Abordagem Adaptativa para Proteção de Redes SDN Utilizando Moving Target Defense

Rodrigo S. S. Nunes[1], **Túlio Pascoal**[2], **Cristian H. M. Souza**[1], **Emídio Neto**[1], **Felipe S. Dantas Silva**[1]

[1]LaTARC Research Lab
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Natal-RN, Brasil

[2]SnT - University of Luxembourg
Luxembourg

Abstract. *O paradigma das Redes Definidas por Software (SDN) vem sendo cada vez mais adotado nas infraestruturas de redes de computadores. No entanto, por se tratar de uma abordagem relativamente recente, poucas estratégias de segurança tem sido empregadas para sua proteção. Um dos problemas relacionados ao uso de SDN é a presença de ataques do tipo scanning. Recentemente, as técnicas de Moving Target Defense (MTD) tem sido usadas para lidar com esses ataques. No entanto, devido à sua natureza de operação, as estratégias baseadas em MTD podem degradar consideravelmente o desempenho da rede. Assim, com o objetivo de conciliar, de forma factível, a proteção contra ataques scanning sem degradar os serviços da rede, principalmente em termos de Qualidade de Serviço (QoS), neste artigo apresentamos o MTD Adaptive Delay System (MADS). Em nossa proposta, os atrasos baseados em MTD são aplicados apenas à resposta dos pacotes quando a rede está sob ataque. Em contraste, os trabalhos existentes aplicam continuamente atrasos à todos os pacotes trafegados e, portanto, incorrem em mais degradação na rede. MADS foi extensivamente avaliado e comparado com o estado da arte em defesas baseadas em MTD. Os resultados das avaliações demonstraram que MADS promove menor degradação na rede em termos de latência, pacotes Bad TCP e vazão.*

1. Introdução

O paradigma das Redes Definidas por Software (SDN, do inglês *Software-Defined Networking*) tem mudado a forma como as infraestruturas de redes são planejadas e operacionalizadas [Silva et al. 2020]. Diversas propostas recentes tem sido lançadas através de distintos esforços para materializar cada vez mais os benefícios do paradigma SDN nas infraestruturas de redes de próxima geração [Neto et al. 2021].

À medida que SDN está se tornando popular, esse paradigma gera novas preocupações de segurança que devem ser mitigadas [Scott-Hayward et al. 2015]. Como exemplo, destaca-se a grande preocupação a respeito do controlador, que por ter seu funcionamento baseado em uma arquitetura centralizada, acaba criando um problema conhecido como ponto único de falha (*single-point-of-failure*). Além disso, diversos trabalhos na literatura demonstram outras maneiras de comprometer redes SDN, como as técnicas apresentadas em [Shin and Gu 2013] e [Sonchack et al. 2016]. Particularmente, esses trabalhos abordam algumas variações de ataques de negação de serviço (DoS, do inglês *Denial of Service*), que tem como objetivo degradar ou interromper o funcionamento da rede

alvo a partir da inserção de tráfego malicioso. Esses novos ataques tem sido criados com o objetivo de indisponibilizar redes SDN em específico, explorando suas vulnerabilidades. Além disso, com a popularização de tecnologias emergentes, como a Internet das Coisas (IoT, do inglês *Internet of Things*) [Silva et al. 2021], ataques DoS de maior escala, como os ataques de negação de serviço distribuídos (DDoS, do inglês *Distributed Denial of Service*) também vem comprometendo serviços de redes SDN [Dantas Silva et al. 2020]. Relatórios[1] recentes de segurança especializados mostram que ataques do tipo DDoS continuam a ameaçar as organizações em todo o mundo, resultando em 5,4 milhões de ataques deste tipo no primeiro semestre de 2021, demonstrando um crescimento de 11% ano a ano.

Durante a preparação de ataques DoS contra redes SDN, os atacantes inicialmente precisam realizar uma etapa imprescindível para o sucesso da atividade maliciosa: o *scanning* de uma rede consiste em adquirir informações essenciais para o ataque e, nesse contexto, se a rede alvo utiliza protocolos SDN ou não, bem como alguns parâmetros de configurações das regras de fluxo configuradas na rede alvo. Por exemplo, inferindo informações sobre as regras de fluxos instaladas na rede, como o tempo de expiração das regras (*timeout*) [Sonchack et al. 2016]. De posse de tais informações, atacantes podem prosseguir com ataques mais danosos [Ma et al. 2014, Pascoal et al. 2017, Pascoal et al. 2020]. Por exemplo, o ataque *Slow-TCAM* envia novos pacotes maliciosos para a rede a fim de manter uma grande quantidade de regras maliciosas instaladas simultaneamente, e como resultado alcança a saturação do *switch* SDN. Portanto, é de suma importância evitar que ataques *scanning* sejam bem sucedidos.

Como resposta a esses ataques, diferentes técnicas de defesa vem sendo oferecidas, dentre as quais destaca-se o *Moving Target Defense* (MTD). MTD é uma nova estratégia de defesa que tem como objetivo mudar parâmetros e/ou características de sistemas de forma dinâmica. Como resultado, obtém-se um ambiente dinâmico que dificulta a detecção de informações do sistema alvo e, consequentemente, a execução de ataques por parte usuários maliciosos. O principal objetivo ao utilizar MTD é reduzir as janelas de oportunidades que atacantes podem detectar para atacar sistemas e ou redes de computadores. Assim, cria-se a oportunidade de diversificação constante do alvo com o intuito de reduzir as chances de exposição às suas vulnerabilidades, o que irá dificultar diversos tipos ataques. Atualmente, diversas estratégias MTD vêm sendo adotadas para mitigação de ataques do tipo *scanning* contra redes SDN, tais como as abordagens apresentadas em [Ma et al. 2014] e [Yuwen et al. 2016]. Devido ao dinamismo de sua operação, MTD requer redes com alta capacidade de programabilidade. Portanto, a sua adoção é fortemente relacionada com a utilização de redes SDN, que facilita a aplicação e mudanças de característica da rede de forma dinâmica.

