Usando o particionamento de switch para mitigar o problema de acoplamento em Redes Definidas por Software

Francisco de A. C. de A. Jr, Túlio A. Pascoal, Vivek Nigam e Iguatemi E. Fonseca

***Resumo*— A arquitetura de rede definida por software (SDN - *Software Defined Nerworking*) tem como principal característica centralizar o controle da rede em um único ativo, o que torna esse ativo um alvo para ataques de negação de serviço (DoS - *Denial of Service*). Este artigo tem como objetivo demonstrar uma maneira de mitigar ataques de negação de serviço que se concentram em explorar a segurança ao se acoplar vários switches ao mesmo controlador. Para a demonstração utilizamos um ataque dificil de ser detectado conhecido como Slow-Saturation.**

***Palavras-Chave*— Rede Definida por Software, Negação de Serviço, Slow-Satution, Particionamento de Switch**

***Abstract*— The network architecture of a software defined networking (SDN) has as main characteristic to centralize the control of the network in a single asset, which makes this asset a target for denial of service (DoS) attacks. This article aims to demonstrate a way to mitigate denial of service attacks that focus on exploring the security of coupling multiple switches to the same central controller. For the demonstration, an attack, called Slow-Saturation, was used.**

***Keywords*— Software Defined Networks, Denial of Service Attacks, Slow-Saturation, Switch Partitioning**

I. INTRODUÇÃO

A Rede Definida por Software (SDN - *Software Defined Networking*) é uma mudança de paradigma que altera os princípios de funcionamento das redes IP, separando a lógica de controle de roteadores e switches e centralizando-a logicamente em um ativo chamado controlador [1].

O controlador logicamente centralizado não apenas fornece suporte técnico poderoso para serviços de rede complexos, mas também pode obter informações do estado da rede de uma perspectiva global, o que é conveniente para monitoramento de rede em tempo real. Além disso, a separação do plano de controle e do plano de dados simplifica o processo de encaminhamento de pacotes, reduz a carga no switch e torna a configuração da rede mais conveniente [2] [3]. No entanto, como o SDN ainda está em fase de desenvolvimento, muitos detalhes técnicos não estão maduros o suficiente, ele pode facilmente se tornar um alvo importante de ataques de rede. A arquitetura de rede SDN tem sérias vulnerabilidades de segurança, portanto, enfrenta grandes ameaças de segurança [4] [5] [6].

Os ataques de negação de serviço (DoS - *Denial of Service*) são uma das principais ameaças ao SDN, tendo como foco a saturação do recurso. Pesquisas anteriores notaram que existem muitos tipos de ataques DoS em SDN. A vítima pode ser o plano de controle, o plano de dados ou a aplicação. Ataques DoS contra plano de dados, plano de controle ou aplicação SDN geralmente têm princípios e recursos diferentes, correspondentes a métodos de detecção especializados. Portanto, os ataques DoS devem ser estudados de acordo com diferentes planos, e os estudos existentes são comumente conduzidos dessa forma [6].

Várias defesas foram propostas para mitigar os ataques de saturação. A principal suposição subjacente dessas medidas é que os invasores enviarão uma taxa muito alta, por exemplo, falsificando IPs. Isso causa mudanças abruptas nos parâmetros da rede, que facilmente acionam contramedidas de defesa. No entanto, ataques DDoS podem ser lançados não necessariamente gerando alto tráfego. De fato, como testemunhado pela classe de ataques DDoS de taxa lenta da camada de aplicação, como Slowloris [7]. Em vez de enviar diretamente uma enorme quantidade de tráfego, ele gera periodicamente fluxos pulsantes de baixa taxa para causar retransmissão contínua de fluxos TCP benignos, o que resulta em degradação significativa da taxa de transferência. Devido à baixa taxa, tal ataque é mais difícil de ser detectado em comparação com ataques baseados em inundação. Além disso, o ataque pode ser lançado em um modo distribuído, o que aumenta ainda mais sua furtividade [5].

