SHADE: Uma estratégia seletiva para mitigar ataques DDoS na camada de aplicação em redes definidas por software

João H. G. Corrêa, Vivek Nigam, Moisés Ribeiro, Diego Mafioletti, Iguatemi E. Fonseca

***Resumo*— Atualmente, os ataques de negação de serviço (DoS –** *Denial of Service*) **, utilizam vulnerabilidades nos protocolos da camada de aplicação. Este trabalho propõe o uso de estratégias seletivas, utilizando as vantagens da separação do plano de dados e de controle, a visão global da rede, e a possibilidade de redirecionamento de fluxo que o paradigma das Redes Definidas por *Software* oferece, para mitigar ataques de DoS na camada de aplicação, do tipo *High-Rate*. Tal defesa, chamada SHADE, se mostrou eficiente através de emulações. Sem a defesa, um servidor *Web* só conseguiu servir 19% dos clientes, enquanto que com SHADE a disponibilidade subiu para 61%.**

Palavras-Chave*— *Software Defined Network*, Negação de Serviço, *Get Flooding

***Abstract*— Currently, denial of service attacks (DoS –** *Denial of Service*) **use vulnerabilities in the application layer protocols. This paper proposes the use of selective strategies, using the advantages of separation of data plane and control, global view of the network, and the ability to redirect flow into the paradigm of Software Defined Networks offers, to mitigate DoS attacks at the application layer, of type High-Rate. This defense, called SHADE, proved efficient through emulations. In our emulations without defense, a *Web* server could serve only 19% of client, whereas with SHADE availability rose to 61%.**

***Keywords*— Software Defined Network, Denial of Service, Get Flooding**

I. Introdução

Tradicionalmente, os ataques de negação de serviço distribuídos (DDoS – *Distributed Denial of Service*) são realizados nas camadas de Rede e/ou Transporte, tendo como características uma grande quantidade de pacotes e visando deixar um servidor, como um todo, indisponível, ou seja, todas as aplicações em execução param de responder a clientes legítimos. Atualmente, os ataques migraram para a camada de Aplicação, que visam indisponibilizar apenas um serviço dentro do servidor. Além disso, os ataques de Negação de Servido na camada de Aplicação (ADDoS – *Application layer Distributed Denial of Service*) tem como característica um tráfego semelhante aos de clientes honestos [1]. Dessa forma, as ferramentas tradicionais de análise do tráfego, não detectam esses tipo de ataque [2].

Dentro dos ataques ADDoS, o tipo *High-Rate* utiliza característica dos protocolos da camada de aplicação, para realizar uma inundação de requisições. Por exemplo, o ataque *Get-flooding*, que utiliza o método *Get* para realizar um pedido a uma página da internet. O ataque consiste em realizar uma grande quantidade de requisições, e como se assemelha a pedidos de clientes legítimos, o servidor Web aloca recursos para atenter a todos esses pedidos. Esse tipo de ataque, de acordo com relatório do ano de 2014, divulgado pela NSFOCUS [3], foi o quarto mais praticado no mundo.

Nas configurações das redes atuais, se torna complicado realizar ação para mitigar esses tipos de ataques, diante da falta de atualização automática e de forma dinâmica em algumas regras de gerencia da rede, sobretudo em políticas de segurança. A proposta desse trabalho é utilizar o novo paradigma: as redes definidas por *software* (SDN - *Software Defined Network*). No paradigma SDN, há a separação do plano de dados (que realiza o encaminhamento do pacote, por exemplo) e o plano de controle (que define a ação a ser tomada por um determinado pacote) [4]. A partir dos benefícios de uma rede SDN, é possível tomar ações que visam mitigar o ataque.

Diante disso, este trabalho propõe uma defesa, chamada SHADE (*Selective High-Rate DDoS Defense*), para ataques de negação de serviço na camada de aplicação, utilizando a ferramenta *SeVen* [2] como monitoramento de carga e a utilização das Redes Definidas por *Software* para realizar o direcionamento de fluxo e realizar uma estratégia seletiva para mitigar os ataques. Para verificar a consistência da defesa, foram realizadas emulações através do emulador de rede Mininet [5].