Neste trabalho, é realizado um estudo aprofundado sobre técnicas utilizadas para a mitigação de ataques *scanning* contra redes SDN. Após ampla revisão da literatura relacionada identificaram-se as propostas apresentadas em [Ma et al. 2014], [Yuwen et al. 2016] e [Hou et al. 2020] como as mais relevantes nesta temática. As defesas mencionadas nos trabalhos analisados baseiam-se na adição de latência a pacotes que trafegam através da rede para assim evitar que atacantes infiram que a rede alvo utiliza SDN, bem como os valores de *timeouts* das regras. Cada uma das propostas estu-

[1] https://www.netscout.com/threatreport

dadas possui suas particularidades, que estendem-se desde a capacidade ou não de identificar uma possível ameaça na rede, até o método de aplicação da latência (*delay*) nos pacotes. Porém, apesar de efetivas, essas estratégias podem acabar degradando a Qualidade de Serviço (QoS, do inglês *Quality of Service*) da rede, pois a adição de latência nos pacotes para camuflar o tempo de expiração das regras acaba aumentando o tempo de resposta médio da rede. Esta confirmação pode ser vista nos resultados dos experimentos realizados na presente proposta, que demonstram que a estratégia introduzida em [Ma et al. 2014] apresenta uma latência média em torno de 20.68 ms, enquanto a rede normal apresenta uma resposta média na ordem dos 0.13 ms. Uma variação como esta impacta diretamente a QoS da rede, e pode desencorajar o seu uso para alguns tipos de serviços (p. ex. aplicações da vertical de e-Health do 5G) [Silva et al. 2022].

Diante desta lacuna, este trabalho introduz o *MTD Adaptive Delay System* (MADS), uma solução adaptativa para proteção de redes SDN apoiada na abordagem MTD. Enquanto a estratégia de [Ma et al. 2014] aplica atraso nos pacotes de forma contínua, MADS aplica a latência nos pacotes de forma adaptativa (pontual), ou seja, somente em situações em que a rede é realmente impactada pelo ataque. Assim, evita-se que pacotes legítimos não sofram um tempo de resposta maior, bem como que a latência adicionada pela MTD não impacte a rede fortemente. A partir de estudos realizados para modelar ataques do tipo *scanning*, podemos detectar limiares para identificar a existência e o impacto do ataque na rede em um dado momento. Esses limiares são então utilizados por MADS para ativar a estratégia de atraso de pacotes, que por sua vez inibem que atacantes possam inferir corretamente o tipo da rede utilizada (neste caso, SDN) e os valores de *timeouts* das regras de fluxos, não sendo, portanto, capazes de executar ataques DoS com sucesso.

Através dos resultados obtidos a partir de um conjunto de avaliações que reproduzem as condições de experimentação de trabalhos anteriores, é possível constatar que MADS é capaz de manter a eficiência da estratégia MTD para mitigar os ataques. Além disso, os efeitos da degradação do QoS observados na operacionalização de MADS são mais amenos quando comparada com o estado da arte. Os resultados também demonstraram uma melhora significativa em outros parâmetros da rede como a vazão e redução de pacotes do tipo *Bad TCP*.

O restante deste artigo encontra-se organizado da seguinte forma: a Seção 2 mostra ataques que podem comprometer redes SDN, assim como alguns métodos para mitigação; a Seção 3 expõe os trabalhos relacionados ao tema abordado; a Seção 4 detalha o funcionamento da abordagem empregada no MADS, método de defesa proposto por este trabalho; a Seção 5 exibe os experimentos realizados e os resultados; por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Ataques à Segurança das Infraestruturas SDN

Esta seção descreve o comportamento de ataques contra redes SDN e, em seguida, introduz a técnica MTD, apresentando suas vantagens e desvantagens quando sendo utilizada para defender redes SDN.

2.1. Ataques que exploram vulnerabilidade de redes SDN

Em redes SDN, é possível perceber uma diferença no tempo de resposta para novos pacotes em relação à pacotes que já possuem uma regra de fluxo instalada na tabela de

fluxos (*Flow Table*) de *switches*. Essa característica dá-se pelo fato de que dado que um pacote ainda não possua uma regra na tabela de fluxo, faz-se necessário ao *switch* enviar uma requisição (*Packet-In*) para o controlador da rede, requisitando qual tomada de ação deve ser executada para o pacote em questão. Neste cenário, é perceptível que a latência do primeiro pacote (que não possui uma regra correspondente) é maior que a latência dos demais pacotes (que possuem regras instaladas). Este tipo de comportamento é bastante explorado em forma de ataques *scanning*, e permite que atacantes reconheçam se a rede alvo utiliza a arquitetura SDN somente analisando os tempos de resposta da rede [Sonchack et al. 2016, Hou et al. 2020, Ma et al. 2014].

A partir da execução de ataques *scanning*, outros ataques mais danosos podem ser realizados, tal como o *Slow TCAM Exhaustion* [Pascoal et al. 2017, Pascoal et al. 2020], que é capaz de saturar a tabela de fluxos do *switch* alvo de maneira silenciosa. Portanto, proteger redes primordialmente contra ataques *scanning* é de suma importância para evitar a susceptibilidade de novos ataques.

2.2. Proteção de redes SDN utilizando MTD

Técnicas de MTD podem ser usadas para mitigar e evitar diferentes tipos de ataques, desde *scanning* à ataques DDoS. [Zhuang et al. 2014] mostrou que, no caso de mecanismos baseados em MTD para proteção de redes, as chances de realizar um ataque com sucesso diminuem com o tempo. Em contraste, em redes estáticas, onde o invasor é capaz de fazer inferências lógicas da rede com menos esforço, as chances de um ataque bem-sucedido ocorrer são bem maiores. A razão para isso é que as redes estáticas mantêm suas configurações por longos períodos até a intervenção de um agente externo (p. ex. administradores de rede). Portanto, com o suporte das facilidades SDN e mecanismos MTD, essas configurações podem ser atualizadas de forma dinâmica, não precisando mais de um agente externo.

