CHRISTIAN FÉLIX GBÈDANDÉ AZONYÈTIN

UMA ARQUITETURA SDN BASEADA NOS PLANOS DE CONHECIMENTO E GERENCIAMENTO PARA DETECÇÃO E MITIGAÇÃO DE ATAQUES DDOS

LAVRAS – MG

2023

CHRISTIAN FÉLIX GBÈDANDÉ AZONYÈTIN

UMA ARQUITETURA SDN BASEADA NOS PLANOS DE CONHECIMENTO E GERENCIAMENTO PARA DETECÇÃO E MITIGAÇÃO DE ATAQUES DDOS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Sistemas de Computação, para a obtenção do título de Mestre.

Prof. DSC Luiz Henrique Andrade Correia
Orientador

LAVRAS – MG
2023

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Azonyètin, Christian Félix Gbèdandé
Uma arquitetura SDN baseada nos Planos de Conhecimento e Gerenciamento para Detecção e Mitigação de Ataques DDoS / Christian Félix Gbèdandé Azonyètin. – Lavras : UFLA, 2023.
77 p. : il.

Dissertação (Mestrado acadêmico) – Universidade Federal de Lavras, 2023.
Orientador: Prof. DSC Luiz Henrique Andrade Correia.
Bibliografia.

1. DDoS. 2. Plano de conhecimento. 3. Plano de gerenciamento. I. Correia, Luiz Henrique Andrade. II. Título.

CHRISTIAN FÉLIX GBÈDANDÉ AZONYÈTIN

UMA ARQUITETURA SDN BASEADA NOS PLANOS DE CONHECIMENTO E GERENCIAMENTO PARA DETECÇÃO E MITIGAÇÃO DE ATAQUES DDOS

AN SDN ARCHITECTURE BASED ON KNOWLEDGE AND MANAGEMENT PLANS FOR DETECTION AND MITIGATION OF DDOS ATTACKS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Sistemas de Computação, para a obtenção do título de Mestre.

APROVADA em 16 de Junho de 2023.

Prof. DSC Luiz Henrique Andrade Correia	UFLA
Prof. DSc. Valderi Reis Quietinho Leithardt	IPP-ippotalegre.pt
Prof. DSc. Hermes Pimenta de Moraes Junior	UFLA

Prof. DSC Luiz Henrique Andrade Correia
Orientador

LAVRAS – MG
2023

Aos meus pais, PAULIN e ALICE, aos meus filhos GRACE e JOSEPH por todo o apoio,carinho e compreensão ao longo desta caminhada.

AGRADECIMENTOS

Agradeço a Deus por seu plano na minha vida.

Agradeço a toda a minha família pelo carinho e apoio, aos meus pais e irmãos pelo porto seguro, conselhos e incentivos, e por não mediram esforços para que eu chegasse até esta etapa na caminhada da minha vida.

Ao professor Luiz Correia, uma pessoa incrível. Para mim, você não é só um professor; mas muito mais. Obrigado por teu coração maravilhoso, pela oportunidade, por toda a orientação, a paciência, os ensinamentos e auxílios no desenvolvimento e implementação do trabalho.

Aos meus companheiros de mestrado Renan Villela Oliveira e Victor Boell , que muito ajudaram na elaboração, resolução deste trabalho e nos estudos.

A ANNA DEBORA Da-Silva por ter ajudado com todos os conselhos, ombro e companheirismo nos momentos difíceis e todo apoio e carinho que poderia me proporcionar.

A professora MÓNICA da IFPB na Paraíba pelo apoio e ajuda. Deus lhe abençoe.

Ao meu amigo RACHAD AMADOU obrigado pelo apoio nos momentos difíceis para que eu possa seguir em frente na minha vida, outros amigos que também participaram nesta caminhada e ajudaram a distrair nos momentos necessários.

A SIMONE Alves Da-Silva por ter ajudado com todos os conselhos, ombro e companheirismo nos momentos difíceis e todo apoio e carinho que poderia me proporcionar.

Agradeço a Universidade Federal de Lavras, especialmente ao Programa de Pós-Graduação em Ciência da Computação, pela oportunidade.

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa de Minas Gerais (FAPEMIG).

Muito Obrigado!

"O que sabemos é uma gota; o que ignoramos é um oceano."
(Isaac Newton)

RESUMO

Os inúmeros ataques às redes de computadores tem causado sérios prejuízos para empresas e usuários em geral. As Redes Definidas por Software, ou *Software Defined Networks* - SDN, surgem como uma alternativa inovadora para isolamento e controle do tráfego da rede. O paradigma SDN permite a criação de regras a partir de um controlador que define ações sobre o comportamento do tráfego na rede. Apesar disso, as SDN também sofrem de problemas de segurança, um dos principais tipos de ataque é baseado em negação de serviço distribuído (*Distributed Denial of Service - DDoS*), que pode atingir tanto os servidores da rede quanto o controlador SDN, deixando a rede inoperante. Na literatura atual existem relatos que os controladores da SDN não são capazes de lidar com um grande número de novos fluxos, criando vulnerabilidade na segurança dessas redes. A maioria das soluções propostas utilizam algoritmos de aprendizado de máquina para classificar o tráfego da rede de uma forma não estruturada dentro da atual arquitetura SDN. Neste trabalho, o objetivo é o desenvolvimento de uma nova arquitetura SDN para a detecção e mitigação de ataques de DDoS, que inclua o plano do conhecimento (*Knowledge Plane -KP*) e o plano de gerenciamento. O novo plano KP aproveita informações dos planos de gerenciamento (*Management Plane - MP*) e de controle para obter uma visão geral da rede e possibilitar um controle mais inteligente. O KP é responsável por aprender o comportamento da rede e, em alguns casos, operar autonomamente a rede, utilizando técnicas de aprendizado de máquina para classificação e análise do tráfego da rede. Para o treinamento dos algoritmos de aprendizado de máquina foram gerados *datasets* de tráfegos legítimo e malicioso em uma estrutura de rede SDN experimental com switches e topologias reais. Os fluxos foram direcionados aos servidores e ao controlador da rede utilizando-se ferramenta de ataque como a Bonesi e T50. Como resultado, a nova arquitetura SDN proposta foi capaz de detectar e mitigar os ataques de DDoS, impedindo o esgotamento dos recursos do controlador SDN e evitando o congestionamento da rede. Como precisão durante os experimentos do cenário híbrido, Naive-Bayes foi o melhor porque teve $92,95\%$ de acertos. Os algoritmos SVM, KNN e Árvore de Decisão tiveram respectivamente $78,18\%$, $79,06\%$ e 64% de precisão de acertos. A métrica Acurácia obtida pelos algoritmo Árvore de Decisão, Naive Bayes, KNN e SVM foi respectivamente $74,28\%$, $93,82\%$, $90,42\%$ e $86,36\%$. A métrica Revocação deu 100% para todos os algoritmos, enquanto a métrica Medida-F deu para os algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM respectivamente 78% , $96,35\%$, $88,31\%$ e $87,75\%$. Para melhorar as técnicas de detecção e mitigação de ataques DDoS, e identificar requisitos de uma solução mais eficaz, são propostos módulos (de pré-processamento, de análise estatistica, de decisão...) para definir características e funções das camadas da nova arquitetura proposta. Todos os Hosts que foram classificados como maliciosos, foram automaticamente bloqueados com sucesso durante um momento de 60s então um resultado de 100% de bloqueio.

Palavras-chave: DDoS. Segurança. SDN. Plano de Conhecimento. Plano de Gerenciamento. Openflow. Aprendizado de maquina.

ABSTRACT

The numerous attacks on computer networks have caused serious damage to companies and users in general. Software Defined Networks, or Software Defined Networks - SDN, emerge as an innovative alternative for isolating and controlling network traffic. The SDN paradigm allows the creation of rules from a controller that defines actions on the behavior of traffic on the network. Despite this, SDN also suffers from security problems, one of the main types of attack is based on Distributed Denial of Service (DDoS), which can reach both network servers and the SDN controller, leaving the network dead. In the current literature, there are reports that SDN controllers are not capable of handling a large number of new flows, creating vulnerability in the security of these networks. Most proposed solutions use machine learning algorithms to classify network traffic in an unstructured way within the current SDN architecture. In this work, the objective is the development of a new SDN architecture for the detection and mitigation of DDoS attacks, which includes the Knowledge Plane (KP) and the management plane. The new KP plan leverages information from the plans Management Plane (MP) and control planes to gain an overview of the network and enable smarter control. The KP is responsible for learning the behavior of the network and, in some cases, operating the network autonomously, using machine learning techniques for classifying and analyzing network traffic. For the training of machine learning algorithms, datasets of legitimate and malicious traffic were generated in an experimental SDN network structure with switches and real topologies. The flows were directed to the servers and the network controller using attack tools such as Bonesi and T50. As a result, the proposed new SDN architecture was able to detect and mitigate DDoS attacks, preventing the exhaustion of SDN controller resources and avoiding network congestion. As for accuracy during the hybrid scenario experiments, Naive Bayes was the best because it had $92,95\%$ hits. The SVM, KNN and Decision Tree algorithms had respectively $78,18\%$, 79.06% and 64% accuracy of hits. The Accuracy metric obtained by the Decision Tree, Naive Bayes, KNN and SVM algorithms was respectively $74,28\%$, 93.82%, 90.42% and 86.36%. The Revocation metric gave 100% for all algorithms, while the F-Measure metric gave Decision Tree, Naive Bayes, KNN and SVM algorithms respectively 78%, 96.35%, 88.31% and 87.75%. To improve DDoS attack detection and mitigation techniques, and identify requirements for a more effective solution, modules are proposed (pre-processing, statistical analysis, decision making...) to define characteristics and functions of the layers of the new architecture proposal. All Hosts that were classified as malicious were automatically successfully blocked for a period of 60s so a result of 100% blocking.

Keywords: DDoS. Security. SDN. Knowledge Plan. Management Plan. Openflow. Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O surgimento da Internet mudou o cotidiano das pessoas no mundo inteiro. De um lado, observa-se com preocupação que com a revolução das novas tecnologias de informação e comunicação, NTIC (chamado também como TIC), novas ameaças cibernéticas têm surgido. É importante verificar que muitas vezes a revolução tecnológica pode alavancar a ciência, trazer benefícios para a sociedade, mas também pode causar prejuízos e riscos à sociedade. De outro lado, o rápido crescimento das redes tornou suas infraestruturas complexas. Com isso, as propriedades essenciais de uma rede, como integridade, confidencialidade, autenticação, disponibilidade de informações e não repúdio, estão se tornando um grande desafio (COSTIN et al., 2013).

Entre os males da revolução da tecnologia da informação está o cibercrime. Existem vários tipos de cibercrimes, mas nos dias de hoje, os ataques distribuídos de negação de serviço (DDoS- *Distributed Denial of Service*) representam uma ameaça real e bastante prejudicial para as redes. Esses ataques que ocorrem em um ambiente amplamente distribuído, ou seja, após infectar um exército de máquinas vítimas (computadores zumbis), o atacante coordena de maneira anônima que esses computadores ataquem ao mesmo tempo, um determinado alvo para interromper serviços de tecnologia da informação, a fim de tornar os recursos de um sistema indisponíveis (BAWANY; SHAMSI; SALAH, 2017).

Nos últimos anos, muitos pesquisadores e indústrias mudaram seu foco para um projeto de redes mais robustas, escaláveis e seguras. As ferramentas de segurança de redes existentes, como antivírus, anti-spam e *firewalls*, usadas em muitas organizações, podem proteger contra ataques de rede. Mas, essas ferramentas não podem reconhecer ataques novos e complicados (DEEPA; RADHA, 2021). Os avanços mais recentes, como a Rede Definida por Software (SDN - *Software Defined Networking*), aliada às técnicas de aprendizado de máquina (ML - *Machine Learning*) e o desenvolvimento de ferramentas de simulação, são passos em direção ao estabelecimento de uma rede de natureza dinâmica, centralizada e mais segura em comparação com o ambiente estático e distribuído das redes tradicionais (SINGH; BEHAL, 2020).

A SDN possui algumas vantagens em relação à rede de computadores tradicional, como possibilidade de criação de regras de fluxos no controlador, visão e controle global e centralizada da rede, mecanismo de autorrecuperação (DABBAGH et al., 2015). O conceito de SDN tem sido apontado como promissor para descalcificar as tecnologias de rede e obter-se avanços expressivos na área. O paradigma SDN separa o plano de dados, do plano de controle, assim todo o controle lógico da rede, antes alocado nos dispositivos de *hardware*, é migrado para um componente centralizado logicamente, o controlador de rede (CHICA; IMBACHI; VEGA, 2020). Entretanto, os autores Liu et al. (2019a), Shaghaghi et al. (2016) e Dabbagh et al. (2015) argumentam que a implementação da SDN possui algumas falhas que

permitem ataques aos planos (dados, controle e aplicação). Na arquitetura SDN, o comprometimento do controlador ou ataque de "homem no meio" são exemplos de ataques DDoS. Portanto, é necessário enfatizar a segurança da SDN para não comprometer seus usuários.

Como parte dessa inovação para facilitar a comunicação entre o switch e o controlador da rede, o protocolo OpenFlow foi desenvolvido pela *Open Network Foundation* (MCKEOWN et al., 2008). O protocolo OpenFlow, é a primeira interface padrão de comunicação entre as camadas da arquitetura SDN. OpenFlow usa o conceito de fluxo para identificar o tráfego de rede e armazenar suas informações (TANG et al., 2016). Pelo protocolo OpenFlow são enviados comandos que manipulam o tráfego por meio das tabelas de fluxos (*flow tables*). Dentro do switch OpenFlow, existem essas tabelas de fluxo que armazenam os comandos de fluxo provenientes do controlador e encaminhado pelo OpenFlow. Esses comandos são usados para excluir, atualizar e adicionar regras nas tabelas para determinar se um fluxo será bloqueado ou não. Além disso, o Openflow possibilita a aquisição de estatísticas de rede diretamente dos switches que suportam esse protocolo. Assim, de posse dessas informações, é possível implementar programas auxiliares ao plano de controle para processar e disponibilizar novas informações aos administradores da rede (FOUNDATION, 2012).

Na SDN, para obter as informações sobre o fluxo de dados entre máquinas e servidores, a análise de redes de computadores se concentra no uso de métodos. Assim, para obter um mapa conceitual da rede ou a possível identificação de anomalias de tráfego, os dados obtidos são processados. (TRAJKOVSKA et al., 2017). As técnicas de Aprendizado de Máquina (ML) podem ser usadas para tal tarefa. O objetivo de ML que é uma subárea de inteligência artificial (AI-*Artificial Intelligence*), é a geração de modelos inteligentes por meio da detecção de padrões em dados. Entre outros algoritmos, árvores de decisão, redes neurais, métodos estatísticos, etc., (baseados em diversas técnicas), são utilizados em ML (PRAKASH; PRIYADARSHINI, 2018). Vários autores, como Elsayed et al. (2020), Sanjaa e Chuluun (2013) utilizaram algoritmos de ML para detecção de ataques. Deve-se notar que os resultados da verificação emitiram o menor número de falsos positivos, como classificar o tráfego malicioso como legítimo e vice-versa (NETO, 2017).

1.1 Motivação

Durante a pandemia de *COVID-19*, houve um rápido crescimento do tráfego da Internet, foram desenvolvidas várias aplicações destinadas à educação, saúde e compras online. Além disso, a implantação de tecnologias 5G aceleraram a proliferação da Internet das Coisas *(IoT-Internet of Things)* no mundo inteiro. Com o desenvolvimento dessas novas tecnologias, novas vulnerabilidades foram de-

tectadas pelos cibercriminosos, ampliando o número de ataques na rede. Segundo “Olhar Digital” [1], a *Cloudflare* relatou um aumento de 175% em nesses incidentes apenas durante o quarto trimestre de 2021. A própria Microsoft relatou que conseguiu evitar o maior ataque DDoS de todos os tempos, que atingiu 3,47 terabits por segundo. A Google afirmou que em junho de 2022, segundo a *Bleepingcomputer*, que um usuário do *Cloud Armor* começou a receber 46 milhões de requisições por segundo *(RPS-requests per second)*, um ataque de DDoS ao HTTPS-*Hyper Text Transfer Protocol Secure*. Em 8 minutos, o ataque começou de 10.000 RPS no balanceador de carga HTTP/S até 100.000 RPS, por meio de 5.256 endereços IP localizados em 132 países; acionando assim *Cloud Armor Protection* do Google e criando um alerta derivado de dados de análise de tráfego. E não foi feito por um único hacker, devido ao uso de solicitações criptografadas (HTTPS). Os equipamentos envolvidos na operação poderiam ter sido suportados por fortes recursos computacionais. O Google afirma ainda não ser capaz de identificar um nome exato para esse tipo exato de malware usado.

Recentemente, eventos geopolíticos na Ucrânia demonstraram a eficácia dos cibercriminosos no lançamento de ataques de DDoS contra infraestruturas críticas e agências governamentais. Os pesquisadores de "(A10) *Five Most Famous DDoS Attacks and Then Some 2021"* [2] detectaram aproximadamente 15,4 milhões de ataques de DDoS. De fato, já no primeiro dia do conflito, houve dois grandes ataques DDoS direcionados da Rússia contra Ucrânia:

a) *PP Infoservis-Link*, com mais de dois milhões de solicitações atendidas pelo protocolo *Apple Remote Desktop (ARD)* na porta UDP 3283. A maioria dos pacotes solicitados foi de 370 bytes;
b) O segundo alvo foi o Secretariado do Gabinete de Ministros da Ucrânia - com mais de 600.000 solicitações de *Network Time Protocol (NTP)*. Este ataque DDoS na porta UDP 123 foi direcionado a um bloco de endereços IP 194.79.8.0, localizado na cidade de Kharkiv, com 1014 bytes de pacotes.

Como observado, nos dois exemplos do primeiro dia do conflito (24 de fevereiro 2022), os ataques foram muito grandes em escala. Foram enviados 2.099.092 requisições de 500 bytes. São cerca de 6 GB, todos solicitados de uma única máquina. Esses exemplos de incidentes mostram claramente que precisamos de novas abordagens para solucionar esse problema de ataques DDoS. De acordo com a CISCO VNI (*Cisco Visual Networking Index*), ataques DDoS representam 25% do tráfego de um país enquanto ele está ocorrendo (BARNETT et al., 2018). Dessa forma, a segurança da informação requer a utilização das tecnologias e ferramentas mais recentes.

[1] https://olhardigital.com.br/2022/08/19/seguranca/google-registra-o-maior-ataque-https-ddos-da-historia/
[2] https://www.a10networks.com/blog/5-most-famous-ddos-attacks

A partir da motivação levantada, emerge a necessidade de se proteger dos ataques dada a sua relevância. Assim, existe a necessidade de desenvolver novas formas de proteção tendo como particularidade ter em conta requisitos de desempenho e escalabilidade. Nos últimos anos, as indústrias e as comunidades de pesquisa de rede implantaram muitas soluções para projetar uma rede futura mais segura, baseadas em protocolo HTTP - *narrow waist*), *Named Data Networking* (NDN), redes programáveis e SDN. Dentre eles, a SDN é apontada como a solução mais viável para lidar com os problemas da rede atual (SINGH; BEHAL, 2020). Devido à possibilidade do controle centralizado na arquitetura SDN, os administradores de rede podem detectar e reagir aos ataques com mais eficiência; sendo possível desenvolver recursos de análise de tráfego baseados em software (BHUSHAN; GUPTA, 2019), (BONFIM; DIAS; FERNANDES, 2019), (YI et al., 2018). É importante observar, que a maioria das soluções comumente oferecidas para mitigar DDoS se baseiam inteiramente na maior alocação de recursos, e apesar desses esforços de pesquisa e desenvolvimento para mitigar os ataques nas redes, o problema ainda persiste.

1.2 Definição do Problema

Dada uma rede SDN que recebe múltiplos fluxos de dados como tráfego de entrada, o problema neste trabalho é propor uma arquitetura SDN que incorpora os planos de conhecimento e de gerenciamento que possa detectar e mitigar os ataques de DDoS num tempo hábil de forma automática, evitando ou minimizando suas consequências desastrosas.

O problema dos ataques DDoS continua crucial devido à falta de soluções capazes de mitigar e realizar ações para classificar o tráfego legítimo do tráfego malicioso em uma rede SDN; e isso pode deixar a rede vulnerável a ataques de saturação nos dispositivos de encaminhamento e controlador (HELLER; SHERWOOD; MCKEOWN, 2012). Este tipo de ataque aproveita o longo atraso na comunicação entre os planos (dados e controle), causando inundação no canal de comunicação entre o switch e o controlador, impedindo que fluxos legítimos possam ser atendidos pelo controlador e assim acarretando numa saturação na rede (SHIN et al., 2013)

Entre outros motivos da persistência e da evolução do problema de ataque DDoS, são as vulnerabilidades do SDN. Embora, a segurança de redes baseadas em SDN tenha sido um ponto de debate, a segurança da própria infraestrutura SDN tem sido questionada.