Uma nova forma de ataque chamada ataque Slow-Saturation, combina o ataque Slow-TCAM com uma instância de taxa mais baixa do ataque Saturation. Um ataque Slow Saturation é capaz de negar serviço usando uma fração do tráfego de ataques típicos de Saturation contra o controlador [7] resultando em um problema de acoplamento de vários switches a um controlador central.

Este artigo tem como objetivo demonstrar uma forma de mitigar ataques de negação de serviço que focam em explorar a segurança do acoplamento de múltiplos switches a um mesmo controlador central. Para demonstrar a falha foi utilizado um ataque denominado Slow-Saturation, que é capaz de causar indisponibilidade de um switch SDN, preenchendo completamente sua memória TCAM (*Ternary Content-Addressable Memory*) com regras enviadas por uma botnet [7] [8]. Por meio de testes com o simulador Mininet, percebe-se que switches

Francisco de A. C. de A. Jr, PPGI, Universidade Federal da Paraíba-UFPB, João Pessoa-PB,facajx@gmail.com; Túlio A. Pascoal, Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, tulio.pascoal@uni.lu; Vivek Nigam, Huawei Alemanha – Munique – Alemanha, PPGI. Prof. Colaborador no PPGI, UFPB, João Pessoa-PB, vivek.nigam@gmail.com; Iguatemi E. Fonseca, PPGI, UFPB, João Pessoa-PB,iguatemi@ci.ufpb.br. Este trabalho foi parcialmente financiado pelo CNPq, pela CAPES e pela CODATA-PB.

de redes diferentes que não possuem comunicação entre si e outros controladores, mas compartilham o mesmo controlador, estão sujeitos a sofrer como resultado de um ataque de negação de serviço que está ocorrendo em outra rede. O presente artigo mostra que esse ataque é mitigado se a rede SDN opera com um esquema de particionamento de switch implementado.

O restante do artigo está organizado da seguinte maneira. A Seção II descreve alguns conceitos e problemas de segurança em redes SDN. Na Seção III, a ideia de particionamento de switches é apresentada, bem como tipos de controladores SDN que utilizam ou não esta tecnologia. A Seção IV mostra os cenários experimentais testados, bem como os resultados dos experimentos e respectivos comentários. O artigo é concluído na Seção V com os comentários finais e perspectivas de trabalhos futuros.

II. Fundamentos sobre redes SDN

A. *OpenFlow*

O OpenFlow é um dos padrões de rede SDN mais utilizados, com a característica de separar os ativos da rede em ativos responsáveis pelos dados e ativos responsáveis pela camada de controle da rede, essas camadas são mostradas na Figura 1.

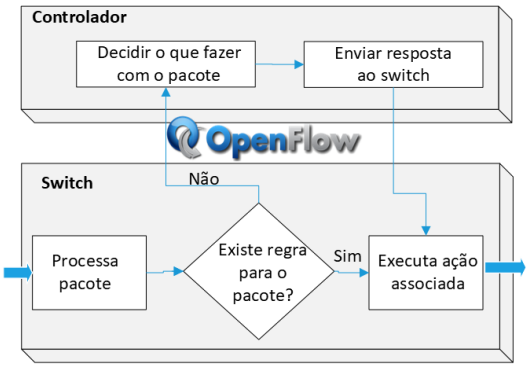

Fig. 1. Processamento de pacotes no switch OpenFlow. Adaptado de [9].

Um dispositivo de comutação de rede, seja um switch ou um roteador, tem suas principais informações armazenadas na tabela de fluxo, que é utilizada para implementar diversas funções como encaminhamento de pacotes, QoS, firewall, etc. O OpenFlow define um conjunto de padrões escaláveis e universais para operar essas tabelas de fluxo.

Quando um novo pacote chega ao switch SDN, ele verifica em sua tabela TCAM se há alguma regra associada ao cabeçalho desse pacote, caso seja encontrada uma correspondência, o pacote será encaminhado pela regra associada. Caso não exista, o switch encapsulará o cabeçalho do pacote em uma mensagem OpenFlow do tipo Packet-In e encaminhará a mensagem ao controlador de rede, que por sua vez analisará o cabeçalho do pacote e, de acordo com sua programação, enviará um pacote de resposta Flow-Mod ao switch com instrução de como lidar com o pacote, conforme Figura 1.