O objetivo principal de um mecanismo MTD é ter a vantagem do tempo contra o invasor [Zhuang et al. 2014], deixando a própria abordagem (ou outros mecanismos) capaz de iniciar contramedidas prontamente e sem interferir na experiência de usuários legítimos. Os estudos de [Okhravi et al. 2013] e [Lei et al. 2018] sugerem que os principais motivos para a adoção recente de técnicas de MTD são:

1. Permitir que os mecanismos limitem e reduzam dinamicamente a superfície de ataque, de modo que os atacantes necessitem realizar um esforço maior para lançar um ataque bem-sucedido. Isso também significa que os mecanismos de proteção têm mais tempo para implantar contramedidas;
2. Como resultado das constantes mudanças nas configurações da rede e do considerável custo computacional necessário para realizar um ataque bem-sucedido, há uma necessidade reduzida de mecanismos para detecção de ameaças;
3. À medida que mais dispositivos são adicionados à rede, haverá uma maior assimetria entre os atacantes e os defensores.

A principal vantagem ao se aplicar técnicas de MTD em cenários SDN contra ataques DDoS é utilizar a alta programabilidade da rede e o poder do controlador sob ela. Esta abordagem permite que mudanças rápidas e dinâmicas ocorram na topologia da rede e nos parâmetros de configuração dos serviços (p. ex. portas abertas, endereços IP), ao contrário das redes convencionais/estáticas. Consequentemente, tentativas de ataques

à alvos dentro da rede encontrarão maiores dificuldades, além de exigir mais recursos computacionais e um maior esforço do invasor.

3. Trabalhos Relacionados

Diversos ataques que exploram vulnerabilidades do paradigma SDN vem sendo apresentados. [Shin and Gu 2013] introduz um método de *scanning* de redes SDN baseados na diferença de latência dos pacotes que possuem e não possuem regras na tabela de fluxo dos *switches*. Já [Klöti et al. 2013] propõe uma ferramenta capaz de revelar informações sobre redes SDN através da análise da latência dos pacotes. De forma semelhante, [Sonchack et al. 2016] introduz um método de inferência de configurações de rede SDN baseado na criação de dois fluxos (*timing probes* e *testing packets*), permitindo que um atacante aprenda mais sobre a rede.

Ataques do tipo *scanning* são de suma importância para a realização de ataques mais engenhosos contra redes SDN. Uma vez que características da rede são adquiridas, ataques como DoS podem ser executados com sucesso. Outras formas são os ataques do tipo *low-rate* e *flooding*, capazes de indisponibilizar a rede SDN alvo de forma silenciosa e sem a geração de tráfego, o que acaba dificultando sua mitigação. Tais razões revelam a importância de realizar a eliminação do primeiro vetor de vulnerabilidade em redes SDN, no caso, mitigando ataques do tipo *scanning*.

Para a mitigação de ataques *scanning*, [Yuwen et al. 2016] apresenta uma proposta que tem como objetivo utilizar uma estratégia MTD chamada *Probability-based Delay*. A estratégia utilizada atrasa pacotes aleatoriamente em uma probabilidade específica de forma que a latência é adicionada apenas em alguns pacotes. Dessa forma, a performance da rede é levada em consideração. Além disso, há duas abordagens que podem ser utilizadas para aplicação da latência. Na primeira é utilizado a ação OFPAT_EXPERIMENTER do protocolo *OpenFlow*, que atrasa os pacotes que correspondem à uma regra específica. A desvantagem é que, para implementação de uma ação OFPAT_EXPERIMENTER, é necessário que haja a modificação no protocolo e no *switch*. A segunda abordagem possui dois cenários: (i) o pacote é enviado ao controlador e depois ao destino, causando o atraso no pacote de forma indireta; e (ii) o pacote é enviado ao controlador, que por sua vez aplica um atraso de tempo antes de ser manipulado e envia a resposta. A desvantagem dessa abordagem é que todos os pacotes que precisam de latência aplicada necessitam passar pelo controlador, podendo ocasionar uma sobrecarga, dependendo do tamanho da rede.

Similarmente, [Ma et al. 2014] apresenta um mecanismo baseado em MTD para proteger a infraestrutura SDN, especialmente o controlador, contra um tipo especial de ataque de *flooding*. Neste tipo de ataque, conhecido como *blind DDoS*, o invasor não conhece nenhuma informação sobre o controlador (p. ex. endereço IP), mas ainda é capaz de interromper o sistema do controlador de destino. Como o uso de um único controlador pode levar à um ponto único de falha, a estratégia de mitigação depende de um conjunto de vários controladores, implantados sob demanda dependendo da gravidade do ataque. Além disso, os autores sugerem o uso de um mecanismo para atraso de pacotes para evitar que o atacante faça uma inferência lógica de que uma rede SDN está sendo usada. Desse modo, a proposta busca igualar a latência dos demais pacotes (*T2*) à latência do primeiro pacote (*T1*), de forma a inserir latência aos pacotes *T2*.

O mecanismo da inserção de latência aos pacotes *T2* em [Ma et al. 2014] de forma contínua é funcional quanto ao anti-reconhecimento de que o usuário está em uma rede SDN. Porém, também é perceptível que essa estratégia irá degradar a QoS da rede, já que haverá um aumento significativo no atraso de todos os pacotes que trafegam nela. Seguindo essa lógica, seria mais viável a adoção de uma estratégia que aplicasse a latência adicional aos pacotes *T2* apenas em determinados momentos (p. ex. ao detectar que está havendo *scanning* na rede). Sendo assim, a QoS da rede não seria tão impactada quanto a proposta apresentada em [Ma et al. 2014].

O trabalho de [Hou et al. 2020] percebe que inserir latência à todos os pacotes que trafegam pela rede, conforme apresentado na proposta [Ma et al. 2014], afeta gravemente a performance do sistema. Portanto, os autores oferecem uma solução probabilística para atrasar o tempo de resposta de pacotes de forma seletiva. A defesa adia a instalação de algumas regras para ocultar a diferença entre os tempos dos pacotes de um mesmo fluxo. Portanto, o trabalho propõe um mecanismo de defesa leve para defender contra esses tipos de ataques de *scanning*, onde sua ideia principal é randomizar a diferença de tempo de transmissão entre diferentes pacotes em um dado fluxo e manter o mínimo possível de informações de *status* dos demais fluxos.