Recentemente, utilização de SDN para mitigar ataques de DDoS foi proposta em alguns trabalhos. No entanto, a maioria da literatura aborda ataques nas camadas de Rede e/ou Transporte [6], [7], [8], [9], [10], como por exemplo, ataque do tipo *SYN-Flood*, que tem uma taxa de volume muito maior que os ataques ADDoS; outra parte da literatura aborda ataques DDoS utilizando brechas provenientes das características de uma rede SDN. Algumas estratégias tem sido utilizadas na literatura, como por exemplo: monitorar a quantidade de *handshakes* TCP (SYN, SYN-ACK e ACK), que não são concluídas para cada IP de origem [6]; uso de redes neurais para avaliar se um fluxo é ou não um ataque, com treinamento da rede baseado em parâmetros que podem ser extraídos durante um ataque sobre a camada de Rede/Trasporte (*SYN-flood, UDP-flood, e ICMP-flood*) [8]; identificar possíveis ataques que exploram pontos fracos da arquitetura SDN [7], em particular, a possibilidade de negar o serviço de todas as aplicações da rede SDN enchendo a memória do *switch* com um grande número de regras. Essas abordagens são bem diferentes do

João H. G. Corrêa, Vivek Nigam e Iguatemi E. Fonseca, Universidade Federal da Paraíba, João Pessoa-PB, Brasil; Moisés e Diego Mafioletti, Universidade Federal do Espírito Santo, Vitória-ES, Brasil, E-mails: jhenrique@gmail.com, {vivek, iguatemi}@ci.ufpb.br, moises@ele.ufes.br, loxxxa@gmail.com

SHADE, ou usam estratégias que não são aplicáveis a ataques ADDoS, que tem um comportamento diferente com tráfego que se assemelha ao tráfego cliente [1].

Por fim, mais recentemente, Fayaz *et al.* [10] propôs a combinação de redes SDN e virtualização de funções de redes (NFV – *Network Function Virtualization*) para mitigar grandes ataques de amplificação. A solução é identificar grandes volumes de tráfego e usar aplicações adequadas (através do NFV) em locais estratégicos (utilizando SDN) para mitigar o ataque. A diferença para o SHADE, é o que se propõe mitigar, pois, o tráfego gerado por ataques ADDoS do tipo *High-Rate*, é insignificante em relação aos ataques de amplificação.

As redes SDN também são utilizadas para mitigar ataques ADDoS do tipo *Low-Rate* [11]. Sempre que qualquer fluxo suspeito é identificado, esse fluxo é redirecionado, utilizando regras SDN, a um aplicativo semelhante ao aplicativo de destino para que o atacante não saiba que foi identificado, e assim não degradar o servidor original. Embora este trabalho se assemelhe a proposta SHADE, infelizmente os autores não explicam como poderiam identificar o perfil do tráfego de ataque ADDoS *Low-Rate*, já que se assemelha com clientes legítimos. Se tal identificação é possível, então também seria fácil de incorporar essa defesa com SHADE.

O restante deste artigo está organizado da seguinte maneira. A Seção II apresenta a proposta deste trabalho e a defesa SHADE. A Seção III discorre sobre o cenário criado para verificar a proposta e os resultados obtidos nas emulações. E por fim, na Seção IV são apresentadas as conclusões e o direcionamento para os trabalhos futuros.