Devido a falhas da arquitetura tradicional SDN, e conforme descrito por Clark et al. (2003) não basta apenas melhorar gradativamente as técnicas e algoritmos de ML que conhecemos, mas sim criar uma nova definição da arquitetura SDN, incorporando o plano de conhecimento (KP - *Knowledge Plane for the Internet*) à sua arquitetura; porque na maioria das pesquisas foi observado que muitos trabalhos

usam modelos de ML supervisionados para treinamento. Além disso, esses trabalhos não definem ou identificam o KP na arquitetura SDN, o que pressupõe que as características da rede provavelmente permanecerão idênticas ao longo do tempo. Portanto, eles não são adequados para redes dinâmicas que precisam de um modelo de treinamento online. Em outros trabalhos, o plano de gerenciamento (*Management Plane*) não é implementado ou considerado dentro da arquitetura tradicionalmente proposta. Dessa forma, o administrador de rede precisa coletar e examinar os parâmetros da rede para tomar uma decisão sobre como definir regras para o tráfego da rede. No entanto, levará horas ou dias para a detecção ou geração de uma nova decisão para um ataque, o que pode implicar na incapacidade de lidar com ataques rápidos.

1.3 Objetivos

O objetivo principal desta pesquisa é apresentar uma nova arquitetura SDN na qual, módulos (aplicações) serão implementados para definir características e funções das camadas. A nova arquitetura incorpora o plano de conhecimento KP e de gerenciamento à SDN tradicional.

1.3.1 Objetivos Específicos

O resultado final deste trabalho deverá ser capaz de:

a) Reduzir a sobrecarga do controlador por meio de regras de fluxo, criando os planos de conhecimento e de gerenciamento para facilitar seu processamento;
b) Implementar módulos (aplicações) nos diferentes planos para aumentar o desempenho da rede e torná-la mais eficiente;
c) Adaptar o plano de gerenciamento para viabilizar a coleta de parâmetros e monitoramento da rede;
d) Gerar datasets com tráfego legítimo e malicioso para treinamento dos algoritmos de ML;
e) Combine o resultado do melhor modelo ML de classificação de tráfego do plano KP para o plano de controle para gerar regras de fluxo com base nesse resultado;
f) Avaliar a precisão dos métodos de detecção, com base, nas métricas de avaliação;
g) Analisar as ações para detectar e mitigar os ataques DDoS na rede utilizando algoritmos de ML e módulos implementados no plano de conhecimento.

Então o produto final desta pesquisa proporcionará um ambiente mais seguro na rede.

1.4 Organização do trabalho

A seguir é mostrada a organização deste trabalho. O Capítulo 2 apresenta o referencial teórico, abordando os principais conceitos em que se baseia o trabalho. O Capítulo 3 sintetiza alguns trabalhos relacionados a esta pesquisa, contribuindo para o seu desenvolvimento. A metodologia é apresentada no Capítulo 4, descrevendo a nova arquitetura proposta, os cenários de experimentos, as ferramentas, os datasets e as métricas usadas. O Capítulo 5 apresenta os resultados obtidos de acordo com os objetivos definidos. Por fim, o último capítulo 6 apresenta as conclusões do trabalho, por meio de uma análise crítica dos resultados obtidos, e a definição de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo são abordados temas relacionados a redes SDN, sua arquitetura, bem como o principal protocolo para a comunicação entre o plano de dados e controle. São apresentados os principais controladores para o desenvolvimento de uma rede SDN, o conceito de plano de dados programável, segurança em redes, segurança em SDN, bem como suas vulnerabilidades aos principais ataques de DDoS presentes em redes de arquitetura tradicional e os mecanismos de detecções. Também são apresentados, diversos algoritmos de ML utilizados no plano de conhecimento KP da arquitetura proposta.

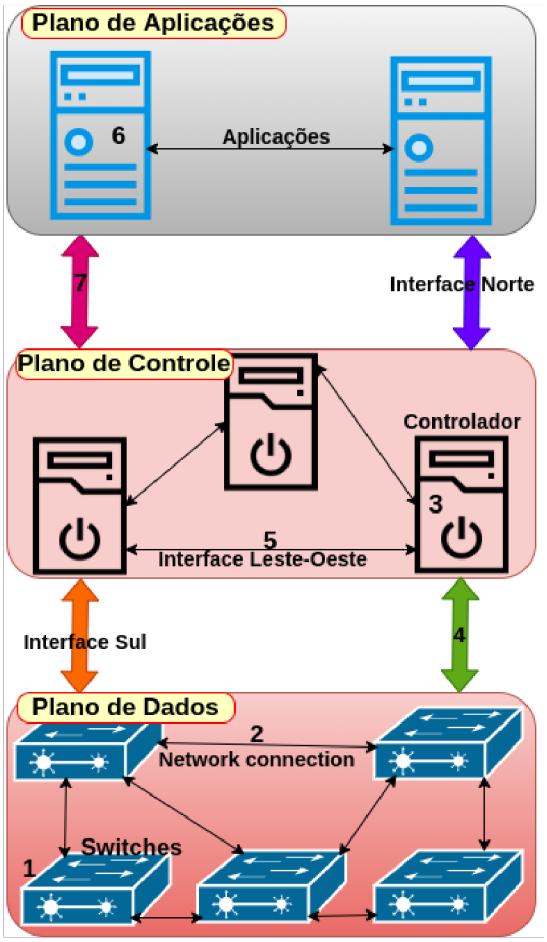
2.1 Redes Definidas por Software – SDN

As redes definidas por software possuem uma arquitetura que permite o gerenciamento flexível de redes de computadores por meio da separação do plano de encaminhamento de dados e do plano de controle (YUREKTEN; DEMIRCI, 2021). Os principais recursos da SDN incluem o gerenciamento centralizado logicamente, interfaces programáveis com padrões abertos, protocolos de gerenciamento de switch de redes, a capacidade de criar redes lógicas virtualizadas e permitir o uso de módulos de monitoramento centralizado. As características como programabilidade, flexibilidade, controle global e centralizado da rede são alcançadas pela separação dos planos (dados e controle), permitindo assim que o processamento de rotas e estatísticas de fluxo sejam realizadas em hardware externo aos switches e roteadores da rede. Isso simplifica as funções do switch. Com o uso do protocolo OpenFlow, é possível a comunicação entre o controlador e os elementos de rede (MCKEOWN et al., 2008).

De acordo com Vaughan-Nichols (2011), a arquitetura SDN consiste em três camadas e de interfaces de comunicação entre camadas, conforme mostrado na Figura 2.1. O switch SDN (1) na camada de dados encaminha o tráfego de entrada (2) de acordo com as regras em suas tabelas de fluxo programáveis e coleta informações estatísticas sobre o tráfego que passa por ele. Se um pacote de entrada não corresponder a nenhuma das regras nas tabelas de fluxo, o switch SDN executa a ação padrão de encaminhar o pacote para o controlador SDN (3) na camada de controle por meio da interface sul (4). O controlador SDN decide o que fazer para os pacotes vindos dos switchs SDN e define novas regras de fluxo (encaminhar, atualizar, descartar, etc.) para os comutadores. Usando a interface sul, o controlador também pode coletar informações estatísticas dos switchs, periodicamente ou sob demanda. Pode haver vários controladores SDN na camada de controle. Nesse caso, os controladores se comunicam entre si por meio da interface Leste-Oeste (5). As aplicações na camada de aplicação (6) são responsáveis pela segurança da rede, monitoramento de tráfego, etc. Elas podem fazer alterações na rede ou monitorar a rede por meio do controlador por meio da interface norte (7). A seguir, são apresentadas a arquitetura

SDN na Figura 2.1, os diferentes planos (dados, controle e aplicações), as interfaces, o switch OpenFlow e diferentes controladores.

Figura 2.1 – Arquitetura de rede definida por software.



Fonte: do próprio autor do trabalho (2023)

2.1.1 Plano de Aplicações

O plano de aplicação é a interface entre a SDN e o administrador da rede. Este plano é responsável por programar equipamentos e executar diversas funções de rede, além de sua capacidade de fornecer informações sobre o estado e notificações de eventos da rede. Estas funções são facilmente executadas usando APIs (*Application Programming Interfaces*) que fornecem comunicação entre o plano de aplicação e o controlador. Além disso, as linguagens de programação utilizadas nas aplicações deste plano fornecem diversas abstrações, facilitando a programação de aplicações e o reaproveitamento de códigos.

Funções semelhantes às das redes de computadores convencionais, como balanceamento de carga, roteamento e políticas de segurança, também são executadas como aplicativos de rede centrais na SDN. Também, algumas funções mais recentes, como engenharia de tráfego, virtualização de rede,

economia de energia, QoS (Qualidade de Serviço) de ponta a ponta, gerenciamento de mobilidade em redes sem fio, são realizadas. A variedade de aplicativos de rede, que podem ser desenvolvidos de forma rápida e fácil por meio de APIs e linguagens de programação, é um dos argumentos mais fortes para a adoção de uma SDN. (COSTA et al., 2016).

2.1.2 Interface Norte

Sendo um canal de comunicação entre o plano de aplicação e o controlador, a interface norte fornece uma visão abstrata da rede. A sua principal função é converter os requisitos das aplicações de gerenciamento em instruções de baixo nível e transmitir estatísticas, geradas nos dispositivos da rede e processadas pelo controlador, sobre a rede. Esta interface contém o A-CPI (*Application-Control Plane Interface*) que permite que aplicativos usados por administradores de rede controlem e monitorem funções de rede sem ter que ajustar detalhes de comunicação (RAMOS; KREUTZ; VERISSIMO, 2015).

2.1.3 Plano de Controle

No plano de controle estão os controladores, responsáveis por programar e gerenciar o plano de dados. Considerado como o cérebro da rede, pois toda a lógica da aplicação passa por ele para chegar ao plano de dados, este plano é responsável por criar funcionalidades para tomar decisões operacionais sobre o comportamento da rede (COMPUTACAO; GOMES, 2013) . Neste plano, o sistema operacional de rede (Network Operating System - NOS), fornece abstrações, serviços principais e APIs para desenvolvedores. O NOS oferece controle lógico para o próprio controlador. Assim, um desenvolvedor não precisa mais saber dos detalhes do processo de transmissão de pacotes para definir políticas de rede, situação que facilita seu desenvolvimento para diminuir o risco de erro.

O NOS geralmente, instalado em um computador de porte médio, tem como funções principais: mecanismos de segurança, recebimento, processamento e encaminhamento de eventos, descoberta de dispositivos conectados à rede, a análise do estado da rede, fornecimento de informações sobre a topologia, distribuição de configurações da rede, gerenciamento de dispositivos, encaminhamento de dados pelo caminho mais curto. Recursos de segurança são de primordial importância, pois eles possibilitam isolamento e aplicação de regras entre serviços e aplicações. Mas como limitação. o NOS não consegue fornecer uma interface comum para as camadas superiores, enquanto permite que uma plataforma de controle use diferentes *Southbound APIs* e *plug-ins* de protocolos, permitindo, assim, além de gerenciar diferentes dispositivos da rede, oferecer compatibilidade entre diferentes versões (STANCU et al., 2017).

2.1.4 Interface sul

Sendo o elemento vital para a separação entre os planos de controle e dados, a interface sul é responsável pelo controle de anúncios, operações de transferência (*forwarding*), relatórios estatísticos e notificações de eventos. Também considerada uma API, é por meio dela que os controladores SDN comunicam os requisitos das aplicações, reprogramando os equipamentos para realizar diversas funções, adicionando ou excluindo regras nas tabelas de fluxo. Entre outras funções, são: sistemas de detecção de intrusão (*Intruder Detection System - IDS*), *Firewall*, controle de fluxo, além de roteamento e comutação (RAMOS; KREUTZ; VERISSIMO, 2015), (KOPONEN et al., 2010).

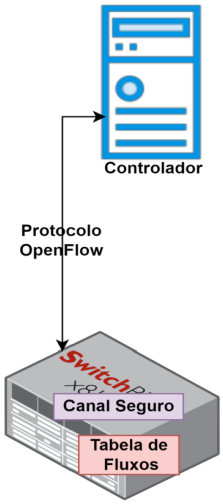
2.1.5 Plano de Dados

Este plano é composto por infraestrutura de rede (equipamentos como switches, roteadores e pontos de acesso) (RAMOS; KREUTZ; VERISSIMO, 2015). Ele é responsável pelo encaminhamento, descarte, encapsulamento, envio dos pacotes na rede para processamento ou para alguma tabela especial, monitoramento de informações locais e coleta as estatísticas. Este plano cominuca com o plano de controle via API *southbound*, por exemplo, OpenFlow (LARA; KOLASANI, 2014), *ForCES* (ISLAM et al., 2020) ou *NetConf* (STANCU et al., 2017). A implementação mais conhecida dessa API é o protocolo OpenFlow, que define um método genérico para permitir que o controlador interaja com os dispositivos de encaminhamentos da rede.

2.1.6 Switch OpenFlow

OpenFlow é um protocolo amplamente adotado na SDN, que estabelece a comunicação entre switches OpenFlow e um controlador logicamente centralizado. Inicialmente, SDN e OpenFlow começaram apenas como experimentos acadêmicos, mas nos últimos anos, ganharam força na indústria (LIU et al., 2019b). O switch Openflow é um equipamento com mecanismos para funções do protocolo Openflow. A Figura 2.2 ilustra a comunicação entre um switch OpenFlow e o controlador.

Figura 2.2 – Protocolo OpenFlow



Fonte: do próprio autor do trabalho (2023)

Um switch OpenFlow é composto por uma ou mais tabelas de fluxo (*flow table*), um grupo de tabelas de fluxo (*group table*), uma tabela de medidas (*meter table*) e um ou mais OpenFlow canais. Em geral, existem switches Openflow puros nos quais os pacotes de rede são processados no pipeline de execução do Openflow; e switches Openflow híbridos nos quais, os pacotes podem ser processados dentro do pipeline Openflow ou fora como em um switche normal. segundo (MAZUR et al., 2018), as principais características de um switch OpenFlow são:

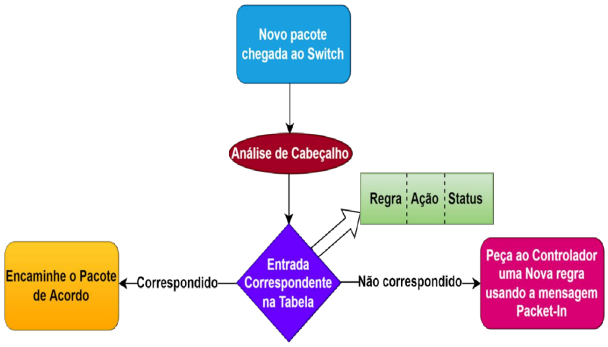
a) *flow table*: esta tabela dentro do switch, contém regras de fluxo, que definem a transferência ou não dos pacotes. Se tiver uma correspondência entre essas regras e as características do cabeçalho do pacote, uma ação é tomada, caso contrário, o pacote é enviado ao controlador para tomar uma decisão após algum processamento.
b) *group table*: utilizado para processar fluxos de pacotes de forma mais complexa, o *group table* é um conjunto de *Flow Tables*.
c) *meter table*: na tabela do medidor (*meter table*), são armazenados os dados de monitoramento de fluxo de rede, que consistem no número de pacotes e são usados para calcular a QoS *(Quality of Service)* da rede.

É necessário entender o funcionamento dos mecanismos que compõem o protocolo em uma rede SDN, depois de estabelecê-lo.

A Figura 2.3 monstra o funcionamento do processamento típico de um switch OpenFlow. A cada pacote que ingressa na rede, o switch OpenFlow realiza a separação dos cabeçalhos e verifica se tiver alguma entrada correspondente em suas tabelas de fluxos. Caso exista, o switch aplica a ação

correspondente de acordo com as regras armazenadas. Do contrário, ele envia o pacote por meio do canal seguro, para o controlador e aguarda as instruções.

Figura 2.3 – Processamento típico num switch ao receber um novo pacote

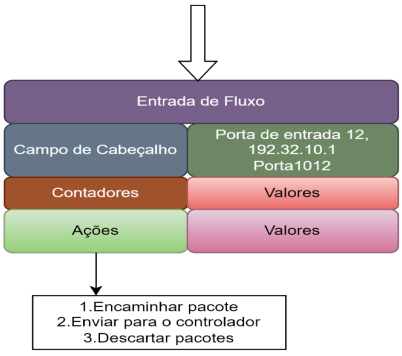


Fonte: adaptado de Singh e Behal (2020)

De acordo com (FRANKS et al., 2009), existe um conjunto de entradas de fluxo, nas tabelas de fluxo do switch OpenFlow. Cada entrada está associada a uma ação, como encaminhar, descartar e enviar pacotes ao controlador. A versão 1.0 do OpenFlow, possui três campos de verificação: cabeçalho (regras), ações e contadores; na tabela que define como devem ser tratados os pacotes que chegam ao switch. Cada campo tem um conjunto específico de entradas. O campo de ação pode indicar o comportamento a ser tomado para determinado fluxo. No campo Cabeçalho é possível observar os atributos relacionados aos fluxos que entram na rede, como o protocolo, o tamanho dos pacotes, o IP de origem, o IP de destino, a porta de origem, a porta de destino, entre outros.

É possível hoje ter uma grande flexibilidade do protocolo OpenFlow a partir da versão 1.3, devido à disponibilidade de diversas tabelas de fluxo. Consequentemente, é possível adicionar ações ao pacote em uma tabela e, se necessário, transferi-lo para outra tabela de fluxo, onde novas ações podem ser adicionadas (ANDERSON et al., 2014). A Figura 2.4 mostra uma tabela de fluxos do switch OpenFlow.

Figura 2.4 – Tabela de fluxos do switch OpenFlow



Fonte: Adaptado de Foundation (2012)

Um fluxo pode ser identificado por uma combinação de campos de cabeçalho de pacote e correlacionado a um conjunto de ações. As regras são formadas a partir desses campos. Na versão 1.0 do protocolo, existem 12 campos *(porta de ingresso, MAC-org/dest, ethernet-org/dest, VLAN-ID/prioridade, IP-org/dest/prot, porta-org/dest)* que podem ser usados no cabeçalho. O processamento de pacotes é definido por essas ações. Os contadores armazenam estatísticas sobre os fluxos do switch, como o tempo decorrido em que o fluxo foi instalado no switch, o número de pacotes, e os bytes transmitidos e recebidos. Esses dados podem ser passados para o controlador de forma coordenada, por meio de um temporizador, que solicita o conjunto atual de estatísticas do comutador para um determinado fluxo ou grupo de fluxos.

OpenFlow Channel é o canal de comunicação entre o switch e o controlador da rede usando o protocolo OpenFlow. Por ele, o controlador envia comandos para atualizar, salvar e deletar as regras da *flow table*. Devido ao crescimento da SDN, muitas grandes empresas como Google, Facebook, Microsoft, Verizon e Deutsche Telekon começaram a financiar a *Open Networking Foundation* (ONF) para promoção e adoção da SDN (SINGH; BEHAL, 2020).

2.1.7 Controladores

No plano de controle de uma arquitetura SDN estão os controladores que fornecem comunicação entre dispositivos de rede e aplicações SDN. Usando o protocolo Openflow, o controlador transmite comandos para equipamentos de rede: existe, portanto, a possibilidade de gerenciar nós de rede. Existem vários tipos de controladores, por exemplo:

• NOX: desenvolvido em C++, o NOX foi o primeiro controlador desenvolvido para SDN, e surgiu com o protocolo Openflow. É rápido e permite entradas e saídas assíncronas (AMIRI; ALIZADEH; REZVANI, 2020). Operando no conceito de fluxo de dados, ele verifica o primeiro pacote de cada fluxo e procura uma entrada correspondente na tabela de fluxo para aplicar uma determinada ação. A interface do controlador gira em torno de eventos, em que um evento corresponde a mudança de fluxo. Para a execução de um determinado evento, o Nox utiliza um conjunto de manipuladores para lidar com esses eventos. Graças a um conjunto de bibliotecas que fornecem implementações de funções comuns de gerenciamento de rede, o Nox é capaz de classificação rápida de pacotes, um módulo de roteamento e uma rede filtrada por políticas. O valor de retorno de um manipulador indica ao controlador se a execução deve ser interrompida ou continuar passando para o próximo manipulador (GUDE et al., 2008). Funcionando com a versão 1.0 do protocolo OpenFlow, e graças à existência de interfaces, este controlador tem sido amplamente aceito pelos pesquisadores; o que permite que o mesmo ambiente seja utilizado

em situações que exijam alto desempenho e nos casos em que seja necessário um desenvolvimento que simplifique o código resultante.(LARA; RAMAMURTHY, 2014).