A Tabela 1 sumariza as contribuições de cada uma dos trabalhos relacionados analisados, e destaca suas principais limitações.

Proposta	**Estratégia principal**	**Abordagem**	**Técnica de melhora da QoS**	**Limitações**
[Ma et al. 2014]	Atraso contínuo dos pacotes.	Adição de latência nas portas dos *switches*.	Não há	Degradação da performance da rede de forma contínua e seleção do valor de latência é realizada considerando estado anterior da rede (tempo de resposta).
[Yuwen et al. 2016]	Atraso seletivo dos pacotes, baseado em probabilidade.	Envio dos pacotes selecionados para o controlador.	Utiliza probabilidade para selecionar os pacotes que serão atrasados.	Sobrecarga no controlador dependendo do tamanho da rede.
[Hou et al. 2020]	Atraso seletivo dos pacotes, baseado no IP e MAC do dispositivo de origem.	Envio dos pacotes selecionados para o controlador.	Atrasa apenas os pacotes gerados por possíveis dispositivos maliciosos.	Sobrecarga no controlador dependendo da quantidade de possíveis ameaças.
MADS (esta proposta)	Atraso dinâmico dos pacotes, baseado na identificação do ataque.	Adição de latência nas portas dos *switches*.	Não aplica latência na ausência de um ataque.	A seleção do valor de latência é realizada considerando estado anterior da rede (tempo de resposta).

Tabela 1. Comparação entre os trabalhos relacionados e a nossa proposta MADS

Conforme comparativo apresentado na Tabela 1, o método de aplicação de latência é diferente em cada proposta citada. Em [Ma et al. 2014] a latência é aplicada diretamente nas portas dos *switches*, de maneira contínua (sem interrupção). Os trabalhos de [Yuwen et al. 2016] e [Hou et al. 2020] possui uma similaridade em suas estratégias: am-

bos enviam os pacotes para o controlador a fim de causar atraso adicional. Porém, enquanto [Yuwen et al. 2016] utiliza uma técnica baseada em probabilidade com o objetivo de selecionar os pacotes que serão enviados ao controlador, [Hou et al. 2020] constrói uma tabela com parâmetros específicos para determinar quais dispositivos conectados à rede podem ser uma ameaça, e assim atrasar os pacotes enviados por estes. A abordagem introduzida por MADS é semelhante à de [Ma et al. 2014]. Entretanto, a técnica empregada por MADS utiliza um método para remoção da latência em determinados momentos, com o intuito de tornar a rede mais eficiente. A próxima seção apresenta em detalhes a proposta de proteção adaptativa para proteção de redes SDN utilizando *Moving Target Defense*.

4. MTD Adaptive Delay System – MADS

MADS avança o estado da arte ao aplicar configurações de latência na rede de forma adaptativa, ao invés de forma contínua (como exaustivamente realizado pelos trabalhos anteriores). Através dessa abordagem, é possível minimizar os possíveis impactos negativos que uma técnica MTD baseada em adição de latência pode acarretar no que diz respeito à QoS da rede. MADS fornece suporte à configuração adaptativa de latência como sendo um bloco funcional inserido no plano de controle da rede. Além disso, as capacidades adaptativas são suportadas pelo monitoramento do estado da rede, que aciona o módulo de adição de latência somente ao identificar que a rede está sob ataque.

4.1. Funcionamento da solução

A Figura 1 materializa o diagrama de sequência da proposta, apresentando as principais mensagens trocadas entre um atacante, os *switches* e o controlador SDN enquanto a rede está sofrendo um ataque de *scanning*.

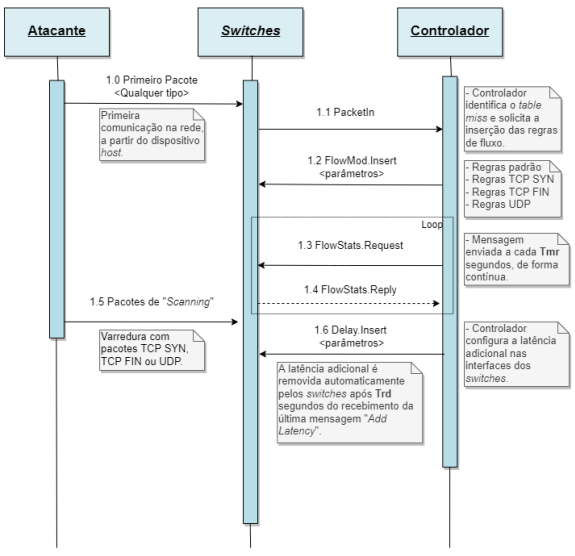

Figura 1. Comunicação envolvida na arquitetura proposta

Quando o *atacante* ingressa na rede SDN e envia seu primeiro pacote, o *switch* em que ele está diretamente conectado aciona o controlador através de uma mensagem *OpenFlow* do tipo `Packet-In` (etapas 1.0 e 1.1). Após isso, o controlador realiza a

inserção de regras nos *switches* (regras TCP SYN, TCP FIN e UDP) através de mensagens `FlowMod.Insert` (etapa 1.2). Uma vez que as regras estejam inseridas, o controlador SDN realiza o monitoramento do estado dessas regras, em intervalos de segundos definido pela variável **Tmr**, através de mensagens `FlowStatsRequests` enviadas para os *switches* (etapa 1.3 e 1.4). Havendo um *scanning* na rede (etapa 1.5), o contador de *bytes* trafegados de alguma(s) dessas regras aumentará de forma que não condiz ao comportamento normal da rede (os parâmetros para a ativação da estratégia são explicados na Seção 4.2). Portanto, o *switch* é instruído a adicionar configurações de latência em suas interfaces através de mensagens `Delay.Insert`. Assim, por meio dessas mensagens, a latência é aplicada às interfaces do *switch*. A remoção da latência é realizada automaticamente pelos *switches* ao não receberem mais mensagens `Delay.Insert` do controlador durante o tempo (em segundos) definido pela variável **Trd**.