II. Selective High Rate DDoS Defense - SHADE

A. SeVen

SeVen foi introduzido por Dantas *et al.* [2], que visa propor uma defesa seletiva para mitigar ataques de negação de serviço na camada de aplicação. *SeVen* funciona como um proxy e monitora o uso do servidor *Web* sob sua proteção. Quando o servidor *Web* não está sobrecarregado, ou seja, que o número de pedido que está sendo processado é menor do que a sua capacidade máxima, *SeVen* permite que qualquer pedido (mesmo de atacantes) possa ser processado. No entando, quando o servidor *Web* está sobrecarregado, isto é, já não consegue processar um novo pedido ao mesmo tempo, e chega um novo pedido, *SeVen* decide (usando alguma função de probabilidade) se a aplicação *Web* deve processar o novo pedido. Há dois resultados:

1) *SeVen* decide que não deve processar este novo pedido, assim simplesmente retorna ao cliente uma mensagem que o servidor não está disponível;
2) *SeVen* decide que ele deve processar este novo pedido. Em seguida, ele também deve decidir qual o pedido que está sendo processado no momento deve ser descartado. Esta decisão é tomada com base em outra distribuição de probabilidade.

Intuitivamente, as estratégias seletivas funcionam porque sempre que um aplicativo está sobrecarregado, é muito provável que ele está sofrendo um ataque. Em vez de bloquear todos os novos pedidos (de ambos, os atacantes e clientes legítimos) como feito sem a defesa, *SeVen* abre a possibilidade de nova solicitação de clientes legítimos serem processados, melhorando assim a disponibilidade do aplicativo. Dantas *et al.* [2] realizou testes com ataques de negação de serviço na camada de aplicação do tipo *Low-Rate* que também exploram vulnerabilidades do protocolo HTTP, mas que tem uma taxa de requisições extremamente baixa. Esses ataques são conhecidos como *Slowloris* e Ataque *POST*.

Embora *SeVen* funcione bem para mitigar ataques *Low-Rate*, ele sofre ao tentar mitigar ataques do tipo *High-Rate*. Isso ocorre porque *SeVen* tem limitada memória e poder de CPU e não pode lidar com a esmagadora maioria das requisições que chegam quando um aplicativo está sofrendo ataques de DDoS com altas taxas, como no ataque *Get-Flood*. Quando o ataque é do tipo *Low-Rate*, como um ataque *Slowloris*, *SeVen* pode garantir serviço a 95% dos clientes legítimos. No entando, quando um ataque é do tipo *High-Rate*, como *Get-Flood*, *SeVen* só pode garantir serviço a uma média de 24% dos clientes legítimos.

O problema é, portanto, desenvolver um mecanismo para defender uma aplicação (servidor *Web*) de ambos os tipos de ataques de negação de serviço na camada de aplicação, *Low-Rate* e *High-Rate*. Embora os tipos *Low-Rate* sejam mitigados por *SeVen*, a proposta deste trabalho é um mecanismo de mitigação a ataques do tipo *High-Rate* usando SDN e *SeVen*.

B. Proposta SHADE

A proposta, chamada SHADE (**S**elective **H**igh r**A**te **D**DoS d**E**fense), utiliza o *SeVen* como monitoramento de carga, ou seja, além da estratégia seletiva que essa estratégia proporciona, o *SeVen* pode indicar quando a aplicação sob sua proteção está sobrecarregado, recebendo muitas requisições.

Para utilizar o *SeVen* como monitoramento de carga, foi elaborado níveis de utilização, descrito abaixo:

- **Verde**: indica que o servidor *Web* está em seu trabalho normal, ou seja, processa uma nova requisição quando ela chega. Na prática, a taxa considerada é até 74% da ocupação do servidor *Web*;
- **Amarelo**: indica que o servidor *Web* está sendo muito utilizado e se continuar a receber mais pedidos irá atingir a sua capacidade máxima. A ocupação nesse nível varia entre 75% e 94%;
- **Vermelho**: indica que a capacidade do servidor *Web* está quase esgotada e não pode, ou em breve não poderá, ser capaz de processar novas requisições. A taxa de ocupação é a partir de 95%.

Portanto, o *SeVen* fica constantemente monitorando a capacidade do servidor e indicando qual o nível de utilização, se está verde, amarelo ou vermelho.