• POX: nesta pesquisa, o controlador POX foi escolhido para os experimentos, devido à sua facilidade de aprendizagem com os algoritmos de ML. Ele é uma evolução do NOX (GUDE et al., 2008), e foi desenvolvido em Python. Fornece uma plataforma para o desenvolvimento e prototipagem rápida de aplicações de rede. Tem sua arquitetura baseada no seu antecessor o NOX, por meio dele é possível configurar o switch openflow, como hub. Operando apenas na versão 1.0 do protocolo OpenFlow, este controlador sofre de uma limitação das aplicações que podem ser desenvolvidas aproveitando a evolução deste protocolo. No entanto, ele fornece um meio eficiente de implementar o OpenFlow.

• RYU: baseado em componentes que podem ser alterados e estendidos para criar novas aplicações, o controlador RYU (*Ryu sdn framework*) é escrito em Python e possui uma interface REST (*Representational State Transfer*) (MEENA; BUNDELE; NAWAL, 2020). Funcionando com o protocolo OpenFlow desde a versão 1.0 até a versão 1.5, este controlador é capaz de gerenciamento de eventos, mensagens e estado na memória, graças à uma série de bibliotecas reutilizáveis, como *NetConf* (STANCU et al., 2017) e *sFlow* (KOPONEN et al., 2010).

• OPENDAYLIGHT: desenvolvido em Java, o controlador *OpenDaylight*. É modular e possui interface gráfica e suporte para outras tecnologias como *OpenStack* (SUH et al., 2016). Baseado em uma arquitetura de microsserviços, permite aos usuários controlar aplicativos e protocolos. Também considerado um *framework* para aplicações SDN, visa criar uma API *northbound* padrão que pode ser usada para programar diferentes protocolos *southbound*, incluindo OpenFlow nàs versões 1.0 e 1.3. Caracteriza-se pela modularidade e flexibilidade, permitindo aos desenvolvedores selecionar a funcionalidade do controlador considerada necessária para atender às necessidades específicas (LARA; RAMAMURTHY, 2014).

A escolha do melhor controlador SDN depende dos objetivos do pesquisador e seus experimentos. Devido aos problemas como: compilação, manutenção, instalação e número significativo de dependências na montagem, variações do NOX surgiram: como NOX com Python e o POX. Importante observar que com software open source, o NOX ou o POX não conseguem gerenciar os componentes de várias infraestruturas virtualizadas. (*OpenStack*); ao contrário do RYU ou OpenDaylight. Podemos comparar os controladores pelas linguagens empregadas, desempenho, a versão do OpenFlow, e quanto á dificuldade de aprendizado, como ilustra a tabela 2.1.

Quadro 2.1 – Comparação dos Controladores

Controladores	NOX	POX	RYU	OPENDAYLIGHT
Linguagem	C++	Python	Python	Java
Desempenho	Alto	Baixo	Baixo	Alto
OpenFlow	1.0 (CPqD: 1.1, 1.2, 1.3)	1.0	1.0, 1.1,1.3,1.4	1.0, 1.3
Dificuldade de aprendizagem	Moderada	Fácil	Difícil	Difícil

Fonte: Adaptado de (BARROS, Abril de 2022)

2.2 Segurança em Redes SDN

A programabilidade das redes SDN graças ao software favorece a criação e implementação de soluções de segurança (muito recorrentes) para controlar todos os aspectos da rede. No entanto, centralizar esse controle é um ponto único de falha que envolve questões de segurança como desempenho da rede, escalabilidade, que podem afetar toda a rede se essa mesma segurança for comprometida (HELLER; SHERWOOD; MCKEOWN, 2012). Consequentemente, a segurança da rede SDN é ainda um desafio por causa da sua própria vulnerabilidade, principalmente aos ataques DDoS e da ausência de confiança entre os componentes da rede.

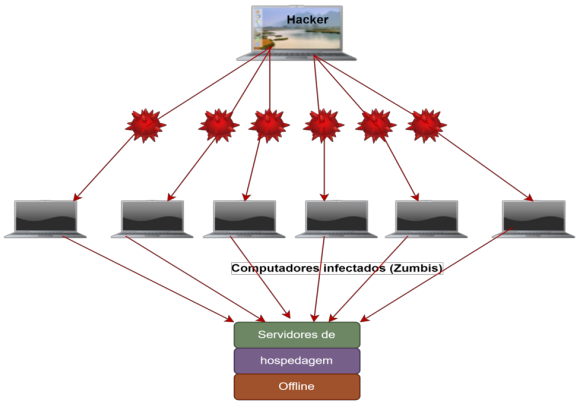
2.2.1 Ataques DDoS em SDN

Hoje em dia, um ataque comum no qual a rede ou aplicação fica temporariamente indisponível para os seus usuários é o de negação de serviço distribuído (DDoS - *Distributed Denial of Service*) (NAGPAL; YADAV, 2017). É um tipo de ataque projetado para impedir que um usuário legítimo acesse um site ou serviço da Web, como um ataque DoS. Ao contrário do ataque DoS, o ataque DDoS não é feito de um computador central, mas de vários computadores para uma única fonte (BAWANY; SHAMSI; SALAH, 2017).

Num ataque DDoS, o intruso encontra vulnerabilidades na rede e injeta um programa malicioso, conhecido como (*Trojan Horse-Cavalo de Tróia*), nos computadores sem o conhecimento dos usuários. Ao replicar este programa malicioso em vários dispositivos conectados à rede, o intruso cria um exército de computadores comprometidos, que eles controlam, para iniciar ataques DDoS (SINGH; BEHAL, 2020). Esses dispositivos são chamados de *"zumbis"* sendo que a rede criada por esses dispositivos é chamada de *"botnet"*.Uma rede privada de máquinas formada por um vírus de computador é chamada de *botnet*. Essas máquinas executam comandos enviados por um centro de controle que monitora o *botnet* (BEBIS et al., 2013).

Assim, toda a *botnet* está remotamente sob o controle de um operador humano chamado *botmaster*. Para lançar um ataque de DDoS, o atacante envia comandos para todas as máquinas comprometidas. Ela consiste em maliciosamente criar um tráfego excessivo em um espaço curto de tempo. Dependendo do número de dispositivos comprometidos (milhões hoje em dia), a vítima provavelmente terá seu computador sobrecarregado (exaustão dos recursos). Consequentemente, os recursos da vítima não são acessíveis a usuários legítimos por causa da inundação na rede e redução na largura de banda da aplicação. Assim a vítima está sob um ataque DDoS (SINGH; BEHAL, 2020), (LAU et al., 2000). A Figura 2.5 ilustra um ataque DDoS:

Figura 2.5 – Técnica usado pelo invasor num ataque DDoS



Fonte: do próprio autor do trabalho (2023)

Assim, uma rede SDN pode sofrer ataques de DDoS que podem ocorrer nos planos de dados e controle (MATTOS; DUARTE; PUJOLLE, 2016).

O plano de dados pode ter esgotado os recursos de largura de banda dos switches da rede ou as entradas da tabela de fluxo por um host mal-intencionado que gera fluxos falsificados para arquivos. Entre o controlador e os switches, pode ocorrer uma negação de serviço no canal de comunicação. O invasor pode esgotar a capacidade de processamento do controlador enviando muitos pacotes com cabeçalhos diferentes, que não correspondem a nenhuma das regras de fluxo já definidas. Esses pacotes serão então transmitidos ao controlador, responsável pelo processamento.

Os autores Shin et al. (2013) mostram como os ataques de DDoS em redes SDN exploram a separação dos planos de dados e controle. O invasor falsifica o endereço IP do switch SDN, de forma que as mensagens de controle podem ser enviadas usando esse switch com o endereço modificado. Quando um switch estabelece uma conexão com o controlador e se comunica, ao mesmo tempo, um segundo switch malicioso (switch com IP falsificado com o mesmo hardware e nome) será ativado e estabelecerá uma conexão com o controlador. O controlador encerrará a conexão com um switch legítimo e se co-

municará com o switch malicioso, degradando gradualmente o desempenho da rede. Este ataque pode causar solicitações falsas ao controlador e deve ser eliminado por meio de algum mecanismo (SINGH; BEHAL, 2020).

Diante dos desafios acima, podemos dizer que a segurança em uma rede SDN continua sendo uma preocupação, principalmente com ataques DDoS, apesar das inúmeras vantagens oferecidas por esta rede. A persistência deste desafio de segurança deve-se principalmente às vulnerabilidades existentes entre as camadas (dados e plano de controle). Assim, a rede SDN também sofre com os mesmos ataques das redes tradicionais. As técnicas frequentemente utilizadas em redes com arquitetura tradicional para detectar esse tipo de ataque podem ser aplicadas ao ambiente SDN com as adaptações que a nova arquitetura impõe à rede.

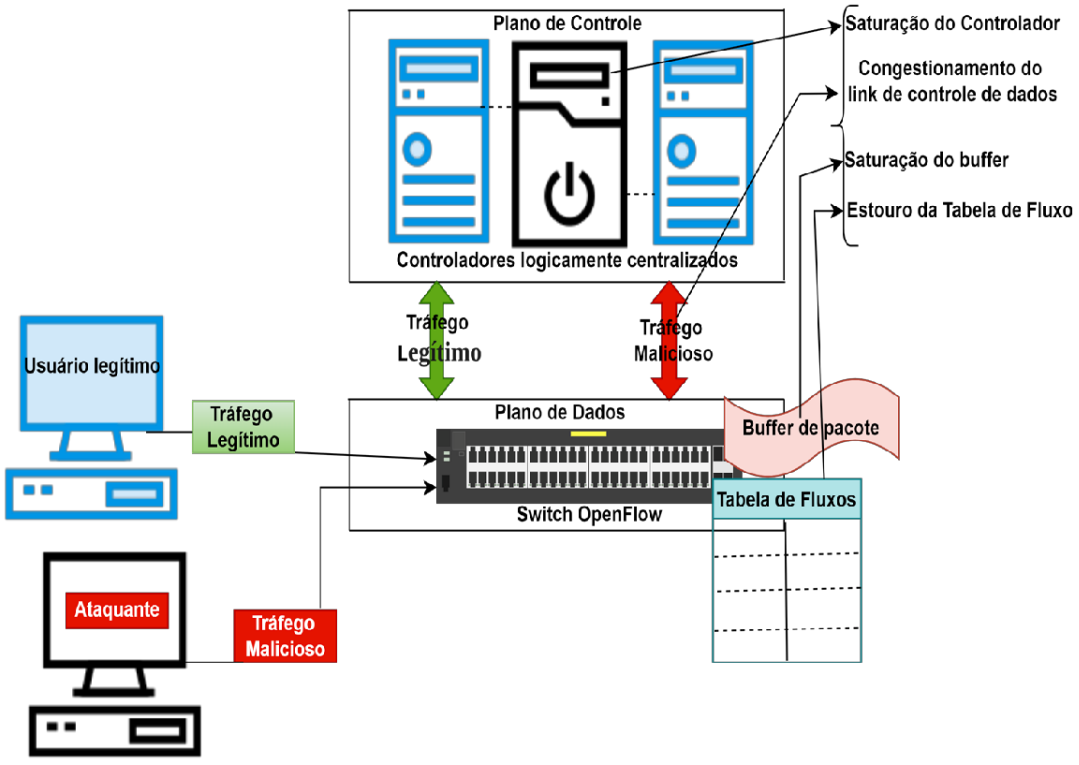
2.2.2 Vunerabilidade da SDN a ataques de DDoS

As vulnerabilidades da SDN a ataques de DDoS podem ser consequência de vários fatores, como, por exemplo:

a) Na SDN, novas políticas inovadoras são complicadas de implementar nos switches tradicionais, pois esses dispositivos são específicos do fornecedor (BEBIS et al., 2013);
b) Normalmente, o controlador é responsável por autenticar aplicativos e autorizar os recursos necessários para os aplicativos como isolamento, monitoramento e auditoria adequados. Portanto, é necessário separar os aplicativos de acordo com suas implicações de segurança antes de conceder acesso a qualquer recurso. Portanto, deve haver verificação de segurança personalizada para diferentes tipos de aplicativos nas APIs norte do controlador. Essa personalização ainda não foi demonstrada (SINGH; BEHAL, 2020);
c) São necessários vários mecanismos que podem inspecionar com eficácia os aplicativos SDN antes de executá-los. Da mesma forma, proteger o controlador de vulnerabilidades causadas pelo aplicativo é uma área de estudo posterior (BAWANY; SHAMSI; SALAH, 2017);
d) Algumas implementações de controladores hoje não são capazes de processar um grande número de novos fluxos usando OpenFlow em redes de alta velocidade com links de 10 Gbps. Nesse ponto, a incapacidade de escalabilidade do controlador se torna uma ameaça à segurança. Portanto, a falta de escalabilidade do controlador o torna a escolha preferida de ataques DoS e DDoS (SINGH; BEHAL, 2020);
e) A falta de confiança entre os nós da rede pode afetar a estrutura da rede, pois as aplicações executadas no controlador podem se comportar de maneira maliciosa (MATTOS; DUARTE; PUJOLLE, 2016).

A Figura 2.6 mostra o mapeamento conceitual de vulnerabilidades na arquitetura SDN. Neste mapeamento, o hacker mal-intencionado, acessa um componente da rede (switch OpenFlow) sobrecarregando, causando travamento e lentidão, impossibilitando o acesso dos usuários/dispositivos legítimos. Como o controlador é responsável pelo principal serviço para controle da rede, ele acaba se tornando um alvo interessante. O hacker realiza diversas solicitações de novas entradas de fluxo aos dispositivos de encaminhamento (o switch Openflow), causando saturação de buffer (*buffer saturation*) e estouro da tabela de fluxo (*flow table overflow*). Assim, o controlador fica superlotado de solicitações (*controller saturation*), esgotando seus recursos de processamento, causando congestionamento do link de controle de dados (*data-control link congestion*), travamento, lentidão e consequentemente negando o serviço de instalações de regras de fluxo aos componentes da rede (SINGH; BEHAL, 2020).

Figura 2.6 – Mapeamento conceitual de vulnerabilidades na arquitetura SDN



Fonte: adaptado de Singh e Behal (2020)

Por causa da vulnerabilidade da SDN, ela torna-se vítima de alguns ataques. O Quadro 2.2 mostra alguns exemplos dos ataques por causa da vulnerabilidade da SDN:

Quadro 2.2 – SDN como vítima de ataques DDoS

Saturação do Switch	Os switches precisam armazenar em buffer parte do campo de cabeçalho do pacote. O switch OpenFlow tem memória limitada para armazenar as informações e por isso pode ser saturada.
Estouro da tabela de fluxo	Switches que têm memória TCAM (Ternary Content Addressable Memory) para armazenar as tabelas de fluxo. Essa memória é limitada nos switches devido ao alto custo e ao consumo de energia da memória.
Congestionamento de controle e link de plano de dados	Para cada novo fluxo de tráfego, o switch solicita novas regras de fluxo usando mensagens *Packet in* do controlador e recebe a resposta usando mensagens *Packet out*, que causam alto tráfego no link.
Saturação do controlador	Para cada switch na rede, o controlador centralizado é responsável por uma nova regra de fluxo. Múltiplas solicitações maliciosas podem sobrecarregar o controlador.

Fonte: do próprio autor do trabalho (2023)

2.3 Aprendizado de Máquina (*ML - Machine Learning*)

O Aprendizado de Máquina (*ML - Machine Learning*), é uma subárea das áreas de estudo em IA (*Inteligência Artificial*) na Ciência da Computação. O objetivo do ML é gerar agentes (modelos) inteligentes por meio da detecção de padrões em dados. Em ML, são desenvolvidos vários tipos de algoritmos, baseados em técnicas, como estatística, probabilidades e biologia (PRAKASH; PRIYADARSHINI, 2018), (LAI; TAN, 2019). O ML é uma ciência e uma arte que permite que os computadores programados aprendam com os dados fornecidos a eles.

No processo de ML, os computadores podem ser treinados usando os dados fornecidos a eles (conjunto de treinamento) e podem mostrar seu desempenho em dados diferentes (conjunto de teste). Desta forma, o problema é resolvido minimizando a intervenção humana (administrador da rede). O ML é usado em lugares onde os métodos clássicos são ineficientes. As áreas de uso podem ser listadas da seguinte forma:

a) Pode interpretar dados muito grandes e complicados;
b) Pode resolver problemas complexos para os quais os métodos tradicionais não conseguem encontrar soluções;
c) Os métodos de ML podem encontrar soluções para situações em que as soluções existentes exigem muita intervenção externa / atualização sem intervenção externa;
d) Pode funcionar em ambientes variáveis. Os métodos de ML podem ser aplicados a uma nova situação examinando os dados (BEBIS et al., 2013).

É necessário fazer o pré-processamento dos dados antes do processamento dos algoritmos de ML, identificando se estão normalizados, balanceados ou com poucas amostras, padronizados, para evitar durante o treinamento dos algoritmos, problemas de *overfitting* ou *underfitting*. O *overfitting* ocorre quando o algoritmo treina ao ponto de não conseguir generalizar para outras entradas de dados que não sejam do treinamento. No *underfitting* o algoritmo treina ao ponto de não aprende efetivamente, ou seja, não conseguir acertar as amostras de treinamento (YIN; VAUGHAN; WALLACH, 2019).

Os algoritmos de ML podem ser divididos em duas categorias: os algoritmos supervisionados e não supervisionados. Eles são diferenciados pelo método de execução e pelas características dos dados. O método supervisionado consiste em inserir dados que permitem induzir o aprendizado do algoritmo, para a indução do resultado esperado. Essa indução é feita com os dados classificados de acordo com o resultado esperado. Os problemas de classificação e regressão são as dificuldades encontradas por este método. No método não supervisionado que trabalha com dados não rotulados, o algoritmo agrupa os dados em grupos distintos por cálculos de similaridade, portanto, também pode ser usado para classificação, agregação e redução de dimensionalidade de dados (RUSSEL et al., 2020).

O método empregado na solução para classificação de tráfego legítimo e malicioso foi não supervisionado. Os algoritmos não supervisionados utilizados neste trabalho são descritos a seguir. O algoritmo *Naive-Bayes*, é um método estatístico que classifica os dados com base no conhecimento prévio obtido dos dados de treinamento e combina a probabilidade de um evento ocorrer com a probabilidade de que um evento já tenha ocorrido (NANDA et al., 2016). É um algoritmo ML simplificado com a adição da condição de independência ao teorema de Bayes (BEBIS et al., 2013), (HARTIGAN, 1985). O teorema de Bayes é expresso pela seguinte equação:

$$P(\frac{a}{b}) = \left\{ P(\frac{b}{a}) * \frac{P(a)}{p(b)} \right\} \tag{2.1}$$

Na qual:

$$P(\frac{a}{b})\text{: A probabilidade de ocorrência de (a) no caso de (b) ocorrências.} \tag{2.2}$$

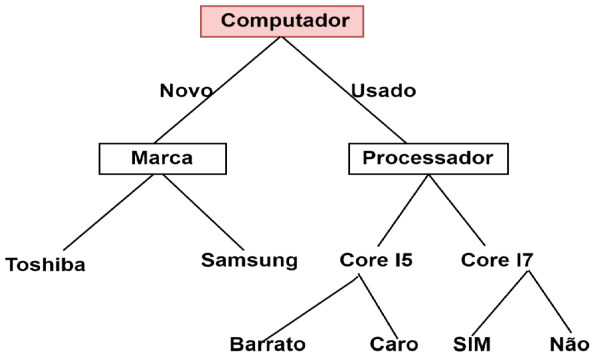
$$P(\frac{b}{a})\text{: A probabilidade de ocorrência de (b) no caso de (a) ocorrências.} \tag{2.3}$$

P (A) e P (B): As probabilidades anteriores de (a) e (b).

Deve-se observar que a falta de amostras de um dado atributo neste algoritmo, torna a probabilidade correspondente ou posterior nula (como zero). Mas esse problema pode ser corrigido com o estimador de Laplace (PENG et al., 2019).

O algoritmo de árvore de decisão usa um método de classificação heurística guloso que alcança a classificação de dados com indução de regras. Ele constrói uma estrutura em árvore que permite visualizar as regras, começando do nó raiz até as folhas. (BREIMAN et al., 2017).