Enquanto a rede está sob ataque e a latência está aplicada nas interfaces do *switch*, o controlador pode continuar a enviar e a receber mensagens do tipo `FlowStatsRequests` e `FlowStatsReply`, respectivamente. Isso pode causar o envio de outras mensagens `Delay.Insert`, se ainda estiver ocorrendo um *scanning*. Quando isso acontece, o *switch* simplesmente ignora as mensagens `Delay.Insert` e mantém a latência que já está sendo aplicada.

4.2. Ativação da latência

Para analisar o comportamento do ataque *scanning*, estudamos o ataque com o uso do *Nmap*[2], uma ferramenta bastante conhecida para testes de intrusão e varredura de vulnerabilidade em redes de computadores [Kelly et al. 2019]. Em conjunto com o *Nmap*, utilizamos a ferramenta *Wireshark*[3] para adquirir o tempo de resposta dos pacotes enviados pelo *Nmap*, assim, criando meios para colher informações e inferir se a rede alvo está utilizando arquitetura SDN ou não (tendo como base a intuição explicada na Seção 2.1).

Dependendo da disponibilidade e velocidade da rede, o uso padrão do *Nmap* poderá varrer milhares de portas por segundo [Zhang et al. 2021]. Porém, esse comportamento pode ser facilmente detectado por *Firewalls* da rede. Com o intuito de burlar as regras de *Firewalls*, o *Nmap* pode ser usado com a opção de paralelismo, que ajusta a quantidade máxima de pacotes de sondagem que serão enviados em paralelo ao alvo. Portanto, adotamos esse comportamento para analisar e definir a quantidade de pacotes que determinarão o início do acionamento de latência em pacotes na rede.

Para entender melhor o comportamento do *Nmap* neste cenário, um conjunto de experimentos foram conduzidos considerando uma rede com 8 dispositivos alvo, o que resultou em um total de 8000 portas escaneadas (*Nmap* realiza uma varredura de 1000 portas por alvo). Além disso, aplicamos a opção de ajuste de paralelismo para enviar 1 pacote de sondagem por vez, ou seja, os pacotes foram enviados em série. Seguindo a metodologia descrita, obteve-se a duração média de acordo com os tipos de pacotes enviados, a saber:

- TCP SYN: 676.23 segundos
- TCP FIN: 678.35 segundos

[2]https://nmap.org/

[3]https://www.wireshark.org/

- UDP: 1106.12 segundos

Dividindo a quantidade total de portas escaneadas (8000 portas) pela duração média de cada teste obtemos aproximadamente as seguintes taxas de envio: 12 portas/segundo para TCP SYN e TCP FIN e 7 portas/segundo para UDP.

Como resultado, de acordo com o tipo de pacote, podemos definir a quantidade máxima de pacotes que a defesa MADS aceita antes de começar a aplicar latência nos mesmos. Dado que MADS realiza o monitoramento de *status* dos fluxos a cada **Tmr** (segundos), o valor de **Trd** é o produto entre a taxa de envio por porta multiplicado pelo valor do **Tmr**.

Vamos considerar, por exemplo, $Tmr = 10$. Então, o valor de **Trd** a ser configurado em MADS (supondo uma topologia com 8 alvos) deverá ser: $Trd = 120$ (12x10) para TCP SYN e TCP FIN, e $Trd = 70$ para UDP. Assim, se o contador de pacotes presente em cada regra inserida no *switch* ultrapassar os valores definidos por **Trd** entre cada intervalo de mensagens *FlowStatsReply* (definido por **Tmr**), o controlador irá adicionar latência (emitindo uma mensagem `Delay.Insert` para o *switch* para ativar a latência).

4.3. Obtenção da latência

O modelo para determinar os valores de latência são definidos conforme a equação (1). Este modelo segue a mesma técnica utilizada por [Ma et al. 2014], onde foram utilizados pacotes ICMP para definir valores para os *FirstPacket* e *LastPacket*, representados por t_1 e t_2, caracterizando a latência dos pacotes iniciais (os que passam pelo controlador) e dos demais pacotes, respectivamente. Uma lista representada por *Dt* é construída a partir da subtrações dos valores de t_1 por t_2.

$$Dt = \{dt(i)|dt(i) = t_1(i) - t_2(i), i > 0\} \tag{1}$$

A equação (2) seleciona um valor, de forma randômica, que deve estar entre o mínimo e o máximo presentes na lista *Dt*. Este valor é somado ao t_2 para obtenção de *T2*, que representa a latência que é adicionada às interfaces do *switch* através da mensagem `Delay.Insert` quando detectado que está havendo um *scanning* na rede.

$$T2 = \{t'_2(i)|t'_2(i) = t_2(i) + Random[Min(Dt), Max(Dt)], i > 0\} \tag{2}$$

5. Experimentação e Resultados

Os experimentos foram realizados em um computador *desktop* com processador Intel Core i7-3770 e 12 GB RAM. A execução se deu por meio de dois *containers Docker*[4] utilizando o sistema operacional Debian na versão 9.13. Um dos *containers* executa a ferramenta *Mininet*[5], responsável pela emulação da topologia de rede utilizada nos experimentos. O segundo *container* é responsável por hospedar o controlador SDN, que implementa as aplicações de defesa para cada experimento. As aplicações foram implementadas utilizando a linguagem Python por meio do controlador SDN *Ryu*[6].

[4]https://www.docker.com/

[5]http://mininet.org/

[6]https://ryu-sdn.org/

Todos os pacotes trafegados na rede foram coletados através da ferramenta *tcpdump*[7] utilizando a interface do *switch* na qual o servidor *web* estava conectado para a leitura do tráfego. Além disso, a ferramenta *Wireshark* foi empregada para realizar a análise dos pacotes referente à comunicação do tráfego cliente com o servidor *web* a fim de averiguar seu desempenho.