A outra tecnologia que compõe a proposta de defesa é a utilização do controle de fluxo através das vantagens que o paradigma de Redes Definidas por *Software* oferece. Ou seja, o controlador da rede SDN recebe os níveis de utilização indicados pelo *SeVen*, se o nível estiver indicando a taxa de utilização Amarelo ou Vermelho, o controlador redireciona para o SHADE todo o fluxo que chegaria ao servidor *Web*.

Na Figura 1 tem-se um exemplo de regra que o controlador SDN insere no *switch*. Na direita tem-se um computador cliente, com endereço ip 10.0.0.50, que irá realizar

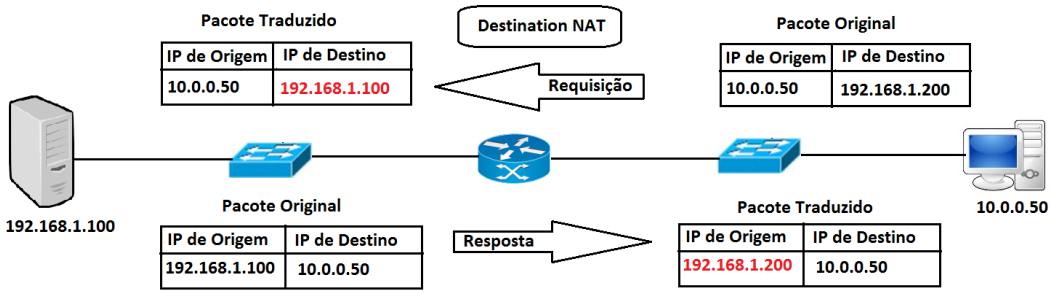

Fig. 1. Exemplo de regra SDN de alteração de fluxo.

uma requisição *Web* para um servidor cujo endereço ip é 192.168.1.200. Com a regra SDN instalada no *switch*, quando o pacote chega no *switch*, o equipamento realiza o *Destination NAT* (*Network Address Translation*), que é a alteração do endereço ip de destino. Ou seja, no pacote original, o cliente enviou para o ip 192.168.1.200, mas quando chegou no *switch* houve a alteração para 192.168.1.100. Da mesma forma o pacote de retorno, o servidor 192.168.1.100 responde ao cliente, quando chega no *switch* o ip de origem é alterado para 192.168.1.200, o endereço original que o cliente enviou. Essa alteração de volta é necessária para não comprometer a confiança do lado do cliente.

Além de realizar a inserção da regra de fluxo no *switch*, alterando o caminho do tráfego, o controlador também informa ao SHADE qual é o nível de utilização, e a partir disso, com o SHADE recebendo o fluxo destinado ao servidor *Web*, é realizado a estratégia seletiva específica para mitigar ADDoS do tipo *High-Rate*, da seguinte forma:

- **Amarelo**: SHADE irá selecionar uma requisição a cada três recebidas para enviar ao servidor *Web*. As requisições não selecionadas são descartadas e o cliente informado que o serviço não está disponível. Ou seja, SHADE reduz dois terços do tráfego destinado ao servidor *Web*;
- **Vermelho**: Com nível vermelho, utiliza-se uma estratégia mais agressiva. Apenas uma requisição a cada cinco recebidas é selecionada, reduzindo drasticamente o tráfego total recebido pela aplicação.

A seleção das requisições é realizada de forma sequencial, ou seja, no nível amarelo a primeira requisição que chegar vai ser aceita, as próximas duas serão negadas, a quarta requisição será aceita enquanto que a quinta e a sexta serão negadas e assim sucessivamente. Da mesma forma no nível vermelho, a primeira requisição é aceita enquanto que da segunda a quinta será negada, a sexta requisição é aceita e da sétima a décima serão negadas.

Por exemplo: tem-se um servidor *Web* que suporta 10 requisições simultaneamente, e o *SeVen*, realizando a proteção desse servidor, define que quando o servidor *Web* estiver com sete clientes entrará no nível amarelo e quando estiver com nove clientes estará com nível vermelho. No início, o nível de utilização é verde. Chega ao servidor seis requisições, o *SeVen* recebe essas requisições e não altera o nível de utilização. Chegando a sétima requisição, o *SeVen* altera para o nível amarelo, e avisa ao controlador da rede SDN que houve uma alteração do nível para amarelo. O controlador, por sua vez, insere uma regra de mudança de fluxo no *switch*, todo o tráfego que iria para o servidor *Web* essa regra encaminha para o SHADE. Além de inserir a regra, o controlador avisa ao SHADE que o sistema está no nível amarelo.

Assim, a oitava requisição será aceita, enquanto que a nona e a décima serão negadas. Chegando mais três requisições, a décima primeira será aceita e as outras duas serão negadas. Com isso o servidor *Web* estará atendendo nove requisições, assim o *SeVen* altera o nível de utilização para vermelho e avisa ao controlador. Como a regra de fluxo já está inserida, o controlador apenas avisa ao SHADE a mudança de nível. Assim o SHADE aceita a décima quarta requisição e nega serviço da décima quinta até a décima oitava requisição, e assim por diante.

Se o nível de utilização do servidor *Web* retornar para acima de sete e abaixo de nove, o nível retornará para amarelo e todo o sistema será avisado da mudança. Note também que se a utilização do servidor *Web* retornar para abaixo de sete, o nível retornará para verde, então o SHADE será desativada e a regra de mudança de fluxo será retirada do *switch*, retornando ao estado da rede original.

Além dos valores escolhidos para compor a defesa SHADE, foram realizados emulações variando os valores de descarte até chegar aos referidos valores. Foram realizados testes com a taxa de descarte de um a cada dois, dois a cada três requisições (escolhido para o nível Amarelo), quatro a cada cinco requisições (escolhido para o nível Vermelho), seis a cada sete requisições e oito a cada nove requisições. Diante das disponibilidades geradas em cada tipo de descarte, foram escolhidas as que obtiveram o melhor desempenho.

III. Emulações e Resultados

A. Cenário

Para a verificação da defesa SHADE, foi elaborado um cenário de acordo com a Figura 2. O cenário é composto por um equipamento SDN, um *switch* que está interligando dois servidores *Web* juntamente com o *SeVen* realizando a proteção e o monitoramento do nível de utilização, um controlador SDN e o SHADE. Ligado ao roteador externo, em outra rede, está uma máquina para simular clientes honestos. Outra máquina, em uma terceira rede, ligada ao roteador, está o atacante.

De acordo com a Figura 2, tanto o tráfego do cliente como do atacante passarão pelo roteador e depois pelo *switch*, é no *switch* que será instalada a regra de mudança de fluxo caso o nível do sistema seja amarelo ou vermelho. Se o nível é verde, o tráfego é encaminhado diretamente ao servidor *Web* de destino, caso o nível seja amarelo ou vermelho, o

tráfego chegando no *switch* é encaminhado ao SHADE, que realiza a estratégia seletiva, e, posteriormente, é encaminhado ao servidor *Web*.

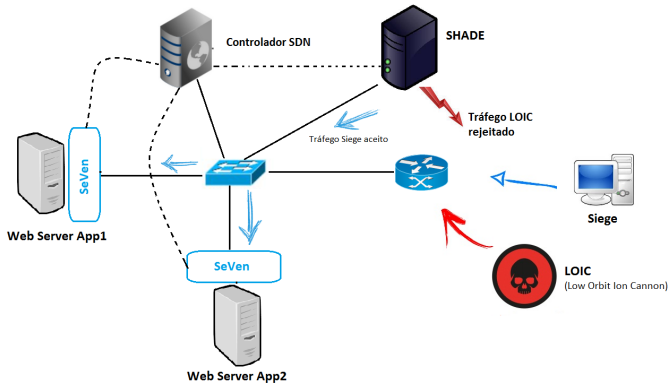

Fig. 2. Diagrama do cenário para avaliação da defesa.