B. *Ataque a memória do switch SDN com Slow-TCAM*

Os switches modernos são normalmente equipados com dois tipos diferentes de memória para encaminhamento de pacotes, Content Addressable Memory (CAM) e Ternary Content Addressable Memory (TCAM), sendo essas memórias especializadas usadas principalmente em equipamentos de rede de alto desempenho. As regras salvas na tabela local do switch têm uma vida útil, que pode durar até que o controlador solicite a remoção, por um tempo fixo (hard-timeout) ou por um tempo de uso (idle-timeout) predeterminado pelo controlador.

Caso a regra seja retirada do switch e o pacote retorne ao fluxo novamente, uma nova requisição deve ser feita ao controlador solicitando instruções de como lidar com o pacote, conforme mostra a Figura 1. Dentre várias técnicas de ataque em switches SDN, o ataque Slow-TCAM tem a característica de não demandar alto consumo de rede, dificultando a detecção e sendo eficaz como forma de preenchimento de toda a tabela de memória TCAM, uma vez que esta tabela é limitada. em número de regras. O atacante usando DDoS (Distributed Denial of Service) pode enviar um grande número de pacotes para o switch da vítima, sendo necessário ter um botnet com tamanho próximo ao limite da tabela TCAM ou metade deste tamanho, pois cada host pode inserir duas regras na tabela, representando a ida e volta do pacote [7]. Por exemplo, para um switch que pode conter apenas 2.000 regras, um botnet com 1.000 hosts seria suficiente para executar o Slow-TCAM.

Para iniciar o Slow-TCAM, o atacante deve identificar o tempo limite de ociosidade utilizado no switch alvo, para sempre renovar o tempo de vida da regra, enquanto insere novas regras no switch, assim, a tabela de fluxo do switch da vítima será preenchida e eventualmente não irá responder a novos clientes, pois não terá espaço para armazenar as regras do controlador. Essa ameaça de switch tem um impacto local, pois faz com que o switch da vítima fique indisponível. Além de aumentar o fluxo de pacotes entre switch e controladora.

Para encontrar o limite de ociosidade o atacante pode testar os valores de tempo de resposta das regras de fluxo para que possa calcular a taxa de ataque mínima para manter as tabelas de fluxo sobrecarregadas. Um pacote sofrerá um atraso de encaminhamento notável se a regra correspondente ao pacote for reinstalada pelo controlador após a expiração do tempo limite. Portanto, é possível estimar os valores de timeout medindo o tempo decorrido entre atrasos notáveis[10].

C. *Atacando a disponibilidade do controlador com Slow-Saturation*

Em um ataque de saturação, o atacante força o switch alvo a instalar um grande número de novas regras de fluxo, consumindo toda a capacidade de buffer do switch, causando aumento de tráfego entre switch e controlador, sobrecarregando o controlador, prejudicando assim o funcionamento de toda a rede [7]. Esse tipo de ataque, por demandar alto consumo de rede, é facilmente detectado.

O Slow-Saturation, por sua vez, inicia-se com um ataque Slow-TCAM em conjunto com um segundo ataque Saturation, mas utilizando apenas rajadas em picos, a fim de minimizar

a detecção do ataque e o preenchimento da tabela TCAM por regras válidas [7].

O Slow-Saturation, além de esgotar a capacidade de memória TCAM do switch alvo por meio do ataque Slow-TCAM, causa um aumento na comunicação entre o switch e o controlador, por meio de um ataque de saturação, o que pode fazer com que o controlador fique sem recursos, não respondendo a solicitações de outros dispositivos da rede [7].

III. Controladores SDN e o Particionamento de Switch

Para os experimentos foram utilizados os controladores Ryu e Floodlight, que são plataformas com uma API completa que suporta o protocolo OpenFlow para gerenciamento e criação de aplicações de controle de rede SDN. O controlador Ryu é escrito em Python, enquanto que o controlador Flooflight em Java.

Cada aplicativo Ryu possui uma fila de recebimento de eventos encadeados que é do tipo FIFO (*First In, First Out*) e preserva a ordem dos eventos. O encadeamento continua drenando a fila de recebimento e chamando o manipulador de eventos apropriado para o tipo de evento.

Em aplicações Python, o GIL (*Python Global Interpreter Lock*) permite que apenas uma *thread* mantenha o controle do interpretador Python por meio de um mutex. Isso pode ser contornado de diversas formas pelos desenvolvedores, mas no caso do Ryu, como o *event handler* é chamado no contexto do encadeamento de processamento de eventos, deve-se ter cuidado ao bloquear, pois enquanto um manipulador de eventos estiver bloqueado, nenhum evento adicional para o aplicativo Ryu será processado [11].

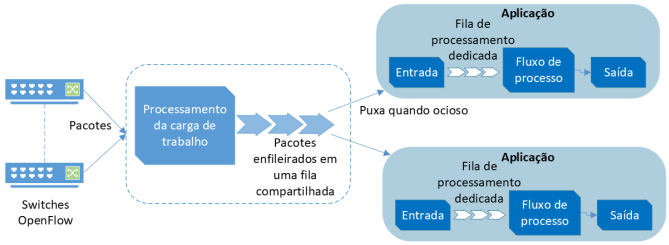

Fig. 2. Fila de processamento compartilhada. Adaptada de [12].

O Ryu, como outros controladores, usa uma fila compartilhada, assim como o controlador Nox. Embora alcance um modelo de programação simples para controlar o desenvolvimento de funções por ter um loop de eventos de *thread* único, não pode aproveitar os avanços atuais. Mesmo utilizando de tecnologia multi-core em alguns controladores como o Maestro uma *thread* principal é responsável por ouvir novas conexões de switch e armazenar os pacotes de entrada em um compartilhamento. Portanto, esses controladores se comportam conforme mostrado na Figura 3, em que há no controlador uma única fila de atendimento compartilhada por vários switches.

O Floodlight, por sua vez, usa Java Netty, uma estrutura de aplicativo de rede assíncrona orientada a eventos, para lidar com *multithreading*. O Floodlight usa assinaturas de *thread* excessivas para garantir que os núcleos não sejam subutilizados. A técnica de particionamento de chave estática é usada tomando um buffer de leitura de tamanho fixo e atribuindo um número fixo de opções a cada *thread*. Após o processamento por pacote, o lote é executado para reduzir a sobrecarga de chamadas do sistema para cada pacote individual.

O particionamento de switch é definido em termos de distribuição e alocação de switches OpenFlow conectados a *threads* em execução no controlador [12]. Controladores como Floodlight e Nox-MT (Nox com *multithreading*) são capazes de realizar particionamento de switch por *thread* do controlador, conforme mostrado na Figura 4.

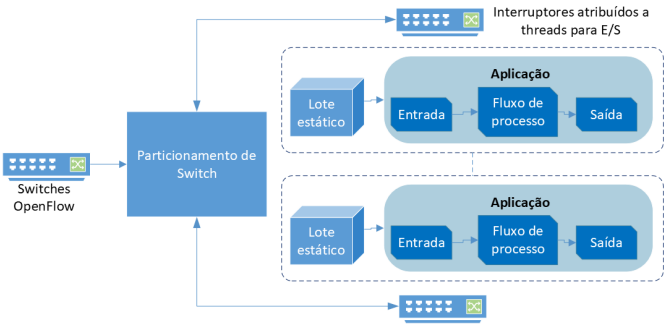

Fig. 3. Particionamento de switch. Adaptada de [12].

A implementação da técnica de particionamento de switches é proposta neste artigo como um meio para tratar ataques DDoS que explorem uma inundação de requisições entre um switch e o controlador. A ideia é evitar que um único switch consuma a fila de requisições do controlador, causando assim indisponibilidade em outros equipamentos que também estejam acoplados ao mesmo controlador. Aproveitando a capacidade *multi-core* dos computadores atuais é possível isolar as requisições por switches ou grupos de switches por *thread*, tendo, portanto, múltiplas filas de atendimento ao fluxo de requisições.

IV. Resultados Experimentais

Para os experimentos, foram utilizadas máquinas virtuais com Oracle VirtualBox Versão 6.1, sendo duas rodando Mininet Versão 2.2.2, uma com o controlador Ryu Framework Versão 4.27, uma com o controlador Floodlight Versão 1.1 e uma com MySQL. A máquina host é um Windows 10 de 64 bits com CPU Intel Core i7-7700HQ a 2,80 GHz e 16 Gb de RAM.

A ferramenta utilizada para analisar o tempo de ida e volta da resposta durante a conexão foi o Apache JMeter Versão 5, que é um software gratuito, multiplataforma, desenvolvido em Java, desenvolvido para medir capacidade e desempenho de aplicações web.

Para os testes realizados, o JMeter foi configurado para criar um grupo de usuários por meio de 3000 *threads*, em um período de execução de 300 segundos, o que faz com que a ferramenta crie em média 10 usuários por segundo, as requisições HTTP executadas possuem um tempo limite de conexão e tempo limite de resposta, ambos de 10.000 milissegundos, tempo recomendado na literatura [13].

A. *Experimento com um controlador*

Para o primeiro ambiente foram utilizadas quatro máquinas virtuais, sendo uma máquina virtual para o controlador, uma máquina com banco de dados MySQL apenas fazendo log das requisições, uma máquina virtual para o Mininet com o Switch S1 e três hosts representando servidor HTTP, cliente e atacante, e outra máquina virtual com switch S2, com um host como servidor HTTP e outro como cliente, conforme a Figura 4. Durante os testes foram verificados o uso de CPU do controlador e o tempo de resposta do servidor HTTP nas redes sem a realização do ataque e também com a execução do ataque.

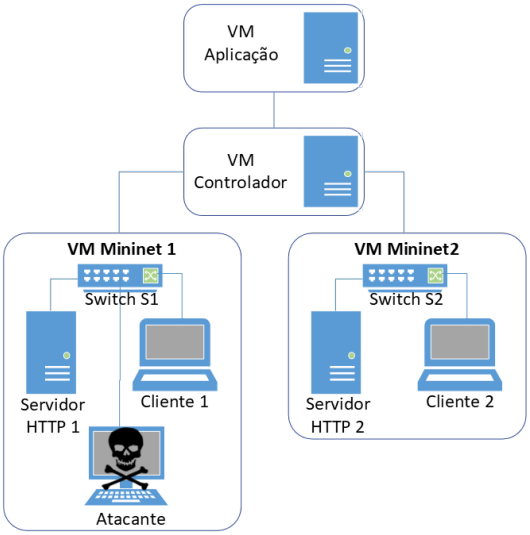

Fig. 4. Experimento com um controlador.

A VM Aplicação, foi responsável por representar uma aplicação simples de log de acesso, sem interferências por parte da aplicação sobre as decisões tomadas pelo controlador.

Os laboratórios foram realizados com os controladores Ryu e Floodlight. Primeiro foi iniciado o ataque Slow-Saturation em um cenário com controlador Ryu, o qual não implementa o particionamento de switch. Logo após a comunicação entre os clientes, como visto na Tabela I, o Mininet VM 1 sofre com o preenchimento de toda a sua tabela TCAM, causando a indisponibilidade total. Por outro lado, o ataque Slow-Saturation realizado entre o atacante e o servidor HTTP 1 é suficiente para preencher a fila de requisições do controlador, fazendo com que ele não consiga atender as requisições do Switch S2, causando uma indisponibilidade de 61,77% na VM Mininet 2, onde de um grupo de 3.000 usuários apenas 1.147 foram atendidos.

Na Tabela II vemos o resultado do mesmo experimento quando o controlador é substituído pelo Floodlight. Como este controlador implementa o particionamento de switches, o ataque não obteve sucesso em inviabilizar a Mininet 2 VM, apenas sofrendo a rede Mininet 1 VM por causa de o Slow-TCAM.

Finalmente, observe nas Tabelas I e II que a rede SDN apresenta 100% de disponibilidade quando não há ataque sendo executado.

TABELA I

EXPERIMENTOS COM RYU PARA TOPOLOGIA COM UM CONTROLADOR.

Sem ataque			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	100%	115 ms	1.224 ms
VM Mininet 2	100%	112 ms	1.363 ms

Com ataque			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	0%	9.071 ms	10.078 ms
VM Mininet 2	38.23%	7.547 ms	18.665 ms

TABELA II

EXPERIMENTOS COM FLOODLIGHT PARA TOPOLOGIA COM UM CONTROLADOR.

Without Attack			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	100%	25 ms	593 ms
VM Mininet 2	100%	21 ms	690 ms

With Attack			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	0%	3.927 ms	10.230 ms
VM Mininet 2	100%	18 ms	689 ms

B. *Experimento com dois controladores*

Para o segundo experimento foi utilizada uma topologia com dois controladores, de forma a prover disponibilidade, em caso falha na comunicação entre o switch e o controlador.

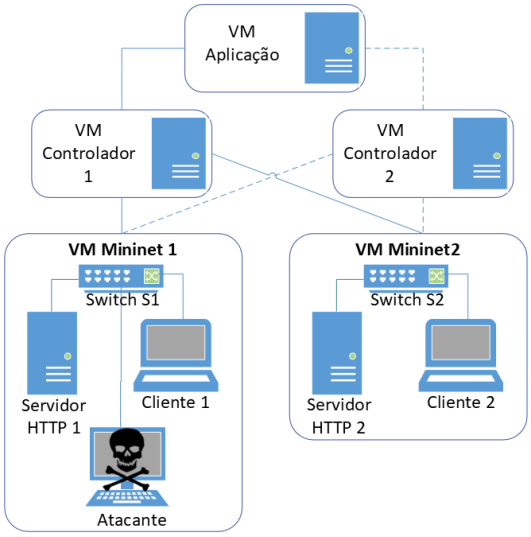

Fig. 5. Experimento com dois controladores.

No experimento foi analisado se a capacidade de disponibilidade provida por esta topologia seria suficiente para conter o ataque de *Slow-Saturation*, durante o teste com duração total de 5 minutos. Foram realizados os seguintes passos, sem a realização do ataque:

1) durante o primeiro minuto os dois controladores estavam online;
2) no segundo minuto o controlador Controlador 1 foi parado;

3) no terceiro minuto o controlador Controlador 1 foi iniciado, estando novamente os dois controladores online;
4) no quarto minuto o controlador Controlador 2 foi parado; e
5) no quinto minuto os dois controladores ficaram online.

Perceba nas Tabelas III e IV que, mesmo executando os procedimentos mencionados na enumeração 1) a 5), a rede SDN apresenta 100% de disponibilidade quando não há ataque sendo executado.

TABELA III
EXPERIMENTS WITH RYU PARA TOPOLOGIA COM DOIS CONTROLADORES.

Sem ataque			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	100%	59 ms	1.140 ms
VM Mininet 2	100%	55 ms	1.090 ms

Com ataque			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	0%	8.801 ms	13.254 ms
VM Mininet 2	57.50%	5.992 ms	12.014 ms

TABELA IV
EXPERIMENTOS COM FLOODLIGHT PARA TOPOLOGIA COM DOIS CONTROLADORES.

Sem ataque			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	100%	6 ms	242 ms
VM Mininet 2	100%	7 ms	286 ms

Com ataque			
Máquina virtual	Disponibilidade	TTS Médio	TTS Max.
VM Mininet 1	0%	5.622 ms	13.229 ms
VM Mininet 2	100%	5 ms	1.487 ms

Note que esta topologia apesar de promover disponibilidade para o caso de perda total de comunicação entre o switch e o controlador, como visto no experimento onde desligamos um dos controladores, o fato do controlador estar com sua fila de processamento saturada não implica em perda de comunicação. Mas com o ataque *Slow-Saturation*, a fila de processamento dos dois controladores foi comprometida, pois o Packet-In enviado pelo switch é sempre encaminhado aos dois controladores.

Chouhan realizou um trabalho analisando o desempenho dos controladores Ryu e Flooflight, comparando-os em diferentes topologias, e afirma que o desempenho do Ryu é melhor que o do Floodlight na maioria dos casos. Em caso de perda de pacotes o desempenho do Ryu é melhor que o do Floodlight[14]. Mas, como visto neste artigo, o fato do Floodlight realizar o particionamento de switchs, fez com que o mesmo não sofra com o problema do acoplamento.

V. CONCLUSÃO

O impacto do ataque Slow-Saturantion foi suficiente para preencher completamente a fila de eventos do controlador Ryu, fazendo com que o controlador ficasse indisponível e prejudicando assim as redes que se conectam a ele. Já o controlador Floodlight implementa o recurso de particionamento atendendo assim as requisições de cada switch em uma fila separada. Assim, mesmo com o switch sendo impactado, o controlador não sofre com ataque e o problema de acoplamento não é encontrado.

Nesse artigo, pode-se observar que o particionamento de switches no processamento de requisições é um recurso importante não apenas por questões de desempenho, mas também pode ser considerado uma boa prática de segurança. Sugere-se como recomendação para evitar que invasores explorem a fila de processamento de pacotes *Packet-In*, pois um invasor pode, por meio de ataques como o *Slow-Saturantion*, onerar a fila de processamento do controlador e, assim, causar indisponibilidade mesmo em redes diferentes que estejam acopladas ao mesmo controlador, independentemente de o invasor ter acesso a essas redes ou não.

Como trabalhos futuros, técnicas de detecção para o ataque *Slow-Saturation* estão sendo investigadas.

REFERÊNCIAS

[1] A. V. Atli, M. S. Uluderya, S. Tatlicioglu, B. Gorkemli, and A. M. Balci, "Protecting sdn controller with per-flow buffering inside openflow switches," in *2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–5, 2017.

[2] R. Amin, M. Reisslein, and N. Shah, "Hybrid sdn networks: A survey of existing approaches," *Commun. Surveys Tuts.*, vol. 20, p. 3259–3306, oct 2018.

[3] T. Li, J. Chen, and H. Fu, "Application scenarios based on sdn: An overview," *Journal of Physics: Conference Series*, 2019.

[4] D. Gao, Z. Liu, Y. Liu, C. H. Foh, T. Zhi, and H. C. Chao, "Defending against packet-in messages flooding attack under sdn context," *Soft Computing*, vol. 22, pp. 6797–6809, 2018.

[5] R. Xie, M. Xu, J. Cao, and Q. Li, "Softguard: Defend against the low-rate tcp attack in sdn," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2019.

[6] M. Yue, H. Wang, L. Liu, and Z. Wu, "Detecting dos attacks based on multi-features in sdn," *IEEE Access*, vol. 8, pp. 104688–104700, 2020.

[7] T. A. Pascoal, I. E. Fonseca, and V. Nigam, "Slow denial-of-service attacks on software defined networks," *Computer Networks*, vol. 173, p. 107223, 2020.

[8] T. A. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow tcam exhaustion ddos attack," in *IFIP International Conference on ICT Systems Security and Privacy Protection*, pp. 17–31, Springer, 2017.

[9] I. A. Karagyulieva, "Openflow and sdn. technical report.," 2014. [Online. https://www.cesga.es/download/openflow-and-sdn-technical-report/. Acessado 17-Maio-2022].

[10] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, and J. Zheng, *Disrupting SDN via the Data Plane: A Low-Rate Flow Table Overflow Attack*, pp. 356–376. 01 2018.

[11] N. Telegraph and T. Corporation., *"Ryu application API,Ryu application programming model. Threads, events, and event queues.* , 2014. [Online; acessado 17-Maio-2022].

[12] S. A. Shah, J. Faiz, M. Farooq, A. Shafi, and S. A. Mehdi, "An architectural evaluation of sdn controllers," in *2013 IEEE International Conference on Communications (ICC)*, pp. 3504–3508, 2013.

[13] Y. G. A. Zarek and D. Lie, "Openflow timeouts demystified," *IEEE Access*, 2012.

[14] R. K. Chouhan, M. Atulkar, and N. K. Nagwani, "Performance comparison of ryu and floodlight controllers in different sdn topologies," in *2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing Communication Engineering (ICATIECE)*, pp. 188–191, 2019.