Figura 2.7 – Árvore de decisão de comprar um computador (Decision Tree)



Fonte: do próprio autor do trabalho (2023)

Para a construção dessa árvore de decisão são necessários critérios de seleção dos atributos que contenham os melhores registros correspondentes ao nó da árvore em questão. Para isso, é considerada a capacidade informativa do atributo, determinada pelo cálculo da entropia. (SHANNON, 1949). Essa Entropia é uma medida da aleatoriedade de uma variável que pode ser expresso na seguinte equação:

$$Entropia(s) = \sum_{i=1}^{m} -Pi \log Pi \tag{2.4}$$

Na qual:

S é um conjunto de exemplos

m é o número de classes

Pi é a proporção de S que pertence á classe i, tendo:

$$Pi = \frac{|Si|}{|S|} \tag{2.5}$$

Na qual:

|Si| é o número de exemplos classificados na i-ésima partição

|S| é o número total de exemplo do conjunto S

KNN (*K Nearest Neighbor*): é um método baseado em amostra, sendo um dos algoritmos de ML mais usados com sua estrutura simples e rápida. KNN identifica a classe de novos dados que não são classificáveis usando dados de treinamento de tipo de classe conhecido. Essa determinação é feita

observando-se os vizinhos mais próximos da nova amostra, para os quais nenhuma classificação é especificada. Em um plano com N propriedades, o número de vizinhos a serem observados para uma amostra não classificada é especificado pelo número K. Para a amostra desconhecida, as distâncias aos vizinhos são calculadas e os menores K números são escolhidos a partir desses valores de distância. A propriedade mais repetida dentro dos valores K é atribuída como propriedade de instância desconhecida (PENTIKOUSIS; WANG; HU, 2013). A escolha adequada da medida de distância a ser utilizada depende do problema a ser resolvido. A distância euclidiana é a mais utilizada, entre muitas outras, e é descrita pela seguinte equação:

$$D_E(p,q) = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2} \tag{2.6}$$

Na qual:

$$p = (p_1, ..., p_n) \tag{2.7}$$

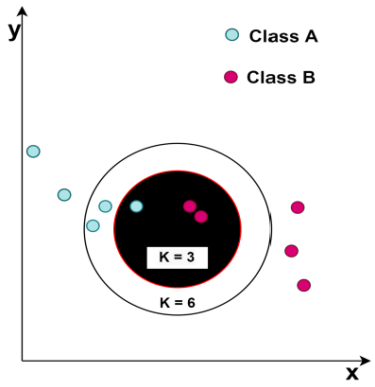
e

$$q = (q_1, ..., q_n) \tag{2.8}$$

são dois pontos n-dimensionais.

Na Figura 2.8, um visual é criado para os valores 3 e 6 de K em um plano tridimensional (N = 3)

Figura 2.8 – Operação do algoritmo KNN para os valores K=3 e K=6

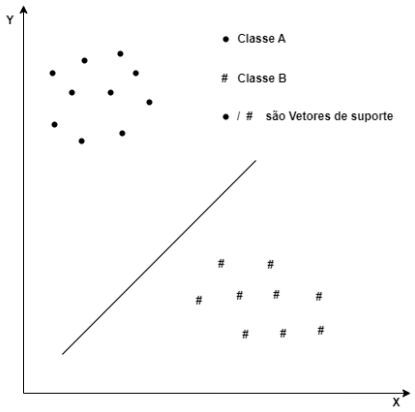


Fonte: do próprio autor do trabalho (2023)

Support Vector Machine: Um dos métodos de aprendizado supervisionado que pode ser usado para classificação ou regressão é o SVM *(Support Vector Machine)*. Este algoritmo funciona com dados linearmente separáveis ou não linearmente separáveis (WOLPERT; MACREADY et al., 1995). O seu funcionamento difere dependendo dos tipos de dados. Em dados linearmente separáveis ele classifica as

amostras e calcula diversos hiperplanos para separar as classes, além da reta é considerado uma margem entre a reta e as classes, assim a reta que melhor divide as classes é a que possui a maior margem (SANJAA; CHULUUN, 2013). Os dados que ficam em cima da margem são chamados de vetores de suporte. A Figura 2.9, mostra um exemplo do funcionamento da SVM com dados linearmente separáveis.

Figura 2.9 – SVM em dados linearmente separáveis



Fonte: do próprio autor do trabalho (2023)

O hiperplano de separação é dado pela equação:

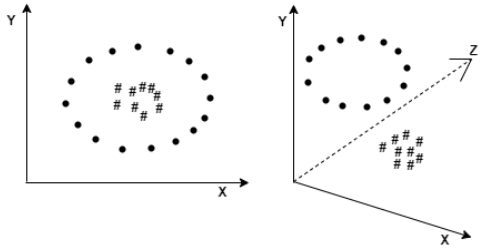
$$f(x) = w(x) + b = 0 \tag{2.9}$$

Na qual:

w é o vetor de pesos (mesma dimensão das amostras) perpendicular ao hiperplano de separação e b é um escalar.

A SVM utiliza o artifício de *kernel Trick* que é feito por meio de uma função que pode ser linear, polinomial, sigmoidal ou Gaussiana. Neste *kernel Trick*, são adicionadas mais de dimensão nos dados, quando o processo de classificação lida com dados não linearmente separáveis mudando o seu plano e os cálculos serão para encontrar um hiperplano que melhor divida as classes. A Figura 2.10 mostra um exemplo do funcionamento da SVM com dados não lineares.

Figura 2.10 – SVM em dados não linearmente separáveis



Fonte: do próprio autor do trabalho (2023)

No próximo capítulo, são apresentados os trabalhos relacionados.

3 TRABALHOS RELACIONADOS

Apesar de fornecer soluções modernas em SDN, o plano de controle e o plano de dados não são amplamente explorados com datasets reais para detectar novos tipos de ataques DDoS, mais comuns em redes de computadores. De um lado, os ataques DDoS são uma das principais ameaças em Sistema de Detecção de Intrusão IDS (*Intrusion Detection System*). De outro lado, o plano de conhecimento KP foi criado para melhorar o tráfego de pacotes dentro da arquitetura tradicional SDN. Na literatura atual existem vários trabalhos relevantes sobre a importância da análise e classificação dos dados; necessária para alcançar um resultado satisfatório, e encontrar soluções estruturadas. Neste capítulo, identificamos e discutimos trabalhos de pesquisa que abordaram detecção, mitigação de ataques DDoS em ambientes SDN e trabalhos que abordaram os planos de conhecimento e de gerenciamento.

3.1 Ataques DDoS na SDN

No artigo de Li et al. (2018) foi proposto um modelo baseado em SVM (*Support Vector Machine*) para detectar ataques de DDoS em SDN. O modelo extrai vários parâmetros dos pacotes, e mede a distribuição de cada parâmetro usando entropia e, em seguida, usa o algoritmo SVM treinado para detectar o ataque de DDoS. Experimentos feitos com o Mininet e o controlador *Floodlight* simularam três ataques de DDoS com diferentes proporções de tráfego malicioso. Outros algoritmos de ML, como Árvore de decisão, *Naive bayes, KNN, Random Forest* também foram usados para análise e detecção de DDoS. Os autores puderam observar que o SVM obteve o melhor resultado e pôde detectar ataques DDoS com maior precisão. Para testar a capacidade de mitigar o ataque DDoS, eles usaram a visão global SDN para localizar a porta do switch que estava lançando o ataque DDoS e coordenar com o controlador para atualizar a tabela de fluxo do switch para mitigar o ataque DDoS. O resultado mostrou que este método pode mitigar os ataques DDoS em tempo real e em larga escala. É importante notar que os tempos utilizados a cada vez são relativamente curtos para a eficiência dos experimentos. Além disso, o artigo carece de conclusões concretas dos resultados.

Os autores (SAHOO; TIWARY; SAHOO, 2018) escolheram o controlador POX e o mininet para realizar os experimentos. A métrica empregada é baseada na teoria da informação, como *General Entropy (GE) e Generalized Information Distance* (GID). As informações da tabela de fluxo dos switches OpenFlow foram usadas para identificar DDoS de alta taxa de ataque durante os FEs (*flash event*) que são semelhantes ao ataque DDoS de alta taxa. Eles têm poucas diferenças paramétricas e alguns dos principais recursos, como a semelhança entre os fluxos, o comportamento de acesso à URL (*Uniform Resource Locator*), a duração do impacto, podem discriminar entre esses dois eventos. Observaram que

essas duas métricas (GE, GID) podem distinguir o tráfego de ataque do tráfego legítimo. Se o valor da entropia for menor que o limite, é assumido que o ataque está em andamento no controlador. Em cada caso, até 2000 pacotes/s são enviados ao controlador. Para um limite adequado, variam o tráfego normal e de ataque por meio da ferramenta *Scapy*, comparam simultaneamente diferentes taxas de entradas de pacotes que chegam ao controlador. Para ataques DDoS de alta taxa, continuaram uma taxa de ataque de 80% e a taxa de ataque foi definida. O ponto negativo neste artigo é a falta de um resultado concreto, com a ausência de uma definição exata de limiar exato de forma mais dinâmica em um cenário de tráfego real para detecção.

Os autores (WANG et al., 2019) propõem o SGS (*Safe-Guard Scheme*), um esquema que contém dois estágios: detecção de tráfego de anomalia no plano de dados e defesa dinâmica dos vários controladores do plano de controle para proteger-los. Um dos controladores atua como o controlador mestre, responsável por processar o envio de solicitações de fluxo dos switches, enquanto os outros controladores desempenham funções de escravo e estão em estado inativo em condições normais. Uma vez detectados os ataques DDoS, o controlador mestre envia mensagens de controle de acesso para os switches, enquanto os controladores escravos são ativados e realizam remapeamento dinâmico para mitigar os ataques DDoS. As métricas usadas são: *byte rate (br), symmetric flows percentage (sfp), variation rate of asymmetric flow (vafr), flows percentage with small amount packets (fpsa)* com o algoritmo *back propagation neural network (bpnn)*. Os resultados do desempenho da SGS em relação a vários esquemas básicos mostram que a SGS pode rapidamente detectar e reagir a ataques DDoS. O tempo de configuração do fluxo foi reduzido em $30,4\%$ em média, e o tempo de resposta do controlador da SGS foi reduzido em $42,1\%$. O teste foi feito no simulador Mininet, com a utilização de múltiplos controladores; o que pode causar problemas de sincronização e prejudicar a flexibilidade da rede, tornando-o lento porque todos não são apropriados para lidar com uma rede de rápido crescimento.

Os autores (XIAO et al., 2019) propuseram um método de descoberta de ataque de DDoS em SDN, usando um modelo de grafo de padrão de recursos *(FPG-Feature-pattern graph)*. Criaram assinaturas de ataque adaptadas ao ambiente SDN através da criação de fluxo com o dataset da rede tradicional e tráfego SDN. Um grafo de padrão de recurso foi modelado baseado nas assinaturas de ataque DDoS, onde cada nó representa um padrão de rede e as arestas ponderadas denotam a semelhança baseada em recursos dos nós. Para melhor descrever as relações entre os parâmetros de rede, introduziram o aprendizado de peso dos enlaces (links). O aprendizado da métrica de distância foi usada para preencher a matriz M e a distância de *Mahalanobis* foi calculada com os valores de M usando-se a semelhança entre nós ou vetores de características. O modelo de grafos é escalável com o propósito de métodos de atualização local e global para inserir novos nós. Para otimizar o desempenho da detecção de ataques

DDoS baseados em FPG, apresentaram também um modelo que é uma combinação de dois métodos. Um calcula a distância entre o fluxo recém-chegado e os nós no grafo para encontrar o nó mais próximo; o outro calcula as distâncias entre o novo fluxo e os recursos em cada nó. Os resultados dos experimentos simulados, demonstraram que o método de descoberta baseado em FPG para comportamentos de ataque DDoS supera substancialmente os métodos de entropia e ML comparados em precisão, *Recall e F-score* nesse trabalho. Os resultados da atualização do grafo atestaram a eficácia das abordagens de atualização local e global; mas infelizmente sem menção dos valores dos resultados dos cálculos. Foi observada a ausência de seletor de contramedidas automatizado para facilitar a seleção de contramedidas com base nos padrões de tráfego. Também, a latência e o uso de recursos computacionais foram muito altos.

Os autores (SEN et al., 2020) apresentam um novo método para detecção de DDoS baseado em ML, implementado em um *testbed* para SDN. Foi empregado o *AdaBoosting* com *decision stump* como classificador fraco para treinar o modelo em um conjunto de dados de rede privada em ambiente SDN simulado. Este modelo mostrou até 93% de precisão de detecção com uma baixa taxa de falsos positivos. Apesar desse resultado ser satisfatório, é importante observar que foi usada uma rede virtual de pequeno tamanho e não uma rede real. Uma sobrecarga no processamento dos fluxos usando um classificador realmente fraco pode causar pontos de falhas que deixam esse trabalho inadequado para uma rede real e densa.

Os autores (UJJAN et al., 2020), apresentaram o modelo de co-detecção de aprendizado profundo com *Snort* IDS (*Intrusion Detection System*) no plano de controle, para otimizar a precisão da detecção de DDoS de nós IoT (*Internet of Things*) indesejados, com a ajuda de sFlow e amostragem de tráfego baseada em sondagem no plano de dados. Foram realizados experimentos no simulador Mininet, com um modelo baseado em *deep learning* para mitigar diferentes tipos de ataques DDoS comuns. A avaliação do sistema proposto demonstra maior precisão de detecção com 95% de taxa de verdadeiros positivos com menos de 4% de taxa de falsos positivos na implementação baseada em sFlow em comparação com a pesquisa adaptativa. O tempo do experimento é relativamente curto (20 s), para um tráfego de ataque DDoS de baixa taxa, que demandaria um tempo de análise maior.

Os autores (GADZE et al., 2021) pesquisaram os ataques DDoS de inundação (TCP, UDP e ICMP) que visam o controlador (*Floodlight e switches OpenFlow*). Para detecção e mitigação, eles usaram modelos baseados em aprendizado profundo, memória de longo prazo (*LSTM-long-short term memory*) e rede neural convolucional (*CNN-convolutional neural network*). Eles compararam o desempenho de modelos de aprendizado profundo com modelos clássicos de ML. A partir de experimentos realizados no Mininet, demonstraram que o RNN LSTM (*recurrent neural network-LSTM*) é um algoritmo de aprendizado profundo viável que pode ser aplicado na detecção e mitigação de ataques DDoS

no controlador SDN. O modelo proposto produziu uma precisão de 89,63%, que superou modelos lineares como *SVM-Support Vector Machine* 86,85% e *Naive Bayes* 82,61%; embora o KNN (*K-Nearest Neighbor*), que é um modelo linear, tenha superado o modelo proposto (atingindo 99,4% de precisão). No entanto, este modelo oferece um bom compromisso entre precisão e Recall, o que o torna adequado para classificação DDoS. Apesar disso, observa-se a restrição da coleta de conjuntos de dados e a falta de análise aprofundada da seleção de recursos e ajuste de hiper-parâmetros; o que pode dificultar a obtenção de um melhor desempenho ao usar modelos de rede neural.

Os autores (YUNGAICELA-NAULA; VARGAS-ROSALES; PEREZ-DIAZ, 2021) apresentaram uma solução usando inteligência artificial para detectar dois tipos de ameaças: ataques DDoS na camada de transporte e na camada de aplicação. Foram usados vários modelos de *machine learning (ML) e deep learning (DL)* a partir de uma arquitetura modular baseada em SDN, com componentes que podem ser modificados ou aprimorados separadamente, oferecendo flexibilidade para testar diferentes métodos inteligentes para detectar vários ataques. A exploração de vários métodos de ML/DL permitiu determinar quais métodos funcionam melhor em diferentes tipos e condições de ataque. Eles testaram os modelos de ML/DL para os conjuntos de dados CICDoS2017 e CICDoS2019, (*Canadian Institute for Cybersecurity-DoS*), que apresentaram uma precisão superior a 99% na classificação de tráfego invisível (conjunto de teste). Eles também implantaram um ambiente simulado usando o *Mininet Network Emulator* e o controlador ONOS *Open Network Operating System*. Nesta configuração experimental, eles obtiveram nesse controlador, altas taxas de detecção de mais de 98% para ataques DDoS de transporte e até 95% para ataques DDoS de camada de aplicação e concluíram que os modelos GRU (*gated recurrent units*) e LSTM (*long short-term memory*) mantiveram as taxas de detecção mais altas na avaliação do modelo online. No entanto, nota-se um baixo índice de escalabilidade para a solução e ausência de uma estratégia de mitigação otimizada.

Os autores (SCARANTI et al., 2022) propuseram um sistema de detecção de intrusão IDS *(Intrusion Detection System)* baseado em *Clusters* para detectar ataques numa rede SDN, aproveitando a entropia dos endereços e portas IP de origem e destino. Usaram o algoritmo *DenStream* como um *kernel IDS* dentro do controlador *Floodlight*, baseado em Java. O IDS proposto foi avaliado em termos de desempenho de detecção, do atraso no reconhecimento de um ataque e de informações sobre o comportamento de cada infecção. Para criar uma comparação crítica, eles consideraram o desempenho do IDS proposto com *Half-Space-Trees (HS-Trees)*, um algoritmo de classificação de fluxo de classe única. Sua proposta visa evitar a demanda por rotulagem e conhecimento prévio, a fim de fornecer um método prático e preciso para lidar com cenários online reais. Além disso, esta proposta abre caminho para uma análise abrangente ao projetar a estrutura do *Cluster* no espaço de características, fornecendo informa-

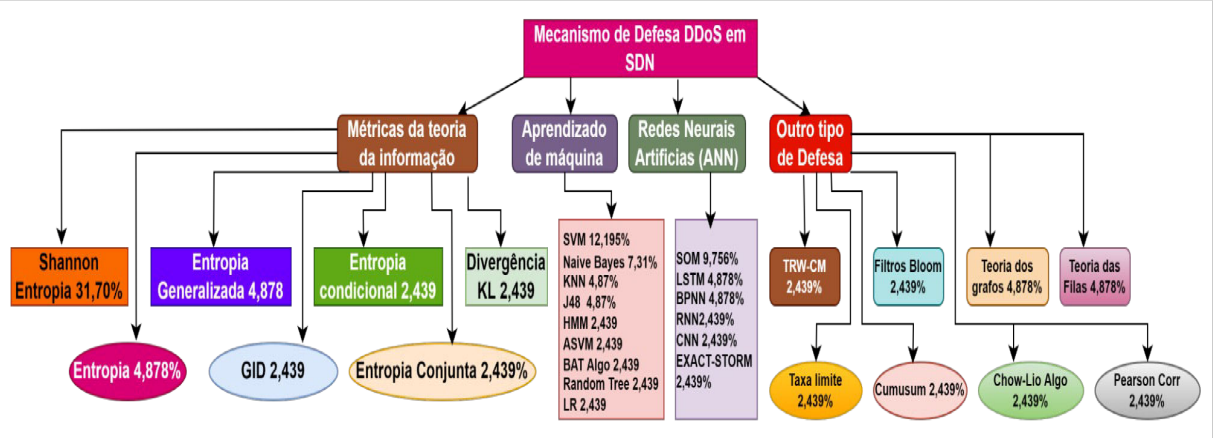
ções sobre a intensidade, sazonalidade e tipo de ataque. Foram realizados experimentos com o algoritmo *DenStream* em vários bancos de dados de DDoS e *portscan* com diferentes intensidades, durações e padrões sobrepostos. Ao comparar o desempenho do *DenStream* com *Half-Space-Trees*, foi possível expor a capacidade da proposta não supervisionada, superando a solução de uma classe e alcançando taxas de F-measure superiores a $99,60\%$. Os experimentos foram realizados usando o Mininet. Foi observado o tempo experimental foi relativamente curto e que a limitação de um *DenStream* ocorre em ataques que não alteram seus recursos e podem não ser detectados corretamente pelo IDS.

Segundo os autores (FOULADI; ERMIŞ; ANARIM, 2022), as abordagens de detecção de ataques DDoS existentes para SDN são baseadas principalmente em abordagens estatísticas (baseadas em limites) e em aprendizado de máquina (ML). Considerando as características dinâmicas do tráfego de rede, eles afirmaram que encontrar um limite dinâmico é de alguma forma problemático. Por outro lado, encontrar um recurso apropriado que possa discriminar o ataque DDoS do tráfego normal é um desafio para abordagens baseadas em ML. Portanto, para resolver isto, os autores propuseram um esquema de detecção e contramedida de ataques DDoS baseado em transformada wavelet discreta (DWT-*discrete wavelet transform*) e rede neural *auto-encoder* para SDN. O esquema proposto extrai características estatísticas da transformada wavelet para serem processadas por uma rede neural *auto-encoder* para detectar amostras de tráfego de ataque DDoS. Posteriormente, para reduzir a carga computacional imposta pelo modelo de rede neural, a taxa média de acertos na tabela de fluxo dos switches foi utilizada para ativar a detecção DDoS do esquema. Primeiramente, eles apresentam uma análise da complexidade dos custos computacionais para os algoritmos SVM e KNN apresentados neste estudo. Então, para avaliar o desempenho da detecção de ataques DDoS, eles implementaram seu esquema em uma rede SDN de exemplo usando o ambiente GNS3 (*Graphical Network Simulator-3*) e o emulador Mininet contra amplificação de DNS, protocolo de tempo de rede e ataques de inundação, usando TCP SYN. Também forneceram uma análise detalhada do desempenho considerando a complexidade de custo computacional dos algoritmos propostos no esquema e a avaliação da taxa de detecção bem-sucedida com simulações. Os resultados da simulação mostram que seu esquema supera as abordagens anteriores com uma taxa de detecção significativamente alta (100%) e uma taxa de alarmes falsos notavelmente baixa 0%.