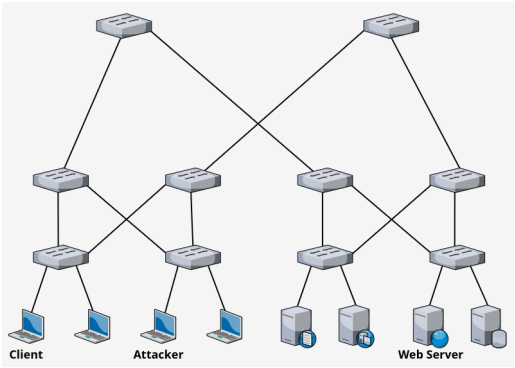

Figura 2. Topologia utilizada nos experimentos

A topologia utilizada, conforme a Figura 2, é composta por dez *switches* e oito (8) dispositivos *hosts*. Os dispositivos finais utilizados nos experimentos foram: (i) um servidor *web* (*Web Server*); (ii) um dispositivo reproduzindo tráfego cliente (*Client*); e (iii) um dispositivo reproduzindo tráfego atacante (*Attacker*). O objetivo da experimentação foi comparar o impacto gerado pelas defesas MTD no desempenho da rede. Portanto, foram comparadas as performances das propostas [Ma et al. 2014], [Hou et al. 2020] e MADS. Diversos parâmetros da rede foram coletados durante os experimentos, a saber: *RTT*, vazão e *Bad TCP*. A proposta de [Yuwen et al. 2016] não foi considerada devido a falta de informação sobre o algoritmo de probabilidade utilizado para detectar a ocorrência de ataques na rede, portanto, impossibilitando uma comparação justa com as demais propostas.

Cada experimento teve uma duração de 4.700 segundos, ou seja, cerca de 78 minutos, e adotou a seguinte abordagem: (i) o cliente gerou tráfego HTTP ao servidor *web* em intervalos de 1 segundo; (ii) o atacante executou ataque *scanning*, enviando requisições à rede em busca de portas TCP abertas. Em todos os testes foram realizadas três varreduras por parte do usuário malicioso (ou seja, rajada de tráfego *Nmap*, com duração de aproximadamente 800 segundos – tempo médio de 14 minutos), cada. As varreduras com *Nmap* tiveram a seguinte configuração: `--max-parallelism 1 -p 1-65500`. O parâmetro `--max-parallelism 1` ajustou o número máximo de envio de pacotes em paralelo para 1, de modo a considerar que um atacante poderia utilizar esta técnica para tentar burlar o controlador a respeito do *scanning*. Já o parâmetro `-p 1-65500` habilita o *Nmap* para varrer até 65500 portas da rede alvo. Porém, durante cada rajada de ataque, que dura aproximadamente 14 minutos, identificou-se que a ferramenta conseguiu varrer em média 14150 portas. A partir da análise do tráfego gerado pelo *Nmap* com a configuração descrita acima, foi identificado um total de 40284 pacotes gerados durante cada experimento, em média.

[7]https://www.tcpdump.org/

Para os experimentos utilizando a proposta de [Hou et al. 2020], a variável **PACT_MAX** foi definida com valor de 40284. Nos experimentos envolvendo o MADS, considerou-se os valores de tempo das variáveis **Tmr** e **Trd** de 10 e 120 segundos, respectivamente. Adotamos $Tmr = 10$ por se tratar de um valor comum de *timeout* para as regras de fluxos em redes SDN [Zarek et al. 2012]. Em relação ao parâmetro **Trd**, esse valor foi escolhido baseado no comportamento do tráfego atacante gerado durantes os experimentos. Ambos os parâmetros podem ser configurados conforme o estado, topologia, tráfego esperado e natureza da rede a ser defendida. As subseções seguintes descrevem os resultados dos experimentos de acordo com o tipo de parâmetro avaliado.

5.1. Latência (RTT)

A Figura 3 apresenta a comparação do RTT entre os pacotes envolvidos na comunicação entre cliente e servidor durante os testes. No experimento da proposta [Ma et al. 2014], durante toda a comunicação a latência dos pacotes permaneceu em torno de 20 ms, devido ao fato de que essa solução aplica a latência nos pacotes de forma contínua. Já as defesas aplicadas por [Hou et al. 2020] e por MADS apresentam um RTT semelhante, permanecendo abaixo de 0.2ms na maior parte do tempo e atingindo uma latência aproximadamente 99.4% menor em relação a proposta de [Ma et al. 2014].

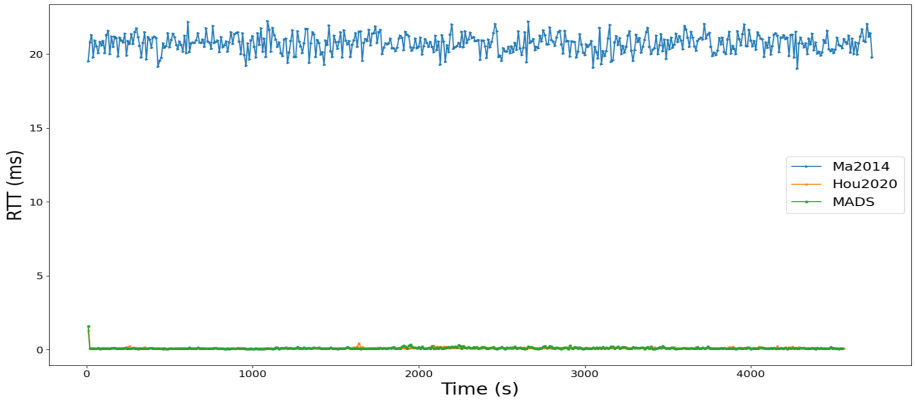

Figura 3. Comparação entre a taxa de RTT.