Todo o cenário foi elaborado no Mininet [5], que é um emulador de rede, inclusive de SDN, que "cria uma rede virtual realista, executando em um kernel real" [5]. A versão utilizada foi a 2.2.1, sendo executada numa máquina virtual. Dentro do Mininet foram utilizadas as seguintes ferramentas:

- **Servidor Web Apache** na sua Versão 2.4.7, com 30 posições de *buffer*, caracterizando um pequeno servidor *Web*.
- **Siege** [13] na sua Versão 3.0.5. É uma ferramenta para simular clientes em um servidor *Web*. Foram simulados 20 clientes legítimos;
- **LOIC** (*Low Orbit Ion Cannon*) [12] na Versão 1.0.8.0. É uma ferramenta, criada pelo grupo Anonymous, para gerar ataques do tipo inundação. Especificamente utilizamos o HTTP *GET-Flooding*. No experimento, o tráfego gerado pelo LOIC chegou a 180 vezes o tráfego do cliente.

Foi utilizado outra máquina virtual, contendo o controlador Ryu [14]. O Ryu é um controlador SDN escrito em *Python*, está em constante atualização e suporta até a versão 1.4 do Openflow. Nas emulações foi utilizado a versão 3.6 do Ryu. As regras de mudança de fluxo também foram escritas em *Python*. A implementação do SHADE e as alterações para verificação do nível do *SeVen*, foram implementadas em C++.

Nas emulações foram utilizados como ataque principal o *HTTP Get-Flood*, que é um ataque do tipo *High-Rate* usado para negação de serviço na camada de aplicação e explora o método *GET* do protocolo HTTP. A ideia é simples: o atacante (ou seus bots infectados) simplesmente enviam uma grande quantidade de solicitações *GET* para o servidor *Web* de destino, esgotando sua memória e CPU, enquanto que clientes legítimos não são atendidos.

Uma vez que o serviço *Web* utiliza apenas uma fração da memória e CPU, portanto não é necessário uma grande quantidade de requisições para o ataque. Além disso, mesmo amadores podem realizar tal ataque usando ferramentas como LOIC [12]. O ataque *Get-Flood* é muito eficaz. É possível fazer um servidor *Web* de tamanho pequeno ou médio indisponível usando poucas instâncias do LOIC. Em 2014, ataque *Get-Flood* foi um dos quatros ataques mais populares [3].

As métricas escolhidas, para verificar a consistência da defesa, foram:

- **Disponibilidade**: Essa métrica é a porcentagem da quantidade de clientes legítimos que fizeram uma requisição ao servidor *Web* de uma página, e esse servidor obteve sucesso na resposta da página ao cliente.
- **Tempo de Serviço (TTS –** *Time to Service*): Essa métrica é o tempo, em segundos, que demora para o cliente enviar uma solicitação, o servidor *Web* receber e responder, e o cliente receber a resposta.

B. Emulações

Para as emulações, as máquinas virtuais foram executadas a partir do VMWare Player. A máquina virtual com Mininet tem quatro núcleos de processadores virtuais, 2 GB de memória e 8 GB de HD. A máquina virtual com Ryu tem 1 núcleo de processador virtual, 512 MB de memória e 8 BG de HD. Ambas com o Sistema Operacional Ubuntu 13.04. O computador em que as emulações foram executados era um processador Core i5 4210u, 6 GB de memória RAM e Sistema Operacional Windows 8.1.

Para verificar a proposta, foram realizados emulações em três cenários diferentes:

- **Sem defesa**: Nesse cenário, não há nenhum mecanismo de defesa, ou seja, as requisições dos atacantes e dos clientes vão diretamente ao servidor *Web*. A intenção de realizar emulações nesse cenário é verificar a consistência do ataque, constatando a negação de serviço por parte do servidor *Web*;
- **Apenas SeVen**: Neste segundo cenário foi adicionado uma camada de defesa, para verificar a ação do *SeVen* com tráfego de ataques *High-Rate*;
- **SHADE**: Finalmente o terceiro cenário inclui toda a proposta de defesa do SHADE.

Todos as emulações tiveram duração de 1800 segundos.

C. Resultados

A Figura 3 mostra a média da quantidade de clientes que obtiveram sucesso, nos três cenários investigados. No eixo 'y' é o valor da disponibilidade em porcentagem, no eixo 'x' cada barra significa um cenário: a barra a esquerda é o cenário sem defesa; a barra central é o cenário apenas com *SeVen*; e a barra a direita é o cenário com SHADE.

No cenário sem defesa, obteve-se uma disponibilidade variando entre 16,43% até 25,10%, tendo como média 19,27%. Tecnicamente obtivemos uma negação de serviço a partir do ataque *Get-flooding*, pois de todos os clientes que tentaram requisitar uma página ao servidor *Web*, apenas, em média dos testes, 19,27% deles conseguiram resposta do servidor.

Analisando o cenário apenas com *SeVen*, a disponibilidade variou entre 15,44% e 31,05%, sendo a média entre essas disponibilidades de 24,28%, um aumento de 23% em relação ao cenário sem defesa.

No cenário utilizando SHADE e todo sistema de defesa proposto, a disponibilidade obtida variou entre 58,89% e 64,44%, tendo como a média 61,58% de disponibilidade entre

as emulações realizadas. Um aumento considerável dispondo que nas emulações sem defesa tem-se uma disponibilidade de 19,27% e com SHADE a disponibilidade sobe para 61,58%, um aumento de 220% na disponibilidade.

Na Figura 4 tem-se a média do tempo de serviço. Esse tempo total é medido em segundos, de acordo com o eixo das ordenadas, e no eixo das abscissas estão apresentadas cada cenário de emulação: a barra a esquerda é o cenário sem a defesa; a barra central é o cenário apenas com *SeVen*; e a barra a direita é o cenário com SHADE.

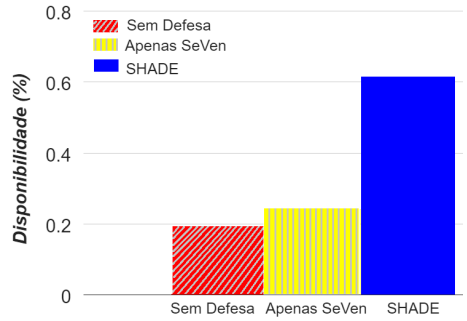

Fig. 3. Disponibilidade nos três cenários.

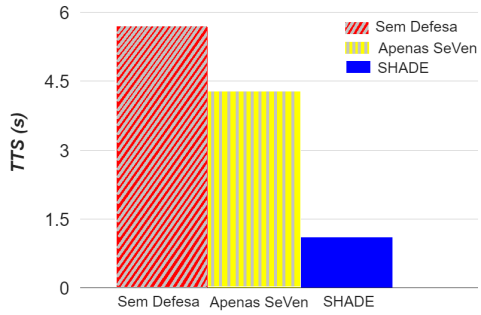

Fig. 4. TTS nos três cenários.

O TTS obtido no cenário sem defesa variou entre 4,01 segundos e 9.03 segundos, tendo como média 5.69 segundos. Um valor que reflete justamente a situação do servidor Web diante de um ataque de inundação, ou seja, com uma grande quantidade de requisições forma-se uma fila de atendimento grande e as requisições têm essa espera para ser atendido. Apesar de, não ter um consenso de quanto um usuário espera para a página carregar, entende-se que cinco segundos é muito tempo de espera.

No cenário em que é inserido a defesa *SeVen* observa-se um TTS variando 1.96 segundo e 6.57 segundos, com uma média de 4.26 segundos nas emulações realizadas. Uma diminuição considerável em relação ao cenário sem a defesa, conseguindo uma diminuição de 37% no tempo de serviço.

Enquanto que no cenário utilizando SHADE, obteve-se o tempo de resposta variando entre 0.69 segundo e 2.6 segundos, tendo como média 1.18 segundo. Há uma diminuição de 70% em relação ao TSS no cenário sem defesa, saindo de 5.69 segundos para 1.18 segundo. Assim como o resultado no cenário apenas com *SeVen*, 1.18 segundo ainda não é o desejável para a experiência de utilização do cliente, que em cenários sem ataque é na casa dos milissegundos. Mas, são resultados ótimos para um cenário em que tem-se uma enxurrada de requisições maliciosas no servidor *Web*.

IV. Conclusões

Este artigo propõe uma nova defesa, chamada SHADE, para mitigar ataques de DDoS na camada de aplicação, do tipo *High-Rate*, utilizando estratégias seletivas em redes SDN. Para isso foi utilizado a ferramenta *SeVen* que mitiga ataques *Low-Rate* e as vantagens oferecidas pelo paradigma das redes SDN. Nas emulações, foi verificado um aumento de 220% na disponibilidade, saindo de 19,27% para 61,58%. Também foi observado uma redução de 70% no TTS, saindo de 5,69 segundos para 1,18 segundo.

Como trabalhos futuros, propõe-se a implementação e experimentação do SHADE em redes SDN reais, utilizando *switchs* que suportem o protocolo Openflow, podendo assim aumentar as proporções utilizadas nas emulações. Sugere-se também a alteração da estratégia seletiva do SHADE, utilizando probabilidade, ou melhorando os valores de descarte escolhidos, ou até utilizar valores variáveis de acordo com a taxa de requisições recebidas. E por fim, propõe-se investigar uma estratégia seletiva para mitigar ataques contra vulnerabilidades SDN, como por exemplo o ataque de inundação de regras que sobrecarrega *switchs* com um grande número de regras, como descrito em [7].

Agradecimentos

Os autores agradecem o apoio do CNPq, CAPES e RNP pelo suporte ao desenvolvimento deste trabalho.

Referências

[1] Saman Taghavi Zargar, James Joshi, and David Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks." *IEEE Communications Surveys and Tutorials*, 15(4), pp. 2046–2069, 2013.

[2] Yuri Gil Dantas, Vivek Nigam, and Iguatemi Fonseca, "A selective defense for application layer ddos attacks." *ISI-EISIC*, 2014. http://www.nigam.info/docs/ddos.pdf.

[3] NS Focus. http://www.prnewswire.com/news-releases/2014-mid-year-ddos-threat-report-documents-high-volume-high-rate-attacks-on-the-rise-276438821.html. 2014.

[4] Hamid Farhady, HyunYong Lee, and Akihiro Nakao, "Software-defined networking: A survey." *Computer Networks*, 81, pp. 79–95, 2015.

[5] Mininet. http://mininet.org/. 2015.

[6] Shin, Seungwon and Yegneswaran, Vinod and Porras, Phillip and Gu, Guofei, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-defined Networks." *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 413–424, 2013

[7] Hommes, S. and State, R. and Engel, T, "Implications and detection of DoS attacks in OpenFlow-based networks." *Global Communications Conference (GLOBECOM), 2014 IEEE*, pp. 537–543, 2014

[8] Braga, R. and Mota, E. and Passito, A, "Lightweight DDoS flooding attack detection using NOX/OpenFlow." *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pp. 408–415, 2010

[9] Wang, Haopei and Xu, Lei and Gu, Guofei, "FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks" *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, pp. 239–250, 2015

[10] Fayaz, Seyed and Tobioka, Yoshiaki and Sekar, Vyas and Bailey, Michael, "Bohatei: Flexible and Elastic DDoS Defense." *24th USENIX Security Symposium, USENIX Security 15, Washington*, pp. 817–832, 2015

[11] Shtern, M. and Sandel, R. and Litoiu, M. and Bachalo, C. and Theodorou, V, "Towards Mitigation of Low and Slow Application DDoS Attacks." *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 604–609, 2014.

[12] Loic. http://sourceforge.net/projects/loic/. 2013.

[13] Siege https://www.joedog.org/siege-home/. 2015

[14] Ryu. http://osrg.github.io/ryu/. 2015