Como observação geral, embora vários trabalhos tenham proposto a detecção de ataques DDoS, a maioria deles não utilizou conjuntos de dados atualizados ou que contenham as ameaças mais recentes. Além disso, foi observado que poucos trabalhos avaliaram suas soluções usando cenários de redes reais ou usando dispositivos em hardwares reais. Por outro lado, observamos que as técnicas propostas têm várias limitações em relação à escalabilidade, grandes conjuntos de dados, precisão, dados complexos, resultados lentos, etc. Assim, notamos que apenas pesquisas preliminares são realizadas para otimizar a

segurança nos planos de (dados e controle). A detecção de ataques DDoS é um problema difícil devido ao fato de que os switches habilitados para OpenFlow possuem menos informações devido a fluxos de tráfego isolados. Mas, acima de tudo, as pesquisas permitiram avanços significativos na detecção, classificação e mitigação dos ataques DDoS. Assim, organizamos na Figura 3.1 as soluções de detecção de DDoS existentes em SDN em quatro categorias diferentes, baseadas em: Teoria da Informação, Aprendizado de máquina, Rede neural artificial e outros tipos. De acordo com (SINGH; BEHAL, 2020) para esta classificação foram utilizadas as métricas de Entropia (e suas variações), divergência e GID. Também, a Tabela 3.1 mostra a síntese de soluções de defesa DDoS nesta seção dos trabalhos relacionados.

Figura 3.1 – Classificação de abordagens de defesa de ataque DDoS em SDN



Fonte: Adaptado de (SINGH; BEHAL, 2020)

A mitigação de ataques DDoS também é um aspecto crucial para proteger os recursos de rede sob ataque. Os pesquisadores usaram muitas técnicas como migração de conexão, migração de pacote, limitação de influências de largura de banda, ajustando tempos limite e controlador para controlador e protocolos de comunicação para mitigar os ataques DDoS em redes baseadas em arquitetura SDN.

Quadro 3.1 – Soluções de defesa DDoS em SDN

Autores / Anos	Alcance	Algoritmo / métricas	Parâmetros	Plano	Controlador	Conjunto de dados	Pontos chave
Li et al 2018	Detecção Mitigação	Entropia de Shanon SVM	IP e porta de origine/destino	Controle	Floodlight	Sintético usando entropia	Detecção e mitigação de ataques de inundação DDoS usando técnicas de entropia
Sahoo et al 2018	Detecção	Entropia Generalizada	IP e porta de origine/destino	Controle	POX	Sintético usando Scapy	Métricas da teoria da informação e coleta estatística no controlador para detectar anomalia
Xiao et al 2019	Detecção	Teoria dos Grafos	IP e porta de origine/destino	Controle e Dados	RYU	Calcula de distância entre fluxo e recursos em cada nó	Uso de distância de Mahalanobis
Wang et al 2019	Detecção Mitigação	Back propagation neural network	IP e porta de origine/destino	Controle	POX	Safe-Guard Scheme	Uso de byte rate (BR), symetric flows percentage
Sen et al 2020	Detecção	ML- Accuracy	IP e porta de origine/destino	Controle	POX	Sintético usando accuracy	AdaBoosting com decision Stump num conjunto de dados de rede privada em SDN
Ujjan et al 2020	Detecção Mitigação	Aprendizado profundo	-	Controle	OVS / sFlow	Sintético usando Snort IDS	Uso de Deep learning para mitigar ataques de DDoS
GADZE et al 2021	Detecção Mitigação	LSTM / CNN	IP e porta de origine/destino	Controle	Floodlight	Sintético usando RNN / LSTM	Aprendizado profundo e rede neural convolucional comparado com ML
Yungaicela2021	Detecção	ML e Deep learning	IP e porta de origine/destino	Controle	ONOS	CICDoS 2017 e CICDoS 2019	ML e DL e concluíram que os modelos GRU e LSTM têm as taxas de detecção mais altas
SCARANTI et al 2022	Detecção	Entropia de SVM	IP e porta de origine/destino	Controle	Floodlight	Detecção de intrusão baseado em Clusters	Uso de Half Space Trees para classificar Denstream em vários bancos de dados DDoS
Fouladi et al2022	Detecção	SVM, KNN	IP e porta de origine/destino	Controle	POX	Sintético usando Rede Neural	Uso de Wavalet, transforme rede neural auto-encoder para SDN
Este trabalho	Detecção Mitigação	SVM, KNN, ÁRVORE de decisão, Naive B	IP/MAC/porta de origine/destino, s/pFlow	Controle, KP,Gerenciamento	POX	Bonesi e script de T50, geração de três Datasets para treinar Algoritmos	Detecção e mitigação de ataques de DDoS usando classificadores

Fonte: do próprio autor do trabalho (2023)

3.2 Planos de conhecimento e de Gerenciamento na SDN

Os autores (CLARK et al., 2003) foram os primeiros a propor o Plano do conhecimento KP (*A Knowledge Plane for the Internet*). O objetivo deste plano é tornar a rede ciente do que está sendo requisitado, para que possa autonomamente tomar decisões básicas, em vez de depender de ações do administrador da rede. Para melhorar o transporte dos fluxos de dados, uma alternativa foi projetar uma nova camada que crie, reconcilie e mantenha os muitos aspectos de uma visão de alto nível e, em seguida, forneça serviços e orientação conforme necessário para outros elementos da rede; isto é o plano de conhecimento, ou KP.

Os autores (MESTRES et al., 2017) exploram as razões para a falta de adoção do plano de conhecimento (KP). Entre outros motivos, é que as redes são sistemas de natureza distribuída, sendo uma dificuldade para a aplicação de ML, essencial para a técnica KP. Isso acontece porque cada nó da rede (switch, roteador) tem controle e visão parcial sobre todo o sistema. Os autores postulam também que a ascensão de dois paradigmas recentes podem facilitar a adoção do KP, como o desenvolvimento da *Software-Defined Networking* (SDN) e *Network Analytics* (NA). Esses novos paradigmas facilitam a adoção de técnicas de Inteligência Artificial (IA) no contexto de operação e controle da rede. Os autores descrevem um novo paradigma que acomoda e explora SDN, NA e IA e fornecem casos de uso que ilustram sua aplicabilidade e benefícios. São apresentados resultados experimentais simples, que suportam alguns casos de uso relevantes. Eles se referem a esse novo paradigma como *Knowledge-Defined Networking* (KDN).

No paradigma KDN, o KP aproveita os planos de controle e gerenciamento (*Management Plane*) para obter visão e controle avançados sobre a rede. Ele é responsável por aprender o comportamento da rede e, em alguns casos, operar autonomamente a rede. Fundamentalmente, o KP processa os dados de rede coletados pelo plano de gerenciamento, e os transforma em conhecimento via técnicas de aprendizado de máquina (ML), e usa esse conhecimento para tomar decisões. O paradigma KDN opera por meio de uma malha de controle para fornecer automação, recomendação, otimização, validação e estimativa. Os autores abordaram o plano de gerenciamento que normalmente opera em escalas de tempo maiores. Esse plano define a topologia de rede e lida com o fornecimento e configuração de dispositivos de rede; responsável por monitorar a rede para fornecer análises críticas. Coleta informações de telemetria do plano de controle e dados, mantendo um registro histórico do estado e dos eventos da rede.

Os autores (LÓPEZ-RAVENTÓS et al., 2019) combinaram planos de dados, de controle e de conhecimento (KP) para fornecer controle automatizado de uma rede sem fio. Nesse trabalho, o paradigma SDN é identificado na forma como a rede é orquestrada, já que o plano de controle é gerenciado por um controlador que se comunica e solicita diferentes informações dos pontos de acesso sem fio por meio do protocolo OpenFlow. Os autores usaram um servidor, no qual os dados são armazenados e os algoritmos de ML executados. O servidor é conectado ao controlador para aproveitar ao máximo as estatísticas da rede para tomar decisões. Através dos resultados dos algoritmos de ML, o processo de tomada de decisão de acordo com o conhecimento obtido pode ser conduzido diretamente pelo KP de forma autônoma com base em um conjunto de regras pré-definidas. No topo do controlador, são executadas aplicações de rede para dar ao controlador as diretivas para o gerenciamento da rede. Assim definido, o SDWN (*Software Defined Wireless Networking*) é uma proposta de combinação do SDN com o KP, aplicada para redes sem fio. As aplicações da camada de aplicação podem fazer alterações na rede ou monitorar a rede através do controlador via a interface norte.

Os autores (GHOSH et al., 2022), propuseram uma abordagem multi-camada de KDN para definir regras de roteamento na rede que é concebida para segregação operacional e funcional. A filosofia SDN de desacoplar o controle e o plano de dados tem sido o principal princípio do projeto para a arquitetura proposta. No entanto, o plano do Conhecimento KP foi integrado na parte superior da arquitetura para suportar recursos SON (*Self Organized Networking*). A arquitetura suporta automação de rede e SON. O KP encontra a rota ideal usando o algoritmo *MRoute (Self-Optimization)*, em seguida, os instala em nós subjacentes, enviando a configuração específica do dispositivo para os dispositivos de borda (autoconfiguração) e garante uma rota mais confiável, mantendo-os atualizados ao longo do tempo (autocura). Assim, atende a todos os três critérios de SON. A abordagem é subdividida em cinco camadas: plano de Infraestrutura ou dados, *Overlay*, Sobreposição ou aplicação, Controle e Conhecimento.

A proposta deste trabalho de dissertação, é uma extensão do trabalho de "VICTOR BOELL 2021"[1], que estudou a "IDENTIFICAÇÃO DE TRÁFEGO MALICIOSO EM REDES SDN"com técnicas de inteligência computacional. Os algoritmos de IA considerados foram: *Naive Bayes, Multi Layer Perceptron, Random Forest, Decision Tree, Support Vector Machine, K-nearest Neighbors algorithm, K-means e Gradient Boosting*. Os experimentos foram realizados no Mininet, e o tráfego, foi gerado, usando o Nmap, que envia diversos pacotes de redes e analisa a resposta do servidor para identificar qual serviço ou porta aberta está funcionando. Para a busca de vulnerabilidades foram usadas as ferramentas NESSUS e Nikto, para, respectivamente, manter uma base de dados e fazer diversas requisições no site com padrões de ataques a aplicações web. A ferramenta T50, foi usada para envio de grande quantidade de diversos tipos de pacotes de rede, os quais variaram entre vários protocolos. O tipo de ataque gerado foi baseado em flooding (DoS, um ataque de única origem). O melhor resultado foi obtido com o algoritmo de Gradient Boosting, com a taxa de acurácia de até 98% durante o treinamento e até 100% de acerto na classificação de tráfego malicioso. Apesar disso, os datasets empregados não foram específicos de ataques DDoS para treinar os algoritmos de ML. Mas os experimentos foram realizados no Mininet com um tempo de tráfego relativamente curto para os cenários.

Quadro 3.2 – Trabalhos relacionados ao KP

Autores	**Alcance**	**Algoritmo**	**Parâmetros**	**Planos**	**Cenários**	**Pontos chaves**
Clark et al 2003	Proposta do KP	-	-	KP	-	Descarregar o Controlador
Mestre et al 2017	Detecção	Técnivas de DL	IP- origine / destino	KP	Mininet	Detecção no KP
Lopez et al 2019	Detecção/ Mitigação	Técnivas de ML	IP / MAC origine/destino	Controle e KP	Mininet	SDWN
Ghosh et al 2022	Detecção	Técnivas de ML	IP- origine / destino	Controle e KP	Mininet	SON

Fonte: do próprio autor do trabalho (2023)

[1] http://repositorio.ufla.br/jspui/handle/1/46815

4 METODOLOGIA

Este capítulo apresenta os planos de conhecimento KP e de gerenciamento PG, na arquitetura proposta, na qual, alguns módulos são descritos para análise estatística e outros para o processo de tomada de decisão (coleta de estatística, pré-processamento, detecção e decisão). Por outro lado, o capítulo apresenta a metodologia utilizada para realizar a experimentação deste projeto. Os experimentos foram realizados num ambiente real de laboratório com a arquitetura descrita. A execução de experimentos é baseada nos cenários propostos, gerando amostras de tráfego e resultados perfilados. São também descritos as ferramentas BoNeSi, um script de T50 (para gerar tráfego considerado legítimo), cenários dos experimentos, a plataforma de testes, o dataset e as métricas utilizadas.

Entre as soluções existentes, os métodos baseados em ML usam diferentes técnicas para detectar anomalia no sistema. Vários tipos de recursos de rede são levados em consideração ao classificar o tráfego. A detecção é baseada nesse padrão de aprendizagem comportamental.

4.1 Plano de conhecimento KP na SDN

O KP é uma nova camada que conta com aprendizado de máquina (ML) e técnicas cognitivas para operar a rede. Diz-se que um Plano de Conhecimento (KP) traz muitos benefícios para a rede, como automação (reconhecer-agir) e recomendação (reconhecer-sugerir), e tem o potencial de representar uma mudança de paradigma na forma como operar, otimizar e solucionar problemas das redes de dados. O desenvolvimento do plano de conhecimento fornece uma base para a implementação, coleta e correlação de dados para detecção de intrusão ou ataques que requerem que observações de vários pontos da rede estejam correlacionados, a fim de obter um resultado mais robusto e confiável (MESTRES et al., 2017).

O KP pode ser caracterizado como:

a) Distribuído - a funcionalidade KP para diferentes regiões da rede é fisicamente e logicamente descentralizada;
b) Ascendente - entidades simples podem compor em entidades maiores e mais complexas;
c) Dirigido por restrições - o sistema pode adotar ou não, qualquer comportamento que não seja especificamente restringido.

Com o KP, a rede vai do simples ao complexo, a integração de dados e conhecimento é uma função central do plano de conhecimento. O KP deve ser capaz de coletar, filtrar, reduzir e rotear observações, afirmações e conclusões de diferentes partes da rede para pontos nos quais são úteis. No caso deste trabalho de pesquisa, o KP é distribuído porque suas funcionalidades são fisicamente e logicamente descentralizadas

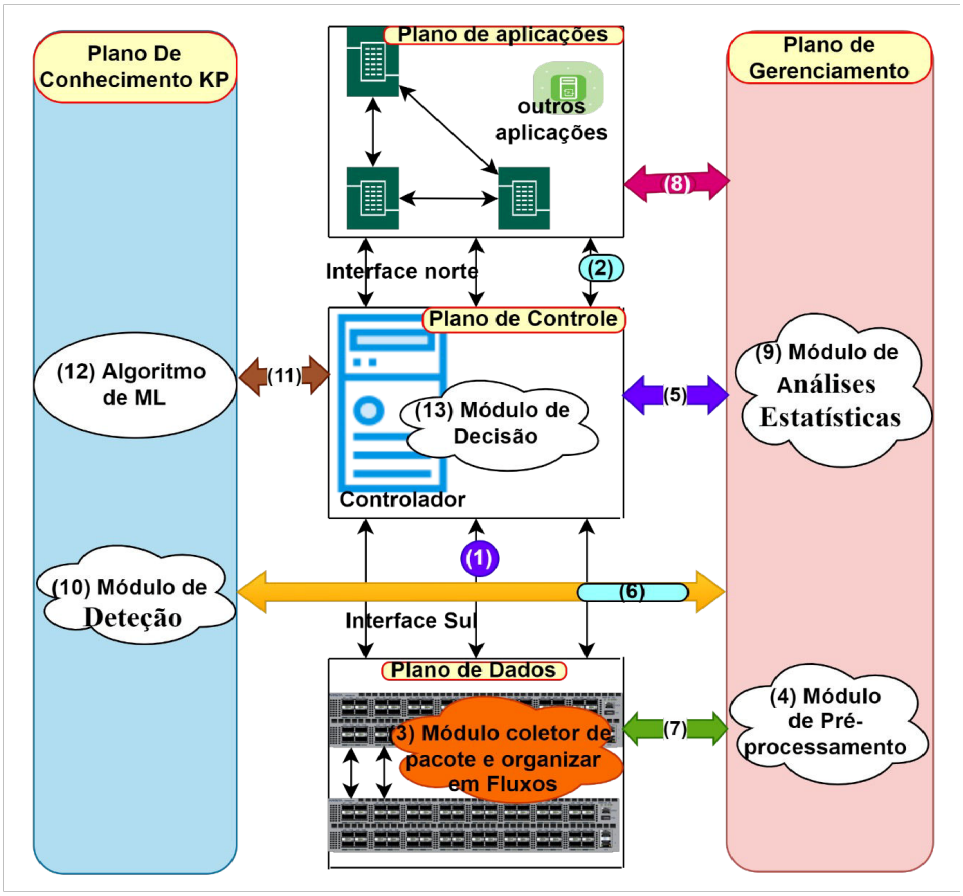
4.2 Plano de gerenciamento na SDN

O paradigma SDN que dissocia o plano de controle do plano de dados reduz a complexidade da rede. Os elementos do plano de dados de rede, como roteadores e switches, são equipados com recursos aprimorados de computação e armazenamento. Assim, o plano de gerenciamento é ortogonal aos planos de controle e dados, e opera normalmente em escalas de tempo maiores. O plano de gerenciamento pode definir a topologia de rede e lida com o fornecimento e configuração de dispositivos de rede, sendo também responsável por monitorar a rede e fornecer análises estatísticas. Além disso, o plano de gerenciamento pode coletar informações de telemetria do plano de controle e dados, mantendo um registro histórico do estado e dos eventos da rede.

4.3 Arquitetura Proposta da nova divisão de rede pelo plano de conhecimento KP e o plano de gerenciamento PG

A arquitetura proposta é dividida em cinco planos: dados, controle, aplicações, gerenciamento e conhecimento. A Figura 4.1 apresenta a nova arquitetura proposta. Nesta arquitetura são inseridos módulos para detecção e mitigação dos ataques DDoS.

Figura 4.1 – Arquitetura proposta



Fonte: do próprio autor do trabalho (2023)

A comunicação (1) entre o plano de dados e o plano de controle é feita pela interface sul, via protocolo OpenFlow. A comunicação (2) entre o plano de controle e o plano de aplicação, interface entre a SDN e o administrador da rede, é feita via A-CPI (*Application-Control Plane Interface*). O plano de aplicação, possui aplicações para monitoramento de tráfego, possibilidade de programar os equipamentos da rede para desempenhar várias funções, além da possibilidade de obtenção de informações sobre o estado da rede e do recebimento de notificações dos eventos.

No plano de dados, definimos um módulo (3) coletor de pacotes de dados de rede que organiza os fluxos e os transmite ao módulo de pré-processamento (4) que definimos no plano de gerenciamento. O projeto deste módulo é baseado numa aplicação coletor de fluxos, que junto com as regras das tabelas de fluxo monitoram os elementos desse plano em tempo real, enquanto eles encaminham os fluxos para acessar informações de tráfego refinadas.

O plano de gerenciamento, comunica com o plano de Controle via (5) para Monitorar o desempenho; com o plano de Conhecimento via (6), para análise de dados coletadas pelo plano de gerenciamento; com o plano de Dados via (7) para a configuração do elemento e com o plano de Aplicações (8) para

estabelecer SLAs (*Service Level Agreement*). Ainda no plano de gerenciamento, é desenvolvido um módulo de pré-processamento para reduzir a dimensão das características de fluxo. Para isso, a aplicação realiza a limpeza de dados e a análise de componentes principais. Esse plano, ortogonal ao plano de controle, pode consultar o controlador SDN, para obter o estado de controle e coletar informações (9). Pode também operar em escalas de tempo maiores, definir a topologia de rede e lidar com o fornecimento e configuração de dispositivos de rede e monitorar a rede para fornecer análises estatísticas.

No plano de conhecimento (KP) também ortogonal ao plano de controle, é desenvolvido um módulo de detecção (10) que emprega um modelo treinado com algoritmos, para classificar os fluxos de entrada pré-processados como normal ou malicioso. Esse plano KP processa as estatísticas coletadas pelo plano de controle via (11) para coleta de dados de controle, as transforma em conhecimento por meio dos algoritmos de aprendizado de máquina (12), e usa esse conhecimento para tomada de decisões. Este novo paradigma consiste em combinar planos de dados, controle, conhecimento e gerenciamento para fornecer controle de rede automatizado no plano de controle, no qual definimos o módulo (13) que envia as informações e decisões sobre fluxos classificados como maliciosos. Este módulo cria e visualiza um histórico de classificação dos fluxos recebidos do módulo de detecção.

Cada um dos módulos propostos é integrado a uma aplicação que é implementada nas diferentes camadas. Assim, o plano de conhecimento, junto com o plano de gerenciamento, aproveitam a estrutura da SDN para obter visão e controle avançados sobre a rede. O KP é responsável por aprender o comportamento da rede e, em alguns casos, operar automaticamente a rede de acordo com as decisões de controle. Para isso, o KP processa os dados de rede coletadas pelo plano de gerenciamento, sejam dados pré-processados ou dados brutos, os transforma em conhecimento via ML e o plano de controle usa esse conhecimento para tomada de decisões. Essa arquitetura pode reunir informações suficientes para oferecer uma visão mais completa da rede. Como benefício, essa separação de planos reduz a sobrecarga de processamento no controlador porque o KP e o PG são implementados em outros hosts para que o controlador possa cuidar da tomada de decisões e criação de novas regras. Além disso, essa arquitetura oferece flexibilidade e automação, pois permite melhorias na detecção e mitigação dos ataques DDoS, sem afetar o restante da arquitetura da SDN.

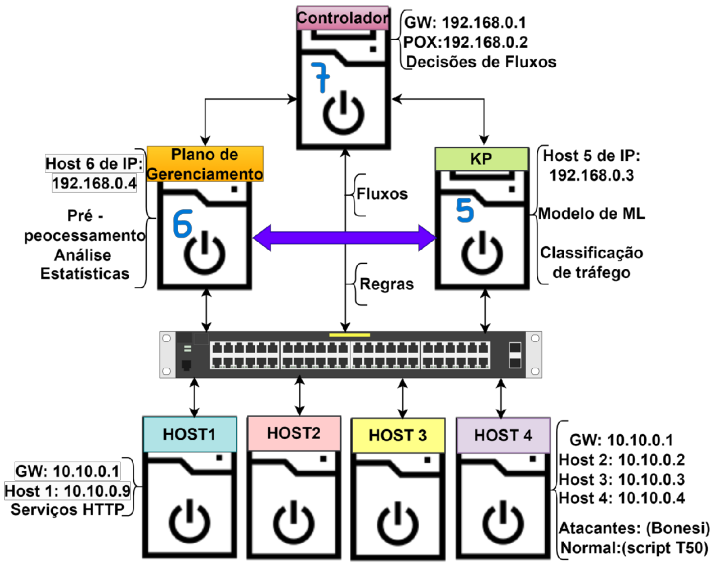
4.4 Definição dos experimentos

Os experimentos foram conduzidos utilizando-se um switch Extreme Summit x440 e sete (7) computadores conforme mostra a figura 4.2. Os Host (2, 3, 4) servem para injeções de tráfegos. Instalamos nesses Host (2, 3, 4) as ferramentas BoNeSi para geração de tráfego malicioso e um script de T50 para geração de tráfego considerado normal (legítimo). Quando entram pacotes (legítimo ou mali-

cioso), o módulo coletor organiza os pacotes em fluxos e envia esses fluxos de pacotes para o módulo de pré-processamento no Host 6. Portanto, limpeza de dados e redução de dimensionalidade podem ser realizadas. Assim, os recursos desses fluxos aumentam a complexidade do treinamento e os tempos de previsão do módulo detecção no Host 5 que recebe um fluxo pré-processado. Este módulo de detecção usa as informações pré-processadas para classificar os fluxos como ataques ou tráfego normal usando modelos de ML treinados.

O Host 7 no qual está definido o módulo de decisão recebe resultado da classificação para mitigação (caso de tráfego malicioso), encaminhamento (caso de tráfego normal) e definição de novas regras (caso de tráfego suspeito ou malicioso). O Host1 serve para rodar um servidor HTTP recebendo conexões HTTP na porta 9000 do switch e realizando ping nos HOST (2, 3, 4). A Figura 4.2 é uma topologia ilustrativa da interligação dos dispositivos seguindo a arquitetura proposta e a configuração da interligação dos dispositivos como VLANs, endereços IP e serviços, apresentados na Tabela 4.1. Nessa tabela, a Vlan-H é para os Host (1, 2, 3, 4, 5, 6) com o Gateway: 10.10.0.1 e Vlan-Ctrld é para o Host 7 (controlador) com Gateway:192.168.0.1

Figura 4.2 – Topologia ilustrativa da interligação dos dispositivos dos experimentos



Fonte: do próprio autor do trabalho (2023)

Quadro 4.1 – Configuração dos equipamentos do experimento

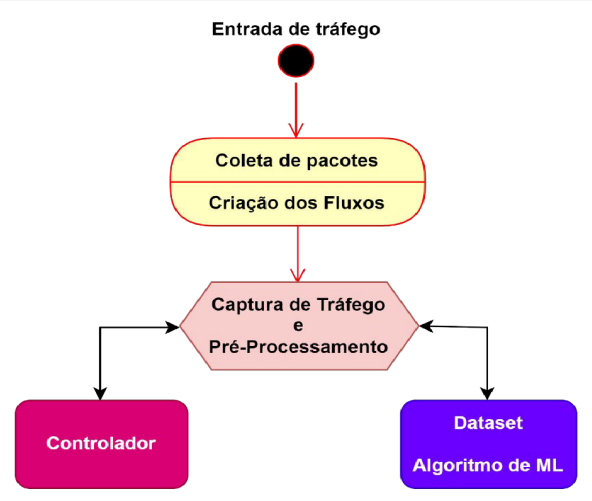
Host	VLANs	IP	GateWay	Função
Switch			10.10.0.1	VLAN de 16portas
HOST1	Vlan-H	10.10.0.9	10.10.0.1	Servidor HTTP, realiza ping nos Host (2, 3, 4)
HOST2	Vlan-H	10.10.0.17	10.10.0.1	Injeção de fluxos com T50 e BoNeSi
HOST3	Vlan-H	10.10.0.18	10.10.0.1	Injeção de fluxos com T50 e BoNeSi
HOST4	Vlan-H	10.10.0.19	10.10.0.1	Injeção de fluxos com T50 e BoNeSi
HOST5	Vlan-H	192.168.0.3	10.10.0.1	Detecção de Fluxos
HOST6	Vlan-H	192.168.0.4	10.10.0.1	Pre-processamento
HOST7	Vlan-Ctrld	192.168.0.2	192.168.0.1	Decisões

Fonte: do próprio autor do trabalho (2023)

4.5 Cenários realizados

Três cenários foram definidos para realizar os experimentos. O primeiro é de tráfego legítimo, o segundo, do tráfego malicioso, e no cenário 3, o teste foi híbrido (tráfego malicioso junto com legítimo). Para cada cenário, foi realizada uma captura do tráfego dentro da SDN proposta para observação do comportamento do módulo de detecção no Host 5 (KP). A Figura 4.3 mostra uma ilustração da Captura de tráfego.

Figura 4.3 – Ilustração da captura de tráfego

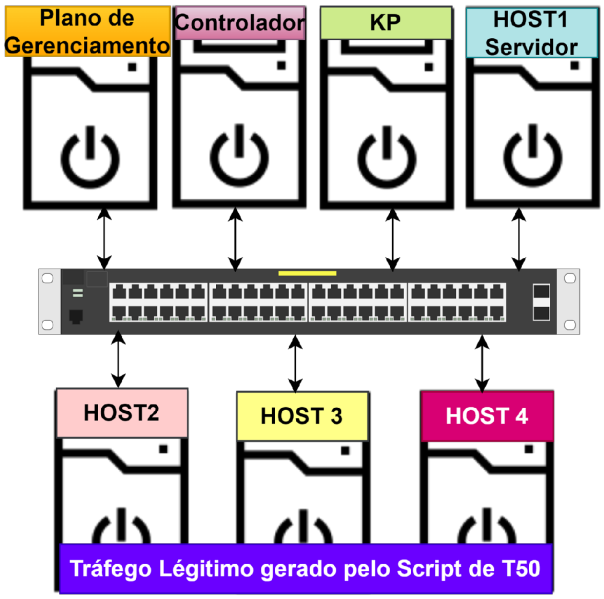


Fonte: do próprio autor do trabalho (2023)

No cenário 1, depois do treinamento dos algoritmos pelo dataset proposto, o primeiro teste foi do **Tráfego Legítimo** (normal) gerado pelo script de T50 instalado nos Host (2, 3, 4) que enviam pa-

cotes sobre o Host1 (servidor). O objetivo aqui é capturar o fluxo de tráfego no Host 5 (KP) após pré-processamento no Host 6 e observar o comportamento dos algoritmos de ML, e do bloqueio (ou não) dos tráfegos, baseando nos módulos implementados; caso haja confusão de tráfego legítimo com malicioso. A Figura 4.4 mostra uma ilustração.

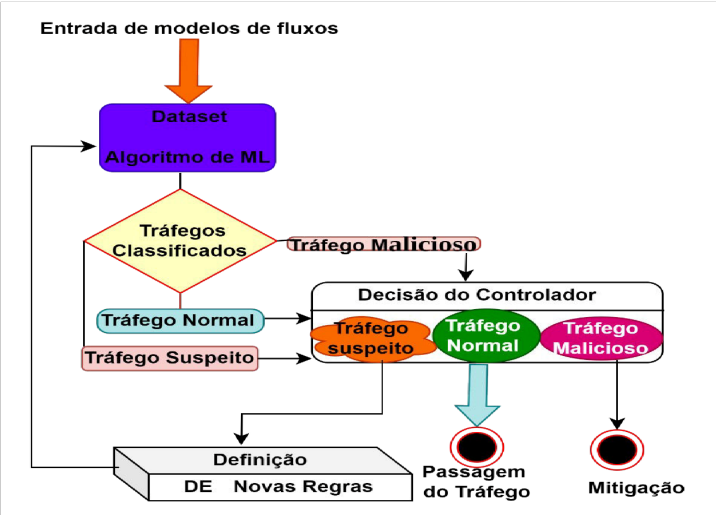
Figura 4.4 – Topologia ilustrativa do cenário 1-tráfego legítimo



Fonte: do próprio autor do trabalho (2023)

No cenário 2, o teste foi do **Tráfego Malicioso** pela ferramenta BoNeSi instalado nos Host (2, 3, 4) que enviam pacotes para o Host1 (servidor). O objetivo aqui é observar no Host 5 (KP), o comportamento dos algoritmos de ML junto com o módulo detecção, do módulo pré-processamento no Host 6, mas também do módulo decisão no Host 7; caso haja confusão de tráfego legítimo com malicioso, principalmente durante a classificação. Usamos os modelos escolhidos nesse cenário 2, para classificar um novo tráfego dentro da SDN, como um meio de validação dos mesmos. Assim, o melhor modelo foi obtido para classificar o tráfego. A Figura 4.5 mostra uma ilustração do Cenário 2.

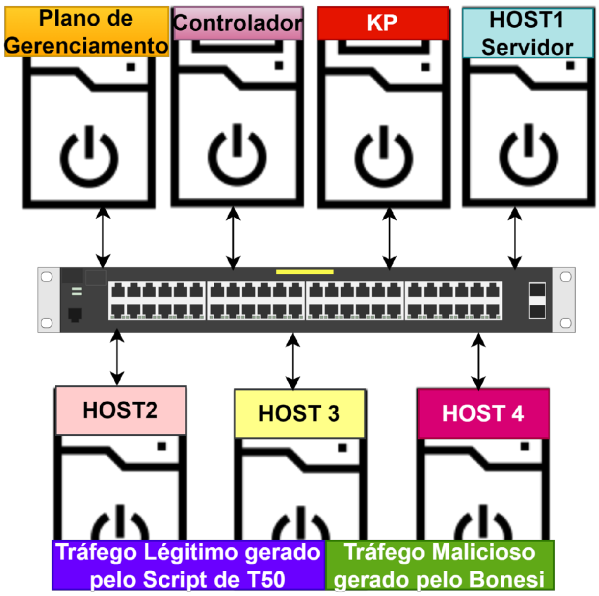
Figura 4.5 – Topologia ilustrativa do cenário 2-Classificação/Bloqueio de tráfego



Fonte: do próprio autor do trabalho (2023)

No cenário 3, o teste foi do **Tráfego Híbrido** (tráfego malicioso junto com legítimo). Os Host (2, 3) servem para atacar. O Host 4, para injeção do tráfego legítimo. Todos enviam pacotes para o Host1 (servidor). O objetivo aqui é observar também no Host 5 (KP), os comportamentos dos algoritmos de ML junto com o módulo detecção, do módulo pré-processamento no Host 6, mas também do módulo decisão no Host 7; caso haja confusão de tráfego legítimo com malicioso. A Figura 4.6 mostra uma ilustração.

Figura 4.6 – Topologia ilustrativa do cenário 3-tráfego híbrido



Fonte: do próprio autor do trabalho (2023)

4.6 Materiais utilizados

As características dos Hosts utilizados nos planos de controle, conhecimento, aplicação e gerenciamento para experimentos são CPU:

a) Processador (10ª geração de Intel® Core™ i7-10700 (8-core, cache de 16MB, 2.9GHz até 4.8GHz);
b) Placa gráfica UHD Intel® Graphics;
c) Memória de 8GB (1x8GB), DDR4, 2933MHz, expansível até 64GB (2 slots UDIMM, 1 slot livre);
d) SSD de 512GB PCIe NVMe M.2;
e) Com sistema operacional Ubuntu 16.0;
f) Configurada com 3 GB de RAM.

Usamos um switch Extreme Summit x440 EXTREME em um cenário real. Este modelo de switch possui 52 portas, CPU Single Core de 500 MHz e o sistema operacional ExtremeXOS 16.2.3.5. e 20 GB de armazenamento. O ExtremeXOS possui um conjunto robusto de protocolos de controle de Camada 2 e Camada 3, fornece uma arquitetura flexível para redes altamente resilientes e foi projetado para suportar IPv6. O ExtremeXOS é uma base de software altamente disponível e extensível para redes convergentes. Oferece alta disponibilidade para serviços de voz e vídeo, com proteção para os planos de gerenciamento e controle de rede.

Nesta pesquisa, foi utilizado o POX Controller, um controlador SDN com interface escrita em Python, e de código aberto. Este controlador (POX) oferece uma maneira eficiente de implementar o protocolo OpenFlow. Ele é distribuído e hospedado no github (PINTEREST, 2020). O OpenFlow instalado no caso desta pesquisa é da versão 1.0 e foram monitoradas as camadas L2 e L3. Para instalação e execução do POX, foi instalado o Python 3.10 e utilizado o sistema operacional Ubuntu 16.04.

4.7 Ferramentas

Entre outras ferramentas neste trabalho de pesquisa, o T-50 foi usado dentro do script para gerar uma frequência menor de pacotes para não ter um volume de ataques DOS. Assim, consideramos seus fluxos como legítimo. O BoNeSi foi usado no caso deste trabalho para gerar tráfego malicioso.

4.7.1 BoNeSi

De acordo com "Anonymous hacker", o BoNeSi é um simulador de botnet DDoS para diferentes tipos de protocolos. A partir de um tamanho de botnet definido (diferentes endereços IP), a BoNeSi

gera ataques de inundação ICMP, UDP e TCP (HTTP). É altamente configurável, incluindo volume de dados, endereços IP de origem, taxas, URLs e outros parâmetros. Por vários parâmetros, os atributos dos pacotes e conexões criados são determinados por acaso, ou podem ser controlados, como taxa de envio ou tamanho da carga útil. Mesmo gerando tráfego TCP, o BoNeSi falsifica os endereços IP de origem. Por isso, inclui uma simples *tcp-stack* para lidar com conexões TCP em modo promíscuo. Importante garantir que os pacotes de resposta sejam roteados para o host no qual o BoNeSi está sendo executado para o trabalho correto. Portanto, não é possível usar o BoNeSi em infraestruturas de rede arbitrárias. Os tráfegos mais avançados que podem ser gerado são solicitações TCP / HTTP. Várias coisas são determinadas por acaso, para tornar os pedidos de HTTP mais realistas: porta de origem ttl: 3..255 opções de tcp: de sete diferentes opções da vida real com diferentes comprimentos e probabilidades *user agent for http header*: fora de uma lista dada por arquivo.

4.7.2 T-50

O T-50 foi projetado para realizar testes numa variedade de dispositivos de rede de infraestrutura e soluções de segurança instaladas (PISSARRA, 2020). Mas neste trabalho, o T-50 foi usado para gerar tráfegos que consideramos legítimo, por motivos de menor quantidade de pacotes que gera em comparação ao BoNeSi. Os protocolos implementados ou suportados no T50 são inúmeros, mas neste trabalho considerar-se apenas:

a) ICMP - Protocolo de mensagens de controle da Internet;
b) TCP - Protocolo de Controle de Transmissão;
c) UDP - Protocolo de Datagrama do Usuário.

4.7.2.1 Características

É uma ferramenta poderosa para injetar pacotes de uma quantidade bastante grande por segundo:

a) Mais de 1.000.000 pps SYN Flood (+50% de uplink de rede) em uma rede 1000BASE-T (Gigabit Ethernet);
b) Mais de 120.000 pps SYN Flood (+60% de uplink de rede) em uma rede 100BASE-TX (Fast Ethernet);
c) Simula ataques DDoS/DoS, validando regras de firewall, ACLs de roteador, sistema de detecção de intrusão e políticas de sistema de prevenção de intrusão.

O principal diferencial do T50 é sua capacidade de enviar todos os protocolos, sequencialmente, usando um único *SOCKET*, além de poder ser usado para modificar rotas de rede, permitindo que profissionais de segurança de computadores realizem “testes de penetração” avançados.

4.8 Dataset

Datasets são conjuntos de dados compilados com informações a uma determinada necessidade. No caso deste trabalho, três categorias de datasets foram criados (para legítimo, malicioso e híbrido) para treinar os algoritmos. O tamanho de cada um contém mais de 20.000 amostras de dados com 104 atributos cada um. Cada amostra registra um fluxo de uma máquina dentro da rede e esse fluxo é identificado por uma chave composta por: Chave = MAC de origem + MAC de destino + ID do switch + Porta do Switch + IP de origem + IP de destino.

Esse modo de gerar a chave para identificar uma máquina e seu fluxo permite agregar diversas informações de vários fluxos que são endereçados a uma mesma máquina. Por causa dessa agregação de fluxos endereçado a uma mesma máquina alguns atributos registram a ocorrência da variação de portas e alguns dados do fluxo, deste modo não se perde informações que podem ajudar na construção do modelo. A Figura 4.7 mostra uma amostra de um dos datasets que foram criados.

Figura 4.7 – Amostra de dataset

Fonte: do próprio autor do trabalho (2023)

4.9 Regras de Encaminhamento/Bloqueio de tráfego

As regras utilizadas neste trabalho são:

a) Regras de encaminhamento Forwarding.l2-learning: o conjunto de regras faz com que o switch OpenFlow usado neste trabalho funcione em aprendizagem de camada *Layer L2* (Camada 2), mapeando o endereço MAC dos hosts para sua respectiva porta de switch; também, foi estabelecida uma tabela de fluxo para cada experimento, com o objetivo de criar roteamento L3 e encaminhamento de portas para determinadas portas físicas do switch;
b) Regras de firewall: essas regras são divididas em regras de firewall L2 e regras de firewall L3 (Camada 3);
   - Regras de firewall L2: regras de bloqueio de MAC foram criadas, determinado um host não pode acessar o outro de acordo com a regra especificada;
   - Regras de firewall em L3: foram criadas regras de bloqueio de endereços IP e portas, para negar o acesso de um determinado serviço em um host a um sub-rede.

4.10 Métricas de avaliação

Para avaliar os resultados do projeto, foram usadas métricas de desempenho como precisão e taxa de falsos positivos. Para monitorar o fluxo de rede em tempo real, foi usado Wireshark (um utilitário gráfico para visualizar pacotes) para coletar os dados da rede. Esses dados foram armazenados como um conjunto de dados para posterior classificação do aprendizado de máquina. Para analisar o desempenho do sistema, é preciso testar o desempenho do modelo de aprendizado de máquina do plano de conhecimento, já que o modelo de ML é também integrado ao sistema. No entanto, para calcular o desempenho do modelo, existem alguns métodos estatísticos já estabelecidos e amplamente utilizados para classificadores de aprendizado de máquina, chamado de exatidão, precisão *(precision)*, revocação *(recall)* e medidas F *(F-score)*. Essas métricas estatísticas de desempenho foram usadas para medir a precisão da detecção e de mitigação de nosso modelo.

De acordo com (LEES; SORENSEN; KIVLICHAN, 2020), ou (HAMDAN et al., 2021), podemos dizer que essas métricas são calculadas a partir de uma matriz de confusão. Na matriz de confusão, o verdadeiro positivo *(TP-true positive)* representa o número de registros positivos reais que estão corretamente classificados. Em contraste, os verdadeiros negativos *(TN-true negative)* são o número de registros negativos reais corretamente classificados. Além disso, os falsos positivos *(FP-false positive)* são o número de registros negativos classificados incorretamente, enquanto os falsos negativos *(FN-false negative)* são o número de registros positivos classificados incorretamente:

A precisão é uma métrica de quantificação do número de previsões positivas corretas feitas. Portanto, calcula a precisão para a classe minoritária. É calculado como a proporção de exemplos positivos previstos corretamente divididos pelo número total de exemplos positivos que foram previstos. A precisão não se limita a problemas de classificação binária. Em um problema de classificação desequilibrada com mais de duas classes, a precisão é calculada como a soma dos verdadeiros positivos em todas as classes dividida pela soma dos verdadeiros positivos e falsos positivos em todas as classes. O maior P reflete o menor número de falsos positivos.

$$P = \left\{ \frac{TP}{(TP+FP)} \right\} \tag{4.1}$$

A Acurácia *(Accuracy-Acc)* é definido como a porcentagem de instâncias da classificação correta dentro do número total de instâncias.

$$ACC = \left\{ \frac{TP+TN}{(TP+FP+TN+FN)} \right\} \tag{4.2}$$

A revocação é uma métrica de quantificação do número de previsões positivas corretas feitas a partir de todas as previsões positivas que poderiam ser feitas. Ao contrário da precisão que apenas comenta as predições positivas corretas de todas as predições positivas, a revocação fornece uma indicação de predições positivas perdidas. Dessa forma, a revocação fornece alguma noção da cobertura da classe positiva. A revocação não se limita a problemas de classificação binária. Em um problema de classificação desequilibrada com mais de duas classes, a recuperação é calculada como a soma dos verdadeiros positivos em todas as classes dividida pela soma dos verdadeiros positivos e falsos negativos em todas as classes. A revocação R é o número de verdadeiros positivos dividido pela soma dos falsos negativos e verdadeiros positivos. Um alto valor R é desejado.

$$R = \left\{ \frac{TP}{(TP+FN)} \right\} \tag{4.3}$$

A medida-F: considera tanto a acurácia quanto a revocação para calcular o desempenho da classificação. Podemos ter excelente precisão com terrível revocação ou, alternativamente, terrível precisão com excelente revocação. A medida F fornece uma maneira de expressar ambas as preocupações com uma única pontuação. Depois que a precisão e a recuperação foram calculadas para um problema de classificação binária ou multiclasse, as duas pontuações podem ser combinadas no cálculo da medida F.

Ela é calculada da seguinte forma:

$$Medida - F = \left\{ \frac{(2 * TP)}{(2TP + FP + FN)} \right\} \tag{4.4}$$

Assim definidas as métricas de avaliação, elas permitiram a obtenção dos resultados que apresentamos no próximo capítulo.

5 RESULTADOS

Nesta seção, apresentamos os resultados da classificação e da mitigação obtidos após os experimentos definidos nos três cenários aplicados a cada um dos algoritmos. Nos cenários, cada um dos experimentos foi executado continuamente. As mudanças no tipo de tráfego (ou cenário), foram equivalentes para cada experimento, tornando justa a comparação entre os resultados obtidos de cada algoritmo. Em cada experimento, um algoritmo de ML foi executado para classificar e bloquear os fluxos. Os experimentos duraram 30 minutos para cada algoritmo, na taxa de 10 min por experiência.

5.1 Cenário 1: Tráfego legítimo

Para iniciar o cenário 1, foram realizados testes de conectividade dando um "*ping*"entre todas as máquinas e o controlador, a fim também de testar as regras de firewall (da seção 4.9) utilizadas, e verificar testes de conexão com o servidor HTTP no host server. Cada vez que mudamos o algoritmo para uma nova experiência, reiniciamos o switch, o controlador, o KP e o PG, para garantir a limpeza de tabela de fluxos e evitar a sobrecarrega das memórias. De outro lado, o objetivo de reiniciar as máquinas é de evitar problemas ou limites como falta de amostras, *overfitting ou underfitting* dos algoritmos.

Neste cenário 1, os *Hosts (2, 3, 4)* de IP: 10.10.0.17; 10.10.0.18 e 10.10.0.19, serviram para enviar requisições legítimas de tráfego para o servidor Host1 de IP: 10.10.0.9. A partir de um *script* desenvolvido emulou-se o tráfego de usuários legítimos. Ao total, foram escritas nos arquivos CSV dos resultados: 104 linhas de amostras, no caso do algoritmo Árvore de Decisão e todos foram corretamente classificados; 64 linhas de amostras no caso de KNN com alguns erros de classificação; 100 linhas de amostras de tráfegos no caso de SVM com alguns erros de classificação, e 106 linhas de amostras no caso de Naive Bayes e todos foram corretamente classificados. A Tabela 5.1 mostra o resumo dos resultados após o cálculo das métricas de avaliação.

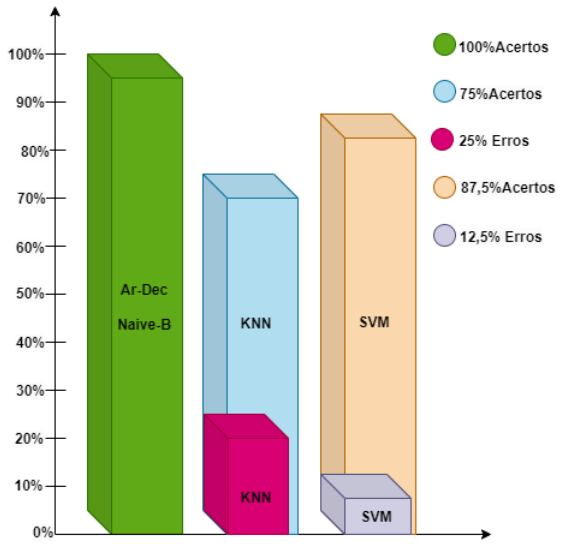
Tabela 5.1 – Resumo dos resultados do cenário 1

Algoritmos	Total Tráfego	TP	TN	FP	FN	Acurácia	Precisão	Revocação	Medida-F
Árvore de Decisão	104	0	104	0	0	100%	-	-	-
KNN	64	3	60	1	0	98,43%	75%	100%	85,71%
SVM	100	28	0	4	68	96%	87,5%	29,16%	93,33%
Naive Bayes	106	0	106	0	0	100%	-	-	-

Fonte: do próprio autor do trabalho (2023)

Analisando os resultados deste cenário 1 de tráfego legítimo, pode ser observado que os números de amostras, variam dependendo do algoritmo. Isto ocorreu devido a vários problemas como a latência no KP para classificar e escrever o resultado da classificação no arquivo CSV dos resultados; problemas de processamento ao receber pacotes; de comunicação e conexões entre os nós da rede. Neste cenário 1, observa-se que algumas métricas não puderam ser calculadas devido ao número zero de FN e TP. Os algoritmos Naive Bayes e Árvore de Decisão se saíram melhores com 100% de precisão de acertos. KNN e SVM obtiveram respectivamente 75% e 87,5% de precisão, devido ao número de TP considerável que apareceu durante a classificação enquanto todos eram tráfegos legítimos. A Figura 5.1 ilustra esse resultado.

Figura 5.1 – Gráfico de precisão de acerto do tráfego legítimo



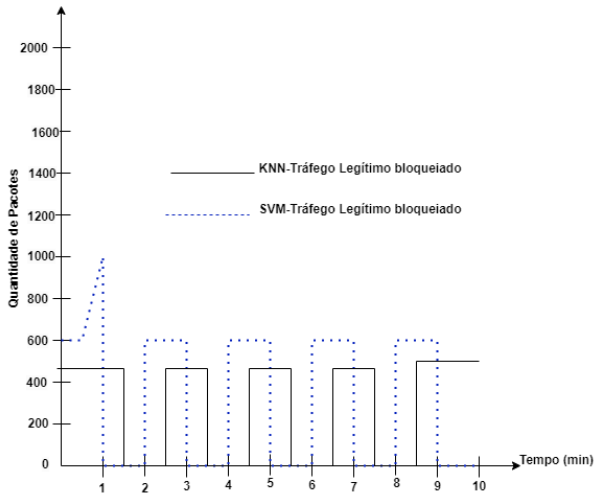
Fonte: do próprio autor do trabalho (2023)

A métrica Acurácia obtida pelos algoritmos Árvore de Decisão e Naive Bayes foi de 100% enquanto KNN e SVM deram respectivamente 98,43% e 96%. Isso é explicado pela baixa taxa de TP e FP. A métrica Revocação foi de 100% para o algoritmo KNN e 29,16% para SVM, devido aos números de TN consideráveis; enquanto a métrica Medida-F para os mesmo algoritmos, foi respectivamente de 85,71% e 93,33%. Isto é devido aos resultados da precisão e da revocação, pois ela é uma combinação dessas duas métricas.

Neste cenário 1 de tráfego legítimo, a soma de verdadeiro positivo e de falso positivo (TP+FP) foi de zero (’0’) para Árvore de Decisão e Naive Bayes devido a uma classificação correta. Consequentemente, não teve bloqueio. Mas, para KNN e SVM, a soma de verdadeiro positivo e de falso positivo (TP+FP) foram respectivamente de ’4’ e ’32’, devido aos erros de classificação. Assim, a regressão das

curvas mostrada na Figura 5.2 que ilustra o bloqueio, é explicada pelo fato do bloqueio do tráfego de acordo com as regras de decisão do controlador.

Figura 5.2 – Gráfico de Bloqueio de cenário 1



Fonte: do próprio autor do trabalho (2023)

Para o algoritmo KNN neste gráfico, a quantidade de pacotes do tráfego é iniciado em 420, entre os tempos 0 e 1min. Para o algoritmo SVM, a quantidade de pacotes do tráfego foi de 600 até 1000, entre os tempos 0 e 1min. Os tráfegos classificados como maliciosos foram bloqueados durante 1min entre os tempos 1,30min e 2,30min para o algoritmo KNN e entre os tempos 1min e 2min para o algoritmo SVM; daí a regressão das curvas. Este comportamento se manteve até o termino dos experimentos.

5.2 Cenário 2: Tráfego malicioso

No cenário 2, os *Hosts (2, 3, 4)* de IP: 10.10.0.17 usando o protocolo TCP; 10.10.0.18 usando o protocolo UDP e 10.10.0.19 usando o protocolo ICMP, enviaram requisições maliciosas de tráfego para o servidor Host1 de IP: 10.10.0.9; a partir da ferramenta BoNeSi instalado nestes Hosts (2, 3, 4). Ao total, foram escritas nos arquivos CSV dos resultados: 46 linhas de amostras de tráfegos no caso do algoritmo Árvore de Decisão com alguns erros de classificação; 51 linhas de amostras de tráfegos no caso de KNN com alguns erros de classificação; 54 linhas de amostras de tráfegos no caso de SVM com alguns erros de classificação; idem no caso de Naive Bayes com 74 linhas de amostras de tráfegos. A Tabela 5.2 mostra o resumo dos resultados após o cálculo das métricas de avaliação.

Fonte: do próprio autor do trabalho (2023)

Analisando os resultados deste cenário 2 de tráfego malicioso, pode ser observado que os números de amostras, variam de novo, dependendo do algoritmo. Isto ocorreu devido aos mesmo problemas

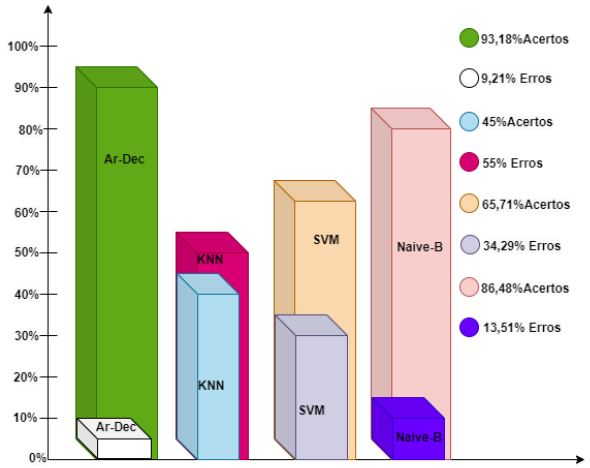
Tabela 5.2 – Resumo dos resultados do cenário 2

Algoritmos	Total Tráfego	TP	TN	FP	FN	Acurácia	Precisão	Revocação	Medida-F
Árvore de Decisão	46	41	2	3	0	100%	93,18%	100%	96,47%
KNN	51	9	31	11	0	78,43%	45%	100%	62,06%
SVM	54	23	19	12	0	77%	65,71%	100%	79,31%
Naive Bayes	74	64	0	10	0	86,48%	86,48%	100%	92,75%

de tempo de latência no KP para classificar e escrever o resultado da classificação no arquivo CSV dos resultados; problema de processamento ao receber pacotes; de comunicação e conexões entre os nós da rede.

Neste cenário 2, o algoritmo Árvore de Decisão saiu melhor com 93,18% de precisão de acertos devido aos números insignificantes de FP e TN que apareceram (esse algoritmo errou pouco). Os algoritmos SVM e Naive Bayes obtiveram respectivamente 65,71% e 86,48% de precisão, devido aos números de TP consideráveis que apareceram durante a classificação. O KNN apresentou um resultado de 45% de precisão de acertos devido a uma alta taxa de TN e FP. A Figura 5.3 ilustra esse resultado.

Figura 5.3 – Gráfico de precisão de acerto dos ataques



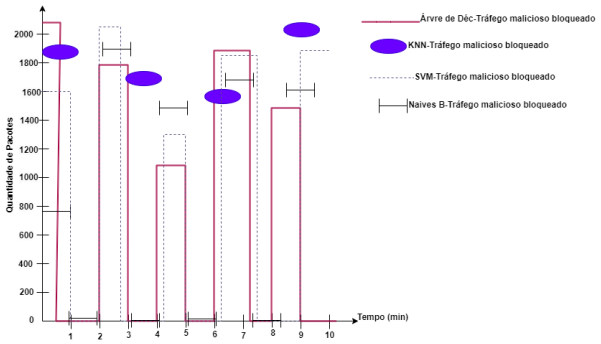
Fonte: do próprio autor do trabalho (2023)

A métrica Acurácia obtida pelo algoritmo Árvore de Decisão foi de 100% enquanto Naive Bayes, KNN e SVM deram respectivamente 86,48% 78,43% e 77%. Isso é explicado pela baixa taxa de TN e FN. A métrica Revocação foi de 100% para todos os algoritmos, devido aos números de TP consideráveis; enquanto a métrica Medida-F foi para os algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM

respectivamente de 96,47% , 92,75%, 62,06% e 79,31%. Isto é devido aos resultados da precisão e da revocação, pois ela é uma combinação dessas duas métricas.

As regras de bloqueio que permitem ao controlador de tomar decisões, têm como objetivo, bloquear ataques. Neste cenário 2 de tráfego malicioso, a soma de verdadeiro positivo e de falso positivo (TP+FP) para os algoritmos de Árvore de Decisão, Naive Bayes, KNN e SVM foram respectivamente: '44', '74', '20' e '35' apesar de alguns erros de classificação. Também, a regressão das curvas mostrada na Figura 5.4 que ilustra o bloqueio, é explicada pelo fato de bloqueio de todos os TP e FP de acordo com as regras de decisão do controlador.

Figura 5.4 – Gráfico de Bloqueio de ataques de cenário 2



Fonte: do próprio autor do trabalho (2023)

Para o algoritmo Árvore de Decisão neste gráfico, a quantidade de pacotes do tráfego é iniciado em 2200, entre os tempos 0min e 0,30min. O tráfego classificado como malicioso foi bloqueado durante 1,30min entre os tempos 0,30min e 2min; daí a regressão da curva neste intervalo de tempo. A quantidade de pacotes do tráfego variou para 1800 entre os tempos 2min e 3min; e o tráfego classificado como malicioso foi novamente bloqueado durante 1min, daí a regressão da curva neste intervalo de tempo. Este comportamento se manteve até o termino dos experimentos.

Para o algoritmo KNN, a quantidade do tráfego de pacotes é iniciado em 1900, entre os tempos 0 e 1min. O tráfego classificado como malicioso foi bloqueado durante 1,30min entre os tempos 1,30min e 3min; daí a ausência da curva neste intervalo de tempo. Entre os tempos 4min e 5min, teve de novo um bloqueio de 1min; motivo de uma nova ausência da curva neste intervalo de tempo. Este comportamento se manteve até o termino dos experimentos.

Para o algoritmo SVM, a quantidade do tráfego de pacotes é iniciado em 1600, entre os tempos 0 e 1min. Para o algoritmo Naives-B, a quantidade foi de 700 até 1900, entre os tempos 0 e 3min. Os tráfegos classificados como maliciosos foram bloqueados durante 1min entre os tempos 1min e 2min; daí, observamos a regressão ou a ausência das curvas. Estes comportamentos foram mantidos até o termino dos experimentos.

5.3 Cenário 3: Tráfego híbrido

No cenário 3 de tráfego híbrido, os *Hosts (2, 3, 4)* usaram os protocolos TCP para o IP: 10.10.0.17; UDP para o IP: 10.10.0.18 e ICMP para o IP: 10.10.0.19, para enviar requisições de tráfego malicioso, e dois *Hosts* de IP: 10.10.0.10 e 10.10.0.25 enviaram requisições legítimas para o mesmo servidor Host1 de IP: 10.10.0.9. Ao total, foram escritas nos arquivos CSV dos resultados: 70 linhas de amostras de tráfegos no caso do algoritmo Árvore de Decisão com alguns erros de classificação; 94 linhas de amostras de tráfegos no caso de KNN com alguns erros de classificação; 88 linhas de amostras de tráfegos no caso de SVM com alguns erros de classificação; idem no caso de Naive Bayes com 81 linhas de amostras de tráfegos. A Tabela 5.3 mostra o resumo dos resultados após o cálculo das métricas de avaliação.

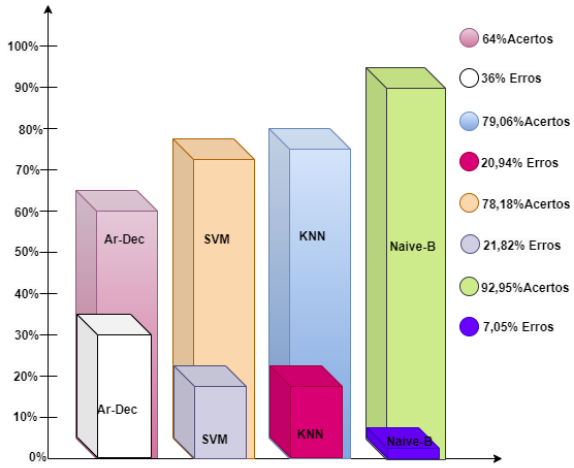
Tabela 5.3 – Resumo dos resultados do cenário 3

Algoritmos	Total Tráfego	TP	TN	FP	FN	Acurácia	Precisão	Revocação	Medida-F
Árvore de Decisão	70	32	20	18	0	74,28%	64%	100%	78%
KNN	94	34	51	9	0	90,42%	79,06%	100%	88,31%
SVM	88	43	33	12	0	86,36%	78,18%	100%	87,75%
Naive Bayes	81	66	10	5	0	93,82%	92,95%	100%	96,35%

Fonte: do próprio autor do trabalho (2023)

Analisando os resultados deste cenário 3 de tráfego híbrido, pode ser observado igualmente aos cenários precedentes, que os números de amostras, variam dependendo do algoritmo. Neste cenário 3, o algoritmo Naive Bayes saiu melhor com $92,95\%$ de precisão de acertos. Em seguida, KNN, SVM, e Árvore de Decisão obtiveram respectivamente $79,06\%$, $78,18\%$ e 64% de precisão, devido aos números de TN e FP consideráveis que apareceram durante a classificação. A Figura 5.5 ilustra esse resultado.

Figura 5.5 – Gráfico de precisão de acerto dos tráfegos Híbridos

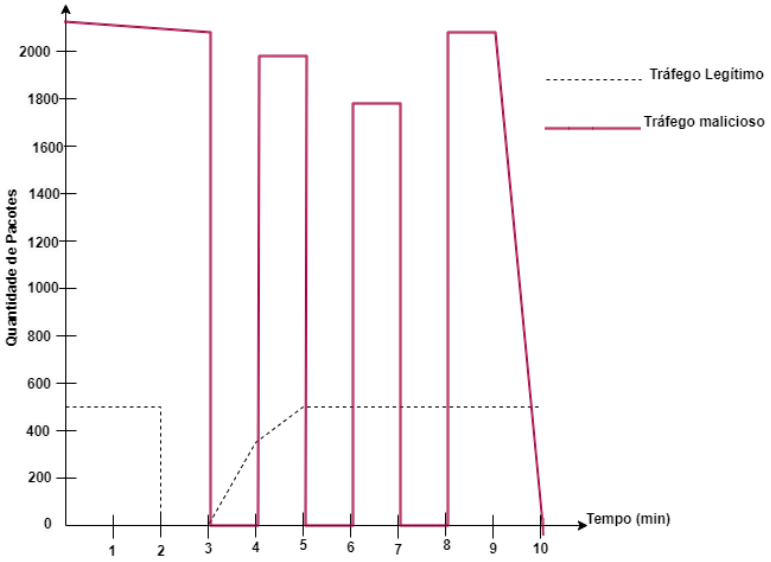


Fonte: do próprio autor do trabalho (2023)

A métrica Acurácia obtida pelos algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM foi respectivamente de $74,28\%$, $93,82\%$, $90,42\%$ e $86,36\%$. Isso é explicado pela baixa taxa de FP e du número zero de FN. A métrica Revocação foi de 100% para todos os argoritmos, devido aos números de TN e TP consideráveis; enquanto a métrica Medida-F foi para os algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM respectivamente de 78% , $96,35\%$, $88,31\%$ e $87,75\%$. Isto é devido aos resultados da precisão e da revocação, pois ela é uma combinação dessas duas métricas.

Neste cenário 3 de tráfego híbrido, a soma de verdadeiro positivo e de falso positivo (TP+FP) para Árvore de Decisão, Naive Bayes, KNN e SVM foram respectivamente de: ’50’, ’71’, ’43’ e ’55’. Observar-se que a quantidade de pacotes do tráfego é iniciado em 500, entre os tempos 0 e 2min para os tráfegos legítimos; e os pacotes que foram classificado como malicioso, principalmente com os algoritmos KNN e SVM, foram bloqueados entre os tempos 2min e 3min. Mas no caso do tráfego malicioso, a quantidade de pacotes do tráfego iniciado, foi superior à 2000 até o terceiro minuto. O bloqueio foi semelhante para todos os algoritmos a cada 60s; daí a regressão da curva entre 3min e 4min. Este comportamento se manteve até o termino dos experimentos com uma variação da quantidade dos pacotes dentro do tráfego. A Figura 5.6 ilustra o resultado dos bloqueios.

Figura 5.6 – Resultado de bloqueio de ataques no tráfego Híbrido



Fonte: do próprio autor do trabalho (2023)

5.4 Classificação e Bloqueio de tráfego

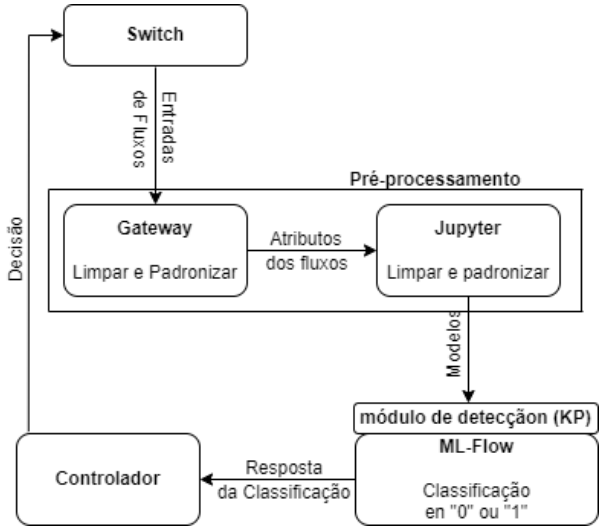
Seguindo os autores (DILLON; BERKELAAR, 2014), com base nas características dos protocolos comuns, espera-se que os atacantes tinham um comportamento padrão. Por exemplo, fluxos TCP podem retransmitir segmentos anteriores quando nenhuma confirmação é recebida e aumentar o tempo de retransmissão, resultando na supressão da taxa de tráfego. O tráfego UDP não depende de confirmações, mas é provável que seja baseado em solicitação-resposta da camada de aplicativo. Se nenhuma solicitação for enviada, nenhuma resposta recebida é esperada. O comportamento deste último é verdadeiro para serviços como DNS *(Domain Name System)* e NTP *(Network Time Protocol)*, frequentemente usados em ataques de falsificação de DDoS, mas pode não se aplicar a outras comunicações baseadas em UDP. Por outro lado, o comportamento de serviços comuns que usam UDP e ICMP deve ser mais investigado.

Quando os pacotes dos hosts entram em nosso switch, como mostrado na Figura 4.1, o módulo (3) os organiza em fluxos, com base no tempo (a cada 60s). No módulo (4) de pré-processamento do plano de gerenciamento, o Gateway limpa e padroniza os atributos dos fluxos e devolve para o Jupyter que repete o mesmo processo, e devolve os modelos para o ML-flow para classificar: isto é um pré-processamento.

O ML-flow no módulo de detecção (10), baseado nos algoritmos de ML, assume a classificação em “0” caso de tráfego legítimo, ou em “1” caso de tráfego malicioso. Quando forem identificadas fontes de tráfego malicioso, os fluxos correspondentes aos endereços IP e MAC dessas fontes serão bloqueados

conforme com as regras de decisão do controlador; obviamente, isso só funciona se o invasor DDoS não falsificar os endereços de origem. A Figura 5.7 mostra o processo.

Figura 5.7 – Pré-processamento no plano de gerenciamento



Fonte: do próprio autor do trabalho (2023)

Se a filtragem baseada na origem não for uma opção, a filtragem baseada no destino ainda pode ser feita para evitar o congestionamento da rede. Observamos que a filtragem baseada no destino, como BGP (*Border Gateway Protocol*) ou RTBH (*Remotely triggered black hole*), descarta todo o tráfego (legítimo ou malicioso) para esse destino. Para evitar esses problemas, desenvolvemos um algoritmo de bloqueio de tráfego malicioso baseado no resultado da classificação, devolvido pelo KP ao controlador. Neste algoritmo, o controlador analisa a lista de resultados de classificação enviada pelo KP. Nesta lista, qualquer Host classificado como malicioso ('1') será identificado a partir da mesma chave de identificação (IP, MAC, Porta-org/dest) dos datasets que serviram para treinar os algoritmos. A decisão de bloqueio é tomada pelo controlador e encaminhado ao plano de dados, e será armazenado dentro do switch. Assim, este Host será bloqueado automaticamente por um tempo de 60s. Esse bloqueio temporário do Host se explica pela ideia de que talvez quando ele voltar a gerar tráfego legítimo, ao invés de malicioso, passaria a não ser mais bloqueado. A Figura 5.8 mostra uma parte do algoritmo de bloqueio.

Figura 5.8 – Algoritmo de bloqueio

```
def block_host (event):
if "1" in classificados:
      msg = of.ofp_flow_mod()
      match.block_host_"1"
      msg.match = match
      msg.idle_timeout = duration[0]
      msg.hard_timeout = duration[60]
self.connection.send(msg)
else:
reconecta()
```

Fonte: do próprio autor do trabalho (2023)

Observamos então como resultado que qualquer Host que estiver dentro da rede, pode ser classificados como legítimo "0"ou malicioso "1". Assim, mesmo no caso às vezes de erros de classificação pelos algoritmos de ML, todos os Hosts classificados como "1"(soma de todos os verdadeiros positivos e falsos positivos) são bloqueados; resultando em 100% de bloqueio para todos os algoritmos.

6 CONCLUSÕES E TRABALHOS FUTUROS

Esta pesquisa ocorreu num ambiente real de laboratório e teve como objetivo, criar uma nova arquitetura da SDN. Nesta arquitetura, módulos foram implementados nos planos de conhecimento KP e gerenciamento PG para descongestionar a rede em geral e particularmente para reduzir a sobrecarga do controlador. Os atributos dos datasets criados de acordo com os cenários para treinar os algoritmos, foram: o tempo, os MAC(origem e destino), os IP (origem e destino), portas de switch e outros são exemplos. Estes algoritmos têm sido usados para classificar o tráfego como legítimo ou malicioso.

Todos os testes se comportaram como esperado, obedecendo às regras de encaminhamento, e de bloqueio dos fluxos (Forwarding.l2-learning, firewall L2, firewall L3). Como precisão de resultado da classificação durante os experimentos, tivemos no cenário 1 de tráfego legítimo, um acerto de 100% para os algoritmos de Árvore de Decisão e Naive Bayes que foram os melhores. Os algoritmos KNN e SVM tiveram respectivamente 75% e $87,5\%$ de acertos. A métrica Acurácia obtida pelos algoritmos Árvore de Decisão e Naive Bayes foi de 100% enquanto KNN e SVM deram respectivamente $98,43\%$ e 96%. A métrica Revocação foi de 100% para o algoritmo KNN e $29,16\%$ para SVM, enquanto a métrica Medida-F foi para os mesmo algoritmos, respectivamente de $85,71\%$ e $93,33\%$.

No cenário 2 de tráfego malicioso, os algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM tiveram, respectivamente $93,18\%$, $86,48\%$, 45% e $65,71\%$ de precisão de acertos. A métrica Acurácia obtida pelo algoritmo Árvore de Decisão foi de 100% enquanto Naive Bayes, KNN e SVM deram respectivamente $86,48\%$ $78,43\%$ e 77%. A métrica Revocação deu 100% para todos os argoritmos, enquanto a métrica Medida-F deu para os algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM respectivamente $96,47\%$, $92,75\%$, $62,06\%$ e $79,31\%$.

No cenário 3 de tráfego híbrido, o algoritmo Naive-Bayes foi o melhor porque teve $92,95\%$ de precisão de acertos. Os algoritmos SVM, KNN e Árvore de Decisão tiveram respectivamente $78,18\%$: $79,06\%$ e 64% de precisão de acertos. A métrica Acurácia obtida pelos algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM foi, respectivamente $74,28\%$, $93,82\%$, $90,42\%$ e $86,36\%$. A métrica Revocação deu 100% para todos os argoritmos, enquanto a métrica Medida-F deu para os algoritmos Árvore de Decisão, Naive Bayes, KNN e SVM respectivamente 78% , $96,35\%$, $88,31\%$ e $87,75\%$.

Ao longo dos experimentos, todos os Hosts que foram classificados como maliciosos, foram automaticamente bloqueados com sucesso durante um tempo de 60s, então um resultado de 100% de bloqueio.

Por um lado, embora nossa pesquisa ocorra em um ambiente de laboratório real, algumas ferramentas como BoNeSi, T50 e outras usadas ao longo desta pesquisa são baseadas principalmente em simulações, que podem não ser adequadas para cenários do mundo real. Os testes desses mecanismos

devem ser realizados para confirmar os limites definidos durante esta pesquisa, ou encontrar limites mais apropriados, com dados reais ao vivo. Esta pesquisa se concentrou na ideia de proteção contra ataques DDoS de fora da rede local. Acreditamos que os mecanismos explorados podem ser usados para detectar ataques vindos da rede local e participando de um ataque DDoS. Assim, como trabalhos futuros, outros experimentos podem ser realizados, usando hosts zumbis infectados ou serviços explorados em ataques de amplificação.

Por outro lado, a técnica de bloqueios temporários que propomos nesta pesquisa baseia-se essencialmente no resultado da classificação. Portanto, em caso de alta taxa de erros de classificação, os hosts legítimos podem ser bloqueados injustamente. Assim, para os trabalhos futuros, é importante selecionar os atributos mais relevantes dos datasets, para melhor treinar os algoritmos de ML, a fim de melhorar os resultados da classificação para evitar o bloqueio injusto de hosts legítimos.

A prevenção de ataques DDoS será assunto da próxima pesquisa, porque este trabalho baseou-se essencialmente na detecção e mitigação destes ataques.

REFERÊNCIAS

AMIRI, E.; ALIZADEH, E.; REZVANI, M. H. Controller selection in software defined networks using best-worst multi-criteria decision-making. **Bulletin of Electrical Engineering and Informatics**, v. 9, n. 4, p. 1506–1517, 2020.

ANDERSON, C. J. et al. Netkat: Semantic foundations for networks. **Acm sigplan notices**, ACM New York, NY, USA, v. 49, n. 1, p. 113–126, 2014.

BARNETT, T. et al. Cisco visual networking index (vni) complete forecast update, 2017–2022. **Americas/EMEAR Cisco Knowledge Network (CKN) Presentation**, 2018.

BARROS, B. M. Laboratório de arquitetura e redes de computadores. Abril de 2022.

BAWANY, N. Z.; SHAMSI, J. A.; SALAH, K. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. **Arabian Journal for Science and Engineering**, Springer, v. 42, n. 2, p. 425–441, 2017.

BEBIS, G. et al. Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics): Preface. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, Springer Verlag, v. 8033, n. PART 1, p. v, 2013.

BHUSHAN, K.; GUPTA, B. B. Distributed denial of service (ddos) attack mitigation in software defined network (sdn)-based cloud computing environment. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 10, n. 5, p. 1985–1997, 2019.

BONFIM, M. S.; DIAS, K. L.; FERNANDES, S. F. Integrated nfv/sdn architectures: A systematic literature review. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 6, p. 1–39, 2019.

BREIMAN, L. et al. **Classification and regression trees**. [S.l.]: Routledge, 2017.

CHICA, J. C. C.; IMBACHI, J. C.; VEGA, J. F. B. Security in sdn: A comprehensive survey. **Journal of Network and Computer Applications**, Elsevier, v. 159, p. 102595, 2020.

CLARK, D. et al. Fara: Reorganizing the addressing architecture. In: **Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture**. [S.l.: s.n.], 2003. p. 313–321.

COMPUTACAO, D.; GOMES, V. S. L. S. Isolamento de recursos em redes definidas por software virtualizadas baseadas em openflow. 2013.

COSTA, L. C. et al. Avaliaç ao de desempenho de planos de dados openflow. 2016.

COSTIN, A. et al. The role of phone numbers in understanding cyber-crime schemes. In: IEEE. **2013 Eleventh Annual Conference on Privacy, Security and Trust**. [S.l.], 2013. p. 213–220.

DABBAGH, M. et al. Software-defined networking security: pros and cons. **IEEE Communications Magazine**, IEEE, v. 53, n. 6, p. 73–79, 2015.

DEEPA, V.; RADHA, N. A survey on network intrusion system attacks classification using machine learning techniques. In: IOP PUBLISHING. **IOP Conference Series: Materials Science and Engineering**. [S.l.], 2021. v. 1022, n. 1, p. 012036.

DILLON, C.; BERKELAAR, M. Openflow (d) dos mitigation. **Technical Report (Feb. 2014)**, 2014.

ELSAYED, M. S. et al. Ddosnet: A deep-learning model for detecting network attacks. In: IEEE. **2020 IEEE 21st International Symposium on"A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)**. [S.l.], 2020. p. 391–396.

FOULADI, R. F.; ERMIŞ, O.; ANARIM, E. A ddos attack detection and countermeasure scheme based on dwt and auto-encoder neural network for sdn. **Computer Networks**, Elsevier, v. 214, p. 109140, 2022.

FOUNDATION, N. K. Kdoqi clinical practice guideline for diabetes and ckd: 2012 update. **American Journal of Kidney Diseases**, Elsevier, v. 60, n. 5, p. 850–886, 2012.

FRANKS, D. W. et al. A foundation for developing a methodology for social network sampling. **Behavioral Ecology and Sociobiology**, Springer, v. 63, n. 7, p. 1079–1088, 2009.

GADZE, J. D. et al. An investigation into the application of deep learning in the detection and mitigation of ddos attack on sdn controllers. **Technologies**, Multidisciplinary Digital Publishing Institute, v. 9, n. 1, p. 14, 2021.

GHOSH, S. et al. A cognitive routing framework for reliable communication in iot for industry 5.0. **IEEE Transactions on Industrial Informatics**, IEEE, 2022.

GUDE, N. et al. Nox: towards an operating system for networks. **ACM SIGCOMM computer communication review**, ACM New York, NY, USA, v. 38, n. 3, p. 105–110, 2008.

HAMDAN, M. et al. A comprehensive survey of load balancing techniques in software-defined network. **Journal of Network and Computer Applications**, Elsevier, v. 174, p. 102856, 2021.

HARTIGAN, J. A. Statistical theory in clustering. **Journal of classification**, Springer, v. 2, n. 1, p. 63–76, 1985.

HELLER, B.; SHERWOOD, R.; MCKEOWN, N. The controller placement problem. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 42, n. 4, p. 473–478, 2012.

ISLAM, M. J. et al. Sdot-nfv: a distributed sdn based security system with iot for smart city environments. **GUB Journal of Science and Engineering**, p. 27–35, 2020.

KOPONEN, T. et al. Onix: A distributed control platform for large-scale production networks. In: **9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)**. [S.l.: s.n.], 2010.

LAI, V.; TAN, C. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In: **Proceedings of the conference on fairness, accountability, and transparency**. [S.l.: s.n.], 2019. p. 29–38.

LARA, A.; KOLASANI, A. B. ramamurthy openflow: A survey,". **IEEE Communications**, n. 1, p. 493–512, 2014.

LARA, A.; RAMAMURTHY, B. Opensec: A framework for implementing security policies using openflow. In: IEEE. **2014 IEEE Global Communications Conference**. [S.l.], 2014. p. 781–786.

LAU, F. et al. Distributed denial of service attacks. In: IEEE. **Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0**. [S.l.], 2000. v. 3, p. 2275–2280.

LEES, A.; SORENSEN, J.; KIVLICHAN, I. Jigsaw@ ami and haspeede2: Fine-tuning a pre-trained comment-domain bert model. In: **EVALITA**. [S.l.: s.n.], 2020.

LI, D. et al. Using svm to detect ddos attack in sdn network. In: IOP PUBLISHING. **IOP Conference Series: Materials Science and Engineering**. [S.l.], 2018. v. 466, n. 1, p. 012003.

LIU, X. et al. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. **arXiv preprint arXiv:1904.09482**, 2019.

LIU, Y. et al. A novel load balancing and low response delay framework for edge-cloud network based on sdn. **IEEE Internet of Things Journal**, IEEE, v. 7, n. 7, p. 5922–5933, 2019.

LÓPEZ-RAVENTÓS, Á. et al. Combining software defined networks and machine learning to enable self organizing wlans. In: IEEE. **2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)**. [S.l.], 2019. p. 1–8.

MATTOS, D. M.; DUARTE, O. C. M.; PUJOLLE, G. A resilient distributed controller for software defined networking. In: IEEE. **2016 IEEE International Conference on Communications (ICC)**. [S.l.], 2016. p. 1–6.

MAZUR, N. I. et al. The respiratory syncytial virus vaccine landscape: lessons from the graveyard and promising candidates. **The Lancet Infectious Diseases**, Elsevier, v. 18, n. 10, p. e295–e311, 2018.

MCKEOWN, N. et al. Openflow: enabling innovation in campus networks. **ACM SIGCOMM computer communication review**, ACM New York, NY, USA, v. 38, n. 2, p. 69–74, 2008.

MEENA, R. C.; BUNDELE, M.; NAWAL, M. Ryu sdn controller testbed for performance testing of source address validation techniques. In: IEEE. **2020 3rd international conference on emerging technologies in computer engineering: Machine learning and internet of things (ICETCE)**. [S.l.], 2020. p. 1–6.

MESTRES, A. et al. Knowledge-defined networking. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 47, n. 3, p. 2–10, 2017.

NAGPAL, R.; YADAV, H. Bacterial translocation from the gut to the distant organs: an overview. **Annals of Nutrition and Metabolism**, Karger Publishers, v. 71, n. Suppl. 1, p. 11–16, 2017.

NANDA, S. et al. Predicting network attack patterns in sdn using machine learning approach. In: IEEE. **2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)**. [S.l.], 2016. p. 167–172.

NETO, J. A. M. **Metodologia científica na era da informática**. [S.l.]: Saraiva Educação SA, 2017.

PENG, T.-Q. et al. Developing an excitation-emission matrix fluorescence spectroscopy method coupled with multi-way classification algorithms for the identification of the adulteration of shanxi aged vinegars. **Food Analytical Methods**, Springer, v. 12, n. 10, p. 2306–2313, 2019.

PENTIKOUSIS, K.; WANG, Y.; HU, W. Mobileflow: Toward software-defined mobile networks. **IEEE Communications magazine**, IEEE, v. 51, n. 7, p. 44–53, 2013.

PINTEREST, E. A. The pox network software platform. 2020.

PISSARRA, J. W. . F. L. mixed packet injector tool. 2020.

PRAKASH, A.; PRIYADARSHINI, R. An intelligent software defined network controller for preventing distributed denial of service attack. In: IEEE. **2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)**. [S.l.], 2018. p. 585–589.

RAMOS, F. M.; KREUTZ, D.; VERISSIMO, P. Software-defined networks: On the road to the softwarization of networking. **Cutter IT journal**, 2015.

RUSSEL, J. et al. Crisprcastyper: automated identification, annotation, and classification of crispr-cas loci. **The CRISPR journal**, Mary Ann Liebert, Inc., publishers 140 Huguenot Street, 3rd Floor New . . . , v. 3, n. 6, p. 462–469, 2020.

SAHOO, K. S.; TIWARY, M.; SAHOO, B. Detection of high rate ddos attack from flash events using information metrics in software defined networks. In: IEEE. **2018 10th International Conference on Communication Systems & Networks (COMSNETS)**. [S.l.], 2018. p. 421–424.

SANJAA, B.; CHULUUN, E. Malware detection using linear svm. In: IEEE. **Ifost**. [S.l.], 2013. v. 2, p. 136–138.

SCARANTI, G. F. et al. Unsupervised online anomaly detection in software defined network environments. **Expert Systems with Applications**, Elsevier, v. 191, p. 116225, 2022.

SEN, S. et al. Leveraging machine learning approach to setup software-defined network (sