apresentar a mitigação do quantitativo de ataques que a rede sofre, quando os endereços IP contidos na Tabela de IP do controlador encontram-se anonimizados.

Esta rede possui o roteador principal com um endereço IP de uma máscara de 20 bits e duas sub-redes, conforme apresentado na Figura 8.9, o *script* de criação da rede é o apresentado no Algoritmo 8.3.

Figura 8.9 – Ambiente simulado criado no Mininet



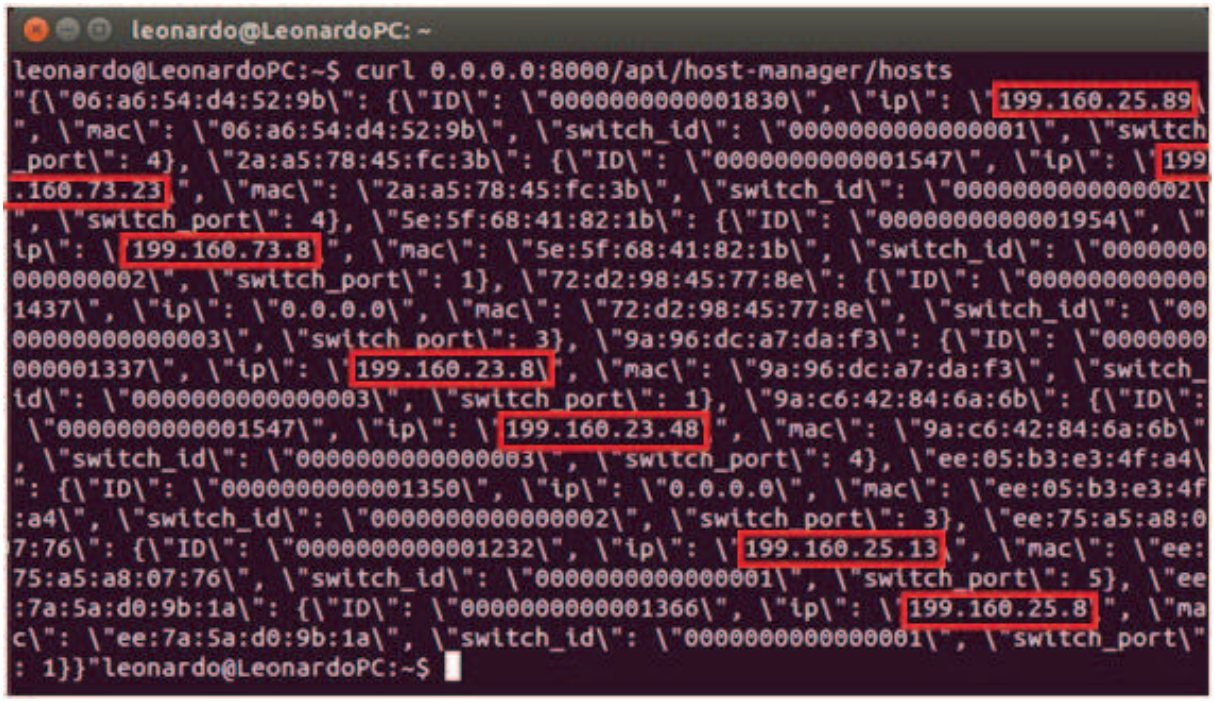
No código 8.3 é criado o roteador principal e 3 *switches*. No primeiro *switch* (S1) está conectado o *host* que possui o controlador RunOS, responsável pelo controle e gerência do tráfego da rede. Em S1 também está conectado um *host* (H1) e um serviço de NAT (nat1). O segundo *switch* (S2) corresponde a uma sub-rede com um *host* (H2) conectado; e o terceiro *switch* (S3) possui também um *host* (H3) conectado.

Este controlador RunOS, executado na rede, utiliza o serviço de anonimização de endereços IP denominado BomIP, responsável por anonimizar de forma randômica, porém, realizando a preservação de prefixos dos endereços.

8.4.1 ANÁLISE DO ESTUDO DE CASO II

Iniciou-se um ataque à rede tomando como ponto de partida que o usuário conseguiu acesso à um *host*. Com base neste *host*, o usuário conseguiu acesso à Tabela de IPs do controlador RunOS. Na Figura 8.10 é apresentada a Tabela de IPs do controlador RunOS, estão destacados em vermelho os endereços IP dos *hosts* da rede.

Figura 8.10 – Tabela de IPs do controlador RunOS



Porém, a Tabela de IP encontra-se anonimizada nos campos referentes ao endereço IP dos *hosts* que compõem a rede. O que impede que o invasor consiga realizar um ataque direto aos IPs encontrados. Os IPs encontrados foram: 199.160.25.89, 199.160.73.23, 199.160.73.8, 199.160.23.8, 199.160.23.48, 199.160.25.13 e 199.160.25.8. Para ter acesso a estes endereços IP, o usuário invasor solicitou a visualização da tabela de endereços IP do controlador RunOS através do comando *curl 0.0.0.0:8000/api/host-manager/hosts*, que corresponde ao comando disponibilizado na documentação do controlador para visualização da tabela.

Quando realizou o acesso ao host, o invasor consegue inferir:

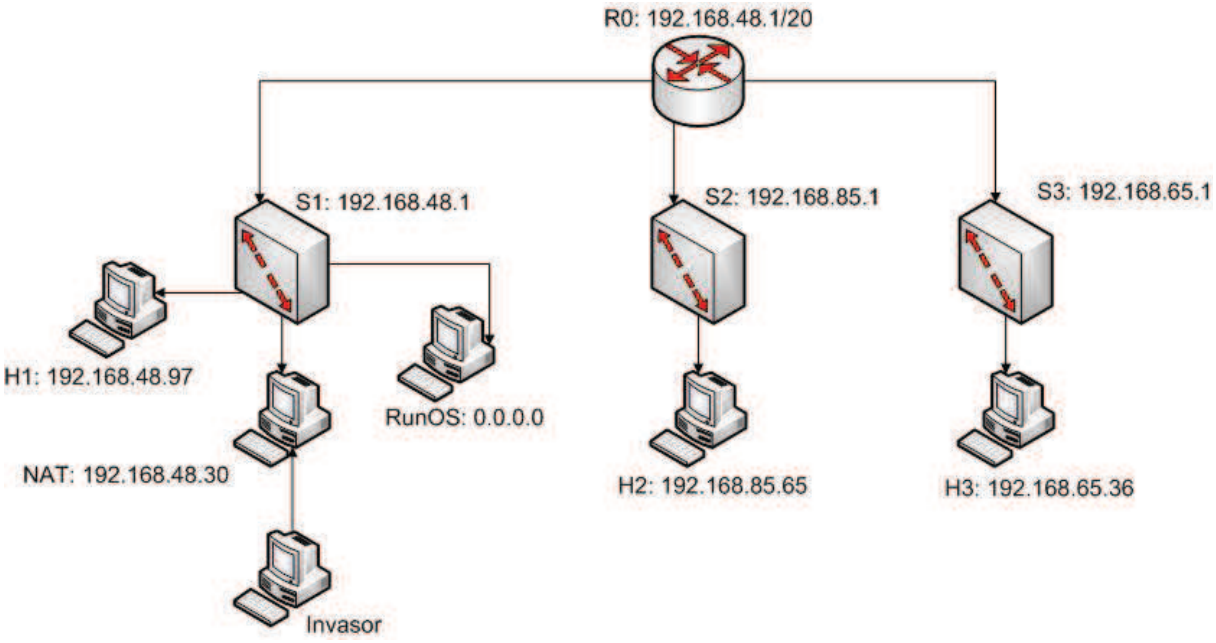
1. Os valores anonimizados 199 e 160 correspondem, respectivamente, a 192 e 168 não anonimizados. Isto porque o IP utilizado para ataque corresponde a 192.168.48.30.
2. Existem 3 sub-redes, 192.168.23.X, 192.168.25.Y e 192.168.73.Z, em que 23, 25 e 73 são valores anonimizados de forma randômica, e que um destes valores corresponde a 48.
3. Os *hosts* disponíveis correspondem ao último octeto, todos anonimizados, com valores 8, 13, 23, 48, 89, e um destes valores corresponde a 30.

Com base nesses pontos, o invasor tem como informação inicial para ataque apenas a classe do endereço, o que representa um endereço de classe B, com 16 bits de rede e 16 bits de *hosts*, com um total de 65.536 endereços possíveis para se encontrar os *hosts*. Caso ele tivesse mais informações, poderia reduzir o intervalo de busca de *hosts*.

Em cada tentativa de localização de um *host*, é usado o programa NMAP (GORDON, 2009) para verificar a existência do *host* e se há porta aberta disponível para ataque. Caso tenha sucesso, é usado o programa DDOSIM (CHENGXU; KESONG, 2011) para realizar um ataque de DDoS no *host* localizado na porta aberta indicada pelo NMAP. A arquitetura deste ataque é a apresentada na Figura 8.11.

De forma a otimizar a simulação de tentativas de ataques à rede, foi desenvolvida uma aplicação em Linguagem C, específica para este Estudo de Caso, com o objetivo de facilitar as chamadas aos programas NMAP e DDOSIM para cada tentativa de ataque a uma porta de um endereço IP. Nesta aplicação são fornecidos os octetos conhecidos e a quantidade de tentativas de ataques desejadas. Para cada tentativa de ataque, a aplicação busca um novo endereço de IP contento os octetos conhecidos fornecidos como parâmetros iniciais. Esta aplicação possibilitou que fosse possível esgotar todos os endereços IP possíveis de ataque em uma dada faixa da rede, de forma a permitir mensurar o tempo gasto com ataque e a quantidade de máquinas afetadas.

Figura 8.11 – Arquitetura de tentativa de ataque à rede



De acordo com (TONNES; ANDRE; ARNE, 2005), algoritmos de criptografia ou randômicos são quebrados através de técnicas de força bruta, ou seja, o usuário (invasor) deve tentar todas as possibilidades disponíveis. Isto implica que, a partir do momento que não há uma dessas técnicas impostas, o invasor pode tentar um acesso direto ou usar heurísticas para conseguir obter com mais facilidade os valores possíveis.

Com o diagrama apresentado na Figura 8.11 pode-se ter dois cenários:

1. A rede não se encontra anonimizada pelo controlador. Logo, quando o invasor entrar na rede, conseguirá todos os endereços IP reais, e assim, pode realizar um ataque de DDoS de forma direta em cada uma das estações desejadas.
2. A rede encontra-se anonimizada. Assim, o usuário tem apenas o conhecimento dos octetos do *host* que conseguiu realizar a invasão, e com isso, apenas uma das 3 classes de IP como conhecimento, tendo que usar todos os bits possíveis de hosts da classe para tentar atacar a rede.

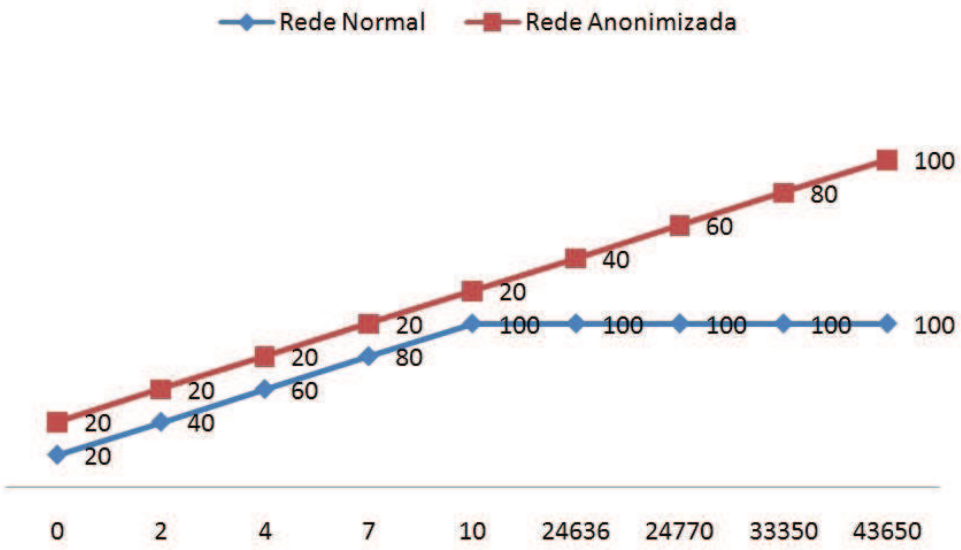
Considerando esses dois cenários mencionados, e a rede apresentada na Figura 8.11, foi realizada uma tentativa de ataque em cada um dos cenários propostos, usando a aplicação desenvolvida para automatizar ataques na rede. Desta forma, foi possível aferir o tempo gasto para cada tentativa de ataque, considerando a obtenção de portas disponíveis em um endereço IP através do uso da ferramenta NMAP, e em caso de encontrar uma porta aberta no endereço IP, realizar uma tentativa de ataque utilizando a ferramenta DDOSIM. Como resultado dessas tentativas de ataque, foi verificado um tempo médio de 2 segundos por IP tentado a ser atacado.

Assim, A Figura 8.12 apresenta o resultado obtido. O eixo X corresponde ao tempo decorrido de ataque em segundos, enquanto que na curva do gráfico é possível verificar o percentual de *hosts* afetados pelas tentativas de ataque.

A rede normal teve a totalidade de seus hosts (100%) afetada em 10 segundos. Enquanto que, devido ao intervalo a ser pesquisado pelo desconhecimento dos endereços reais dos hosts, a rede anonimizada foi afetada 100% pelo ataque com 43.650 segundos (aproximadamente 12 horas).

Importante visualizar que quando a rede normal teve a totalidade de seus *hosts* sob ataque, a rede anonimizada tinha sido 20% afetada.

Figura 8.12 – Resultado de tentativa de ataque à rede



Esta diferença no tempo mostra a mitigação do ataque a que a rede anonimizada está sofrendo, bem como permite que medidas sejam tomadas quando um ataque é iniciado, já que há um intervalo entre o ataque a uma estação e outra.

Assim, no pior cenário, o invasor deve percorrer todo o intervalo possível da faixa de IP de *hosts*, com base na quantidade de bits disponíveis na máscara da classe do IP. Neste exemplo, considerando um tempo gasto de 2 segundos por tentativa de invasão a um *host*, e sabendo que uma máscara de classe B possui até 65.536 endereços, o que levaria um tempo de, aproximadamente, 36 horas de tentativa de ataque.

8.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os Estudos de Caso realizados para validação da proposta desta dissertação, que consiste na implementação de um serviço de anonimização para Redes Definidas por Software. O serviço desenvolvido, denominado de BomIP, apresentou uma análise assintótica de ordem quadrática, o que representou um ganho de tempo de aproximadamente 65% em relação ao anonimizador Crypto-Pan. Esta análise comparativa entre o tempo de execução do BomIP e Crypto-Pan foi realizada utilizando os *traces* da DARPA (MIT). Em outro experimentou foi possível realizar a comparação entre os *traces* originais da DARPA, com os mesmos sendo anonimizados pelo BomIP, e também sendo anonimizados com Crypto-Pan, o que permitiu aferir a rastreabilidade dos pacotes dos arquivos. E por fim, um último experimento foi realizado para verificar a eficácia do serviço

de anonimização em mitigar tentativas de ataque de Negação de Serviço, comparando uma rede IP tradicional com uma SDN utilizando o serviço BomIP, e neste cenário, houve uma redução de, aproximadamente, 80% dos ataques sofridos pela rede SDN em comparação com a rede tradicional.

Algoritmo 8.3 Script de criação da rede no Mininet

```
1: classLinuxRouter(Node) :
2: defconfig(self, **params) :
3: super(LinuxRouter, self).config(**params)
4: self.cmd('sysctlnet.ipv4.ip_forward = 1')
5: defterminate(self) :
6: self.cmd('sysctlnet.ipv4.ip_forward = 0')
7: super(LinuxRouter, self).terminate()
8: classNetworkTopo(Topo) :
9: defbuild(self, **_o pts) :
10: defaultIP =' 192.168.48.1/20'
11: router = self.addNode('r0', cls = LinuxRouter, ip = defaultIP)
12: s1, s2, s3 = [self.addSwitch(s)forsin('s1',' s2',' s3')]
13: self.addSwitch('s1', protocols = ["OpenFlow13"])
14: self.addSwitch('s2', protocols = ["OpenFlow13"])
15: self.addSwitch('s3', protocols = ["OpenFlow13"])
16: self.addLink(s1, router, intfName2 =' r0 − eth1',
17: params2 = 'ip' : defaultIP)
18: self.addLink(s2, router, intfName2 =' r0 − eth2',
19: params2 = 'ip' :' 192.168.85.1/20')
20: self.addLink(s3, router, intfName2 =' r0 − eth3',
21: params2 = 'ip' :' 192.168.65.1/10')
22: h1 = self.addHost('h1', ip =' 192.168.48.97/20',
23: defaultRoute =' via192.168.48.1', inNamespace = False)
24: h2 = self.addHost('h2', ip =' 192.168.85.65/20',
25: defaultRoute =' via192.168.85.1', inNamespace = False)
26: h3 = self.addHost('h3', ip =' 192.168.65.36/20',
27: defaultRoute =' via192.168.65.1', inNamespace = False)
28: self.addLink(s1, s2)
29: self.addLink(s1, s3)
30: nat1 = self.addNode('nat1', cls = NAT, ip =' 192.168.48.30/20',
31: inNamespace = False)
32: self.addLink(nat1, s1)
33: self.addLink(nat1, s2)
34: self.addLink(nat1, s3)
35: forh, sin[(h1, s1), (h2, s2), (h3, s3)] :
36: self.addLink(h, s)
37: defrun() :
38: topo = NetworkTopo()
39: net = Mininet(topo = topo,
40: controller = lambdaname : RemoteController(name, ip =' 0.0.0.0'),
41: )
42: net.start()
43: CLI(net)
44: net.stop()
45: setLogLevel('info')
46: run()
```

9 CONCLUSÕES

Este trabalho apresentou uma solução voltada para segurança de Redes Definidas por Software através da implementação de um serviço de anonimização de redes SDN. Durante o desenvolvimento do trabalho, foi levantada a hipótese de que a utilização de uma ferramenta capaz de realizar o ocultamento de estações em uma rede iria mitigar a quantidade de ataques que a própria rede poderia sofrer, devido ao invasor não conseguir obter um conhecimento acerca dos endereços IP reais das estações e serviços disponíveis. Foi proposta, então, a criação de um serviço de anonimização em redes SDN, com o objetivo de anonimizar os endereços IP da rede.

9.1 CONCLUSÕES DOS RESULTADOS OBTIDOS

O serviço proposto, denominado de BomIP, foi desenvolvimento em Linguagem C e adicionado ao controlador SDN RunOS. Este controlador ficou encarregado de gerenciar a anonimização dos endereços IP dos *hosts* da rede, mantendo a tabela de endereços IP anonimizada. Em seguida, simulou-se um ambiente SDN utilizando o Mininet. E, para simular ataques na rede, utilizou-se o NMAP para obtenção de portas disponíveis na rede e o DDOSIM para simular um ataque de Negação de Serviço na rede.

A análise assintótica do BomIP verificou que o mesmo possui ordem quadrática, o que demonstrou uma melhora em relação ao Crypto-Pan, que possui ordem à quarta. Ainda nesta análise, utilizou-se os *traces* disponibilizados pela DARPA (MIT) para aferir o tempo real gasto por cada ferramenta para realizar a anonimização dos arquivos. O

BomIP anonimizou os *traces* com um tempo 65% melhor que o Crypto-Pan, o que validou a eficácia em relação ao tempo de processamento do BomIP com o Crypto-Pan.

Em seguida, um primeiro Estudo de Caso foi realizado, para validar a rastreabilidade dos pacotes gerados pelo BomIP, para desta forma, validar a preservação de prefixo no processo de anonimização. O experimento consistiu de utilizar os *traces* originais da DARPA, contendo ataques de Negação de Serviço, e analisá-los no SNORT. Em seguida, gerou-se novos *traces* com o BomIP, tendo como parâmetro de entrada os *traces* da DARPA, e o mesmo procedimento foi realizado com o Crypto-Pan. Em seguida, comparou-se a quantidade de ataques de Negação de Serviço obtidos com os *traces* gerados pelo Crypto-Pan e BomIP em relação aos valores originais. Os resultados demonstraram que, aproximadamente, apenas 1% dos pacotes não foram anonimizados ou não classificados com ataque em relação aos originais, isto foi explicado devido ao BomIP e Crypto-Pan filtrar os pacotes IPv6. Porém, a análise das tentativas de ataque foram detectados nos *traces* gerados pelo BomIP.

Por fim, outro cenário de Estudo de Caso foi proposto, para validar a mitigação de ataques em uma rede. Para este experimento comparou-se uma rede IP tradicional sofrendo ataque de Negação de Serviço, e uma rede SDN com serviço de anonimização do BomIP também sob ataque de Negação de Serviço. Os resultados apresentados demonstraram que a rede SDN teve uma mitigação de 80% em relação a rede tradicional IP, devido ao desconhecimento prévio do usuário dos IP reais da rede por conta da anonimização dos endereços no controlador SDN.

Diante dos resultados apresentados, foi possível perceber que a utilização do serviço proposto, apresentou-se de forma positiva, com resultados que mitigaram a quantidade de ataques sofridos pela rede em comparação a uma rede que não utilizava o serviço de anonimização, conduzindo a um maior tempo até que a rede fosse invadida, permitindo assim, que administradores da rede possam tomar medidas para conter a invasão.

É possível destacar como principal contribuição deste trabalho, a apresentação de um serviço de anonimização em redes SDN, com a capacidade de aplicar as técnicas preservação de prefixo através de uma anonimização randômica, garantindo que novas sub-redes sejam criadas, mantendo assim a rastreabilidade dos *traces* gerados, o que permite que os dados coletados possam ser analisados pelos administradores da rede para avaliar as tentativas de ataque que a rede pode ter sofrido e o funcionamento da mesma.

9.2 TRABALHOS FUTUROS

Encontra-se em desenvolvimento a anonimização das informações dos pacotes referentes à portas e endereços MAC, o que vai garantir maior confiança ao serviço. Também encontra-se em fase de desenvolvimento a anonimização de pacotes IPv6, já que atualmente, a ferramenta apenas analisa os pacotes em formato IPv4.

E por fim, estudos estão sendo realizados para alterar o vetor que armazena os valores anonimizados, para um conjunto de vetores, um vetor para cada octeto, permitindo que um valor anonimizado possa ser utilizado para dois ou mais valores reais, e mesmo assim, mantendo a preservação de prefixo.

REFERÊNCIAS

ACCETTURA, N.; NEGLIA, G.; GRIECO, L. The capture-recapture approach for population estimation in computer networks. *Computer Networks*, v. 89, p. 107–122, 2015. Citado na página 45.

AKHUNZADA, A. et al. Securing software defined networks: Taxonomy, requirements, and open issues. *IEEE Communications Magazine*, v. 54, p. 36–44, 2015. Citado 3 vezes nas páginas 16, 28 e 30.

ALEXANDER, S. et al. The runos openflow controller. In: IEEE. *2015 Fourth European Workshop on Software Defined Networks*. [S.l.], 2015. p. 103–104. Citado 7 vezes nas páginas 8, 22, 50, 59, 60, 61 e 62.

ALI, S. et al. A survey of securing networks using software defined networking. *IEEE Transactions on Reliability*, v. 64, n. 3, p. 1086–1097, 2015. Citado 3 vezes nas páginas 45, 47 e 49.

ALLAN, V.; CHRISTIAN, R.; FABIO, V. The libfluid openflow driver implementation. In: *32nd Brazilian Symposium on Computer Networks (SBRC). SBC*. [S.l.: s.n.], 2014. p. 8. Citado 4 vezes nas páginas 8, 59, 60 e 61.

ALSMADI, I.; XU, D. Security of software defined networks: A survey. *Computers and Security*, v. 53, p. 79–106, 2015. Citado na página 45.

AYALA-RIVERA, V. et al. A systematic comparison and evaluation of k-anonymization algorithms for practitioners. *Transactions on Data Privacy*, v. 7, p. 337–370, 2014. Citado na página 36.

BAIARDI, F. et al. Seas: A secure e-voting applet system. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 3233, p. 318–329, 2004. Citado 4 vezes nas páginas 44, 47, 48 e 49.

BAIARDI, F. et al. Seas a secure e-voting protocol: Design and implementation. *Computers and Security*, v. 24, n. 8, p. 642–652, 2005. Citado 4 vezes nas páginas 44, 47, 48 e 49.

BALASHOV, V. et al. An analysis of approaches to onboard networks design. In: . [S.l.: s.n.], 2014. Citado na página 45.

BALLARD, J.; RAE, I.; AKELLA, A. Extensible and scalable network monitoring using opensafe. *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, v. 1, 2010. Citado na página 29.

BENTON, K.; CAMP, L. J.; SMALL, C. Openflow vulnerability assessment. *ACM SIGCOMM HotSDN*, v. 1, p. 151–152, 2013. Citado na página 29.

BRAGA, R.; MOTA, E.; PASSITO, A. Lightweight ddos flooding attack detection using nox/openflow. *IEEE 35th Conference on Local Computer Networks*, v. 1, p. 408–415, 2010. Citado na página 29.

BREMLER-BARR, A. et al. Deep packet inspection as a service. In: . [S.l.: s.n.], 2014. p. 271–282. Citado na página 44.

BRIAN, K.; DENNIS, R. *C: a linguagem de programação*. [S.l.]: Campus, 1989. Citado na página 51.

CARNIELLI, A.; AIASH, M. Will tor achieve its goals in the 'future internet'? an empirical study of using tor with cloud computing. In: . [S.l.: s.n.], 2015. p. 135–140. Citado na página 44.

CHAKRABARTY, S.; ENGELS, D.; THATHAPUDI, S. Black sdn for the internet of things. In: . [S.l.: s.n.], 2015. p. 190–198. Citado na página 44.

CHAVEZ, A.; STOUT, W.; PEISERT, S. Techniques for the dynamic randomization of network attributes. *Proceedings of the 49th Annual International Carnahan Conference on Security Technology*, v. 1, p. 01–06, 2015. Citado 4 vezes nas páginas 34, 65, 67 e 69.

CHAVEZ, A.; STOUT, W.; PEISERT, S. Techniques for the dynamic randomization of network attributes. In: . [S.l.: s.n.], 2016. v. 2015-January. Citado 4 vezes nas páginas 45, 47, 48 e 49.

CHEN, C. et al. Automated verification of safety properties of declarative networking programs. In: . [S.l.: s.n.], 2015. p. 79–90. Citado na página 45.

CHEN, Y. et al. Packetcloud: A cloudlet-based open platform for in-network services. *IEEE Transactions on Parallel and Distributed Systems*, v. 27, n. 4, p. 1146–1159, 2016. Citado 4 vezes nas páginas 45, 47, 48 e 49.

CHENGXU, Y.; KESONG, Z. Detection of application layer distributed denial of service. In: IEEE. *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*. [S.l.], 2011. v. 1, p. 310–314. Citado na página 85.

CHON, K. et al. A history of computer networking and the internet in korea. *IEEE Communications Magazine*, v. 51, n. 2, p. 10–15, 2013. Citado na página 45.

CRISTOPHER, U. et al. Flaxor: A symmetric-key block cipher plug-in using usb mass storage device for extracted block storage. *JUPITER: 1st ITE Research Colloquium/TIP-QC/ March 28*, v. 1, 2008. Citado 2 vezes nas páginas 74 e 76.

CUSTERS, B. Risk profiling of money laundering and terrorism funding - practical problems of current information strategies. In: . [S.l.: s.n.], 2007. ISAS, p. 90–94. Citado na página 44.

DABBAGH, M. et al. Software-defined networking security: pros and cons. *IEEE Communications Magazine*, v. 53, p. 73–79, 2015. Citado na página 14.

DARA, S. Network telemetry anonymization for cloud based. *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, v. 1, p. 01–07, 2014. Citado na página 35.

DESHPANDE, P. et al. The mask of zorro: preventing information leakage from documents. *Knowledge and Information Systems*, v. 45, n. 3, p. 705–730, 2015. Citado na página 44.

DORIA, A. et al. *Forwarding and Control Element Separation (FORCES) Protocol Specification*. 2015. ⟨https://ietf.org/⟩. Accessed: 2015-08-11. Citado na página 19.

DU, X. et al. Traffic-based malicious switch detection in sdn. *International Journal of Security and its Applications*, v. 8, n. 5, p. 119–130, 2014. Citado na página 44.

FARAH, T.; TRAJKOVI, L. Anonym: A tool for anonymization of the internet traffic. *IEEE International Conference on Cybernetics (CYBCONF)*, v. 1, p. 261–266, 2013. Citado 2 vezes nas páginas 34 e 37.

FARHADY, H.; LEE, H.; NAKAO, A. Software-defined networking: A survey. *Computer Networks*, v. 85, p. 79–95, 2015. Citado 4 vezes nas páginas 8, 22, 24 e 25.

FEGHALI, A.; KILANY, R.; CHAMOUN, M. Sdn security problems and solutions analysis. *International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, v. 1, p. 01–05, 2015. Citado na página 16.

FOUNDATION, O. N. *OpenFlow Switch Specification v1.3.0*. 2012. ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflowspec-v1.3.0.pdf⟩. Accessed: 2015-06-15. Citado na página 23.

FOUNDATION, S. *Software Defined Networking*. 2015. ⟨https://www.opennetworking.org⟩. Accessed: 2015-06-20. Citado na página 19.

GABOR, A.; SZABO, Z. *Semantic technologies in business process management*. [S.l.: s.n.], 2013. 17-28 p. Citado na página 44.

GARG, G.; GARG, R. Security of networks using efficient adaptive flow counting for anomaly detection in sdn. *Advances in Intelligent Systems and Computing*, v. 394, p. 667–674, 2016. Citado na página 44.

GERSON, L. *The Cambridge history of philosophy in late antiquity volume II*. [S.l.: s.n.], 2010. 1-1284 p. Citado na página 44.

GOLLASCH, S.; ROSENTHAL, H. *The world's busiest man-made waterway and biological invasions*. [S.l.: s.n.], 2006. v. 83. 5-90 p. Citado na página 44.

GORDON, L. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. [S.l.]: Insecure, 2009. Citado na página 85.

HALEPLIDIS, E. et al. *Software-Defined Networking (SDN): Layers and Architecture Terminology*. 2015. RFC 7426. Citado 2 vezes nas páginas 19 e 20.

HAN, S. et al. Packetshader: a gpu-accelerated software router. *ACM SIGCOMM CCR*, v. 40, p. 195–206, 2010. Citado na página 25.

HARSH, V.; RAVINDRA, S. Performance analysis of rc6, twofish and rijndael block cipher algorithms. *International Journal of Computer Applications (0975–8887) Volume*, Citeseer, p. 1–7, 2012. Citado na página 73.

HASEGAWA, T. A survey of the research on future internet and network architectures. *IEICE Transactions on Communications*, E96-B, n. 6, p. 1385–1401, 2013. Citado na página 45.

HELLER, B.; SHERWOOD, R.; MCKEOWN, N. The controller placement problem. *ACM SIGCOMM HotSDN*, v. 1, p. 07–12, 2012. Citado na página 32.

HELLER, B.; SHERWOOD, R.; MCKEOWN, N. *Software-Defined Networking: A Perspective from within a Service Provider Environment*. 2014. RFC 7149. Citado na página 29.

HOFSTEDE, R. et al. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys and Tutorials*, v. 16, n. 4, p. 2037–2064, 2014. Citado 4 vezes nas páginas 45, 47, 48 e 49.

HU, Z. et al. A comprehensive security architecture for sdn. *International Conference on Intelligence in Next Generation Networks (ICIN)*, v. 1, p. 30–37, 2015. Citado na página 16.

IRFAN, A.; TAJ, N.; MAHMUD, S. A novel secure sdn/lte based architecture for smart grid security. In: . [S.l.: s.n.], 2015. p. 762–769. Citado na página 45.

JAFARIAN, H.; AL-SHAER, E. Adversary-aware ip address randomization for proactive agility against sophisticated attackers. *IEEE Conference on Computer Communications*, v. 1, p. 738–746, 2015. Citado 2 vezes nas páginas 66 e 69.

JAFARIAN, J. H.; AL-SHAER, E.; DUAN, Q. Openflow random host mutation: Transparent moving target defense using software defined networking. *ACM Workshop Hot Topics in Software Defined Networks*, v. 1, p. 127–132, 2012. Citado 6 vezes nas páginas 45, 47, 48, 49, 66 e 69.

JAMMAL, M. et al. Software defined networking: State of the art and research challenges. *Computer Networks*, v. 72, 2014. Citado 2 vezes nas páginas 19 e 32.

JINLIANG, F.; JUN, X.; MOSTAFA, A. Crypto-pan: Cryptography-based prefix-preserving anonymization. *Computer Networks*, v. 46, n. 2, 2004. Citado 2 vezes nas páginas 73 e 79.

JINLIANG, F. et al. Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*, Elsevier, v. 46, n. 2, p. 253–272, 2004. Citado na página 52.

JOAN, D.; VINCENT, R. The block cipher rijndael. In: SPRINGER. *International Conference on Smart Card Research and Advanced Applications*. [S.l.], 1998. p. 277–284. Citado na página 73.

JOSHI, K.; CHATURVEDI, P. Therapeutic efficiency of centella asiatica (l.) urb. an underutilized green leafy vegetable: An overview. *International Journal of Pharma and Bio Sciences*, v. 4, n. 1, p. 135–149, 2013. Citado na página 44.

KANIZO, Y.; HAY, D.; KESLASSY, I. Palette: Distributing tables in software-defined networks. *2013 Proceedings IEEE INFOCOM*, v. 1, p. 545–549, 2013. Citado na página 25.

KAYDAN, M.; KOZAR, F.; ERKILIC, L. New and little known scale insect species (hemiptera: Coccoidea) in turkey. *Acta Zoologica Academiae Scientiarum Hungaricae*, v. 60, n. 3, p. 227–238, 2014. Citado na página 44.

KHAN, S. et al. Network forensics: Review taxonomy and open challenges. *Journal of Network and Computer Applications*, v. 66, p. 214–235, 2016. Citado na página 45.

KIRAN, L.; ADAM, S. Evaluating the utility of anonymized network traces for intrusion detection. In: ACM. *Proceedings of the 4th international conference on Security and privacy in communication netowrks*. [S.l.], 2008. p. 17. Citado na página 79.

KONSTANTINOV, I. et al. The development of infrastructure security for distributed information computer environment based on secured portal network. *International Journal of Applied Engineering Research*, v. 10, n. 24, p. 45034–45042, 2015. Citado na página 44.

KOUKIS, D. et al. A generic anonymization framework for network traffic. *IEEE International Conference on Communications*, v. 1, p. 2302–2309, 2006. Citado na página 16.

KUROSE, J. *Redes de Computadores e a Internet - Uma Abordagem Top-down*. 6th. ed. [S.l.]: Pearson Education, 2013. Citado na página 27.

KUZNIAR, M. et al. Automatic failure recovery for software-defined networks. *ACM SIGCOMM HotSDN*, v. 1, p. 159–160, 2013. Citado na página 31.

LEONG, S. *New media and the nation in Malaysia: Malaysianet*. [S.l.: s.n.], 2013. 1-167 p. Citado na página 44.

LI, S.; DOH, I.; CHAE, K. An anonymous ip-based privacy protection routing mechanism for cdni. In: . [S.l.: s.n.], 2016. v. 2016-March, p. 75–80. Citado 4 vezes nas páginas 45, 47, 48 e 49.

LI, W.; MENG, W.; KWOK, L. A survey on openflow-based software defined networks: Security challenges and countermeasures. *Journal of Network and Computer Applications*, v. 68, p. 126–139, 2016. Citado 3 vezes nas páginas 45, 47 e 49.

LUO, Y. et al. Accelerating openflow switching with network processors. *ACM/IEEE ANCS*, v. 1, p. 70–71, 2009. Citado na página 25.

MANBER, U. *Introduction to algorithms: a creative approach*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1989. Citado 3 vezes nas páginas 73, 74 e 75.

MARTIN, G. L. Programming with libpcap-sniffing the network from our own application. *Hakin9-Computer Security Magazine*, p. 2–2008, 2008. Citado na página 51.

MARTIN, R.; CHRIS, G. Snort users manual. *Snort Release*, v. 1, n. 3, 2003. Citado na página 79.

MATTSON, K. P. R. F. S. M. M. Systematic mapping studies in software enginnering. *Proccedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, v. 1, p. 68–77, 2008. Citado na página 40.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM*, v. 38, p. 69–74, 2008. Citado na página 19.

MELO, M.; GUEDES, D. Anonv: uma arquitetura para verificação do grau de anonimização em coletas de tráfego de rede. *Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais*, v. 1, p. 367–380, 2010. Citado na página 28.

MENDONCA, M.; SEETHARAMAN, S.; OBRACZKA, K. A flexible in-network ip anonymization service. *IEEE Int. Conf. Communications*, v. 1, p. 6651–6656, 2012. Citado 8 vezes nas páginas 16, 34, 45, 47, 48, 49, 67 e 69.

MIET, W. Searching optimization in p2p distributed system. In: . [S.l.: s.n.], 2010. v. 1, p. 272–277. Citado na página 44.

NATASHA, G. et al. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 3, p. 105–110, 2008. Citado na página 47.

NI, D. et al. Time-based ddos detection and mitigation for sdn controller. In: IEEE. *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. [S.l.], 2015. p. 550–553. Citado na página 82.

NIA, M. et al. A software solution for realtime malware detection in distributed systems. In: . [S.l.: s.n.], 2015. p. 144–149. Citado na página 44.

OTHMAN, K. O. M. Hybrid control model for flow-based networks. *IEEE COMPSAC*, v. 1, p. 765–770, 2013. Citado na página 21.

PAN, H. et al. The flowadapter: enable flexible multi-table processing on legacy hardware. *ACM SIGCOMM HotSDN*, v. 38, p. 85–90, 2013. Citado na página 25.

PAULO, V. Towards secure and dependable software-defined networks. *ACM SIGCOMM HotSDN*, v. 1, p. 55–60, 2013. Citado 2 vezes nas páginas 29 e 30.

PORRAS, P. et al. A security enforcement kernel for openflow networks. *ACM SIGCOMM HotSDN*, v. 1, p. 121–126, 2012. Citado na página 31.

REBOLLO-MONEDERO, D. et al. Optimizing the design parameters of threshold pool mixes for anonymity and delay. *Computer Networks*, v. 67, p. 180–200, 2014. Citado 4 vezes nas páginas 45, 47, 48 e 49.

RICHARD, L. et al. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, Elsevier, v. 34, n. 4, p. 579–595, 2000. Citado 5 vezes nas páginas 10, 70, 72, 73 e 79.

ROGERIO, O. et al. Using mininet for emulation and prototyping software-defined networks. In: IEEE. *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*. [S.l.], 2014. p. 1–6. Citado na página 82.

ROMEO, V. et al. Correlative imaging in a patient with cystic thymoma: Ct and mr and pet/ct comparison. *Polish Journal of Radiology*, v. 80, n. 1, p. 22–26, 2015. Citado na página 44.

ROSSITER, N. Coded vanilla: Logistical media and the determination of action. *South Atlantic Quarterly*, v. 114, n. 1, p. 135–152, 2015. Citado na página 44.

SAHANI, D. et al. State-of-the-art pet/ct of the pancreas: Current role and emerging indications. *Radiographics*, v. 32, n. 4, p. 1133–1158, 2012. Citado na página 44.

SAMSUDIN, A.; SURYANA, N. Peer selection for multi-source streaming approach on p2p file sharing application. In: . [S.l.: s.n.], 2009. p. 1–6. Citado na página 44.

SANATINIA, A.; NARAIN, S.; NOUBIR, G. Wireless spreading of wifi aps infections using wps flaws: An epidemiological and experimental study. In: . [S.l.: s.n.], 2013. p. 430–437. Citado na página 44.

SCHWEITZER, J.; HANNAN, A.; COREN, J. The role of social networking web sites in influencing residency decisions. *Journal of the American Osteopathic Association*, v. 112, n. 10, p. 673–679, 2012. Citado na página 44.

SCOTT-HAYWARD, S. Sdn security: A survey. *IEEE SDN for Future Networks and Services*, v. 1, p. 01–07, 2013. Citado na página 28.

SCOTT-HAYWARD, S. Design and deployment of secure, robust, and resilient sdn controllers. *IEEE Conference on Network Softwarization*, v. 1, p. 01–05, 2015. Citado na página 15.

SHIN, S.; GU, G. Attacking software-defined networks: a first feasibility study. *ACM SIGCOMM HotSDN*, v. 1, p. 165–166, 2013. Citado na página 31.

SHIN, S. et al. Fresco: Modular composable security services for software-defined networks. *Internet Society NDSS*, 2013. Citado na página 30.

SHIRALI-SHAHREZA, S.; GANJALI, Y. Flexam: Flexible sampling extension for monitoring and security applications in openflow. *Proc. 2nd ACM SIGCOMM Wksp. Hot Topics in Software Defined Networking*, v. 1, p. 167–168, 2013. Citado na página 31.

SHIREY, R. *Internet Security Glossary*. 2000. RFC 2828. Citado 3 vezes nas páginas 26, 27 e 28.

SHUANGYU, H. et al. Hierarchical solution for access control and authentication in software defined networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 8792, p. 70–81, 2014. Citado na página 44.

SIEK, J.; VACHHARAJANI, M. Gradual typing with unification-based inference. In: . [S.l.: s.n.], 2008. Citado na página 44.

SKOWYRA, R. et al. Verifiably-safe software-defined networks for cps. *2nd ACM Int'l. Conf. High Confidence Networked Systems*, v. 1, p. 101–110, 2013. Citado na página 31.

SON, S. Model checking invariant security properties in openflow. *IEEE ICC*, v. 1, p. 1974–1979, 2013. Citado na página 31.

SONY, C. Traffic data repository at the wide project. In: *Proceedings of USENIX 2000 Annual Technical Conference: FREENIX Track*. [S.l.: s.n.], 2000. p. 263–270. Citado na página 71.

SOO, W.; SAMSUDIN, A.; GOH, A. Efficient mental card shuffling via optimised arbitrary-sized benes permutation network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 2433, p. 446–458, 2002. Citado na página 44.

SUTIKNO, T.; STIAWAN, D.; SUBROTO, I. Fortifying big data infrastructures to face security and privacy issues. *Telkomnika (Telecommunication Computing Electronics and Control)*, v. 12, n. 4, p. 751–752, 2014. Citado na página 44.

SWEENEY, L. K-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, v. 1, p. 557–570, 2002. Citado na página 36.

TAN, Y. et al. *Accelerating global supply chains with IT-innovation: ITAIDE tools and methods*. [S.l.: s.n.], 2011. 1-379 p. Citado na página 44.

TANENBAUM, A. *Redes de computadores*. 5th. ed. [S.l.]: Pearson Education, 2011. Citado na página 26.

TANYINGYONG, V.; HIDELL, M.; SJöDIN, P. Improving pc-based openflow switching performance. *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, v. 1, p. 01–02, 2010. Citado na página 25.

TONNES, B.; ANDRE, A.; ARNE, O. Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In: SPRINGER. *International Workshop on Privacy Enhancing Technologies*. [S.l.], 2005. p. 179–196. Citado na página 86.

UBIK, S. Real-time anonymization in passive network. *Third International Conference on Networking and Services*, v. 1, p. 01–05, 2007. Citado na página 35.

UHLIG, D. K. F. R. P. E. V. C. E. R. S. A. S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 130, p. 14–76, 2015. Citado 3 vezes nas páginas 14, 19 e 26.

VAN, J.; MCCANNE, S. libpcap: Packet capture library. *Lawrence Berkeley Laboratory, Berkeley, CA*, 2009. Citado na página 51.

W, W. X. et al. A survey on software-defined networking. *IEEE Communications Surveys and Tutorials*, v. 17, n. 1, p. 27–51, 2015. Citado na página 44.

WANG, J. Towards a security-enhanced firewall application for openflow networks. *Cyberspace Safety and Security, Springer*, v. 1, p. 92–193, 2013. Citado na página 31.

WANG, W.; CHEN, M.; XING, C. Sdsnm: A software-defined security networking mechanism to defend against ddos attacks. In: . [S.l.: s.n.], 2015. p. 115–121. Citado na página 44.

WEN, X. et al. Towards a secure controller platform for openflow applications. *ACM SIGCOMM HotSDN*, v. 1, p. 171–172, 2013. Citado na página 31.

WONGKHUENKAEW, T.; BOONMA, P. An efficient and scalable coordinating algorithm for distributed network intrusion detection system. *International Joint Conference on Computer Science and Software Engineering (JCSSE)*, v. 1, p. 218–223, 2015. Citado na página 27.

WU, L. et al. Study on attacking and defending techniques in ipv6 networks. *IEEE International Conference on Digital Signal Processing (DSP)*, v. 1, p. 48–53, 2015. Citado na página 28.

YANBING, L. et al. Sdsa: A framework of a software-defined security architecture. *China Communications*, v. 13, n. 2, p. 178–188, 2016. Citado na página 44.

YU, M. et al. Scalable flow-based networking with difane. *ACM SIGCOMM CCR*, v. 41, p. 351–362, 2011. Citado na página 21.

ZHANG, H. et al. Survey on cyberspace security. *Science China Information Sciences*, v. 58, n. 11, 2015. Citado na página 44.

ZHANG, J. et al. On rule placement for multi-path routing in software-defined networks. *Lecture Notes of the Institute for Computer Sciences and Social-Informatics and Telecommunications Engineering*, v. 163, p. 59–71, 2016. Citado na página 44.

ZHAO, Z.; GUO, Y.; LIU, W. The design and research for network address space randomization in openflow network. *Journal of Computer and Communications*, v. 3, p. 203–211, 2015. Citado 4 vezes nas páginas 65, 66, 69 e 82.

ZHOU, B.; PEI, J. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems*, v. 28, p. 44–77, 2011. Citado 3 vezes nas páginas 8, 36 e 37.