5.2. Vazão

A Figura 4 ilustra a comparação entre a taxa de vazão coletada durante os testes. Os resultados destes experimentos para as defesas de [Hou et al. 2020] e MADS mantiveram-se parecidos. É possível perceber que a taxa de ambos se manteve entre 750 Kb/s e 1750 Kb/s na maior parte do tempo, onde MADS atingiu uma vazão 4.87% maior quando comparada a solução concorrente. Já para [Ma et al. 2014], foi obtida uma taxa de vazão 27 vezes menor quando comparada às duas propostas citadas anteriormente, permanecendo abaixo de 60 Kb/s em todo o experimento.

5.3. Bad TCP

A Figura 5 apresenta a comparação em relação aos pacotes *Bad TCP* gerados durante os experimentos. A existência de pacotes *Bad TCP* indica a necessidade de retransmissão de pacotes entre o servidor e o cliente HTTP devido à falhas de comunicação. Portanto, a quantidade de pacotes *Bad TCP* ou *TCP Error* influencia diretamente na QoS da rede,

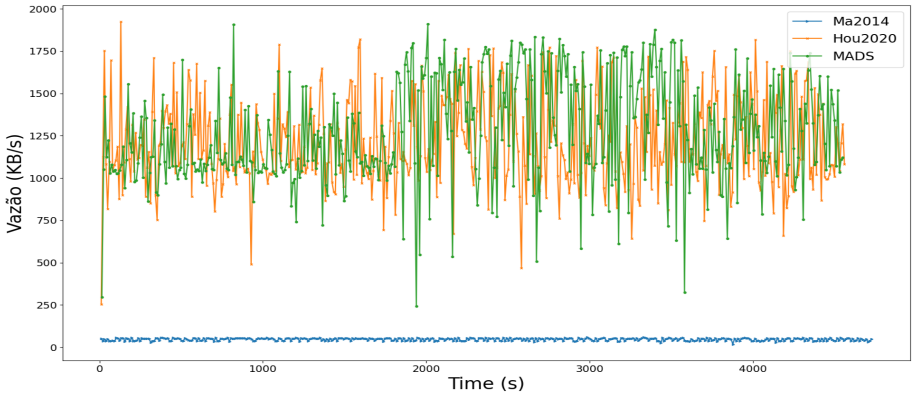

Figura 4. Comparação entre a taxa de vazão.

	Ma et al. 2014		Hou et al. 2020		MADS	
Parâmetro	**Média**	**Desvio Padrão**	**Média**	**Desvio Padrão**	**Média**	**Desvio Padrão**
RTT	20,72 ms	0,61 ms	0,12 ms	0,06 ms	0,11 ms	0,08 ms
Vazão	47,05 KB/s	7,64 KB/s	1215,19 KB/s	266,48 KB/s	1274,44 KB/s	306,67 KB/s
Bad TCP	17,15	4,14	Não Aplicável	Não Aplicável	Não Aplicável	Não Aplicável

Tabela 2. Comparação da média e desvio padrão dos parâmetros coletados

pois a retransmissão proporcionada pelo protocolo TCP gera atraso na rede. Para os experimentos com a defesa [Hou et al. 2020] e MADS, não foram percebidos pacotes deste tipo durante os testes, ou seja, nenhum pacote TCP precisou ser retransmitido por causa de erros ou perdas. Nestes testes, mesmo ocorrendo um ataque de *scanning* na rede, não houve degradação da comunicação entre cliente e servidor já que ambos não foram afetados pelo latência adicional que foi inserida aos pacotes gerados pelo atacante. Já para a defesa de [Ma et al. 2014], foram identificados pacotes *TCP Error* durante todo o experimento, variando entre 10 e 30 pacotes *Bad TCP* a cada 10 segundos.

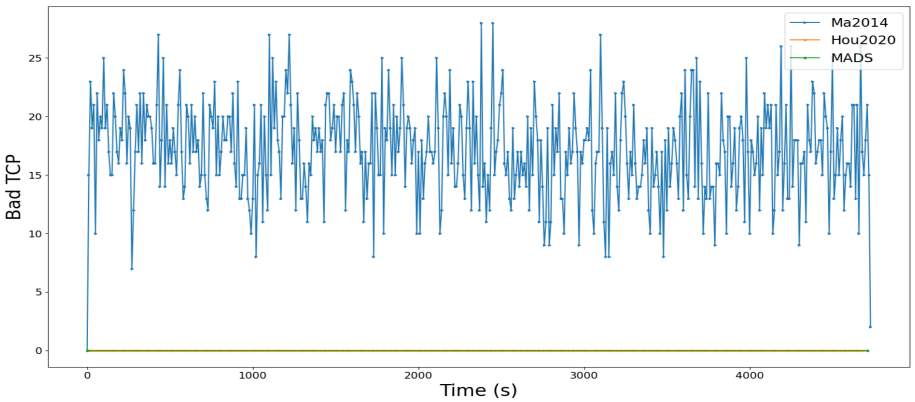

Figura 5. Comparação entre a taxa de Bad TCP

5.4. Comparação e Discussão dos Resultados

A Tabela 2 apresenta a comparação de todas as métricas analisadas nos experimentos, representadas em forma de média e desvio padrão. A partir destes resultados ficou perceptível que a proposta [Ma et al. 2014] teve uma diferença de desempenho maior em relação às outras duas, devido ao fato de que ela não propõe-se a identificar qualquer possível atacante na rede, aplicando a latência à todos os dispositivos presentes na rede.

Para todos os parâmetros, as propostas [Hou et al. 2020] e MADS tiveram seus resultados muito semelhantes. Neste caso, a latência adicional é inserida apenas para o tráfego gerado pelo atacante, não prejudicando a comunicação do tráfego entre o cliente e o servidor. No entanto, o método de aplicação de latência de [Hou et al. 2020], onde os pacotes são enviados ao controlador, pode representar um problema para a rede. Tal problema se dá pela sobrecarga de processamento de pacotes no controlador e, consequentemente, queda no desempenho da comunicação entre os outros dispositivos. De modo distinto, MADS insere a latência diretamente na porta do *switch* em que o dispositivo atacante está conectado, não afetando a comunicação entre quaisquer outros dispositivos com o controlador.

6. Conclusões e Trabalhos Futuros

É evidente que técnicas de MTD estão sendo cada vez mais utilizadas, principalmente para o combate à ataques de DoS. Neste trabalho, consideramos defesas MTD para proteção contra ataques *scanning* em redes SDN. A principal forma de mitigação contra esse ataque é garantir que o tempo de resposta observado por atacantes sejam similares, de forma a inibir a possibilidade de inferência de conhecimento do uso de SDN. Usualmente, MTD alcança esse objetivo a partir da aplicação de latência nos pacotes a fim de uniformizar o tempo de resposta. Apesar de eficiente, neste trabalho mostramos que a utilização de tais mecanismos pode causar degradações na performance da rede, demandando a adoção de técnicas especializadas para lidar com tal excepcionalidade.

Para evitar tal degradação na rede, oferecemos uma nova forma de ativação da latência. Particularmente, utilizamos uma estratégia adaptativa que aciona a latência somente na presença de ataque, ao contrário de estratégias existentes que aplica a latência de forma contínua. Através de um amplo conjunto de experimentações, demonstramos que nossa estratégia apresenta o mesmo nível de proteção contra o ataque quando comparada com o estado da arte, porém, afetando parâmetros de performance da rede protegida de forma mais amena. Em termos gerais, MADS destaca-se em termos de desempenho da rede, diminuindo consideravelmente a latência e, consequentemente, aumentando a vazão, bem como eliminando a existência de pacotes *Bad TCP*. Além disso, MADS avança o estado da arte ao promover uma abordagem de proteção eficaz sem impor degradação ao controlador SDN (comportamento observado nas demais soluções), pois precisa processar uma grande quantidade de pacotes e, desta maneira, diminui sua capacidade de resposta.

Em trabalhos futuros pretendemos oferecer novas métricas para a aquisição de valores de latência, levando em consideração parâmetros adicionais da rede protegida, como por exemplo, a quantidade de saltos e o tamanho da topologia. Como mecanismo auxiliar, enxergamos oportunidades de acoplar um módulo baseado em inteligência artificial para melhor estipular o valor de latência a ser aplicado de acordo com o comportamento e nível de segurança da rede.

Referências

Dantas Silva, F. S., Silva, E., Neto, E. P., Lemos, M., Venancio Neto, A. J., and Esposito, F. (2020). A taxonomy of ddos attack mitigation approaches featured by sdn technologies in iot scenarios. *Sensors*, 20(11).

Hou, J., Zhang, M., Zhang, Z., Shi, W., Qin, B., and Liang, B. (2020). On the fine-grained fingerprinting threat to software-defined networks. *Future Generation Computer Systems*, 107:485–497.

Kelly, J., DeLaus, M., Hemberg, E., and O'Reilly, U.-M. (2019). Adversarially adapting deceptive views and reconnaissance scans on a software defined network. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 49–54.

Klöti, R., Kotronis, V., and Smith, P. (2013). Openflow: A security analysis. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE.

Lei, C., Zhang, H.-Q., Tan, J.-L., Zhang, Y.-C., and Liu, X.-H. (2018). Moving target defense techniques: A survey. *Security and Communication Networks*, 2018.

Ma, D., Xu, Z., and Lin, D. (2014). Defending blind ddos attack on sdn based on moving target defense. In *International Conference on Security and Privacy in Communication Networks*, pages 463–480. Springer.

Neto, E. P., Silva, F. S. D., Schneider, L. M., Neto, A. V., and Immich, R. (2021). Seamless mano of multi-vendor sdn controllers across federated multi-domains. *Computer Networks*, 186:107752.

Okhravi, H., Hobson, T., Bigelow, D., and Streilein, W. (2013). Finding focus in the blur of moving-target techniques. *IEEE Security & Privacy*, 12(2):16–26.

Pascoal, T. A., Dantas, Y. G., Fonseca, I. E., and Nigam, V. (2017). Slowtcam xehaustion ddos attack. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 17–31. Springer.

Pascoal, T. A., Fonseca, I. E., and Nigam, V. (2020). Slow denial-of-service attacks on software defined networks. *Computer Networks*, page 107223.

Scott-Hayward, S., Natarajan, S., and Sezer, S. (2015). A survey of security in software defined networks. *IEEE Communications Surveys & Tutorials*, 18(1):623–654.

Shin, S. and Gu, G. (2013). Attacking software-defined networks: A first feasibility study. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 165–166.

Silva, F. S. D., Neto, E. P., Oliveira, H., Rosário, D., Cerqueira, E., Both, C., Zeadally, S., and Neto, A. V. (2021). A survey on long-range wide-area network technology optimizations. *IEEE Access*, 9:106079–106106.

Silva, F. S. D., Schneider, L., Rosário, D., and Neto, A. (2022). Network slicing mobility aware control to assist handover decisions on e-health 5g use cases. In *2022 International Wireless Communications and Mobile Computing (IWCMC)*, page 1.

Silva, J. B., Dantas Silva, F., Neto, E. P., Lemos, M., and Neto, A. (2020). Benchmarking of mainstream sdn controllers over open off-the-shelf software-switches. *Internet Technology Letters*, n/a(n/a):e152. e152 ITL-19-0098.R1.

Sonchack, J., Aviv, A. J., and Keller, E. (2016). Timing sdn control planes to infer network configurations. In *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 19–22.

Yuwen, H., Zhang, L., Wang, Z., and Kong, Y. (2016). Probability-based delay scheme for resisting sdn scanning. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 1096–1101. IEEE.

Zarek, A., Ganjali, Y., and Lie, D. (2012). Openflow timeouts demystified. *Univ. of Toronto, Toronto, Ontario, Canada*.

Zhang, Z., Towey, D., Ying, Z., Zhang, Y., and Zhou, Z. Q. (2021). Mt4ns: Metamorphic testing for network scanning. In *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*, pages 17–23.

Zhuang, R., DeLoach, S. A., and Ou, X. (2014). Towards a theory of moving target defense. In *Proceedings of the First ACM Workshop on Moving Target Defense*, pages 31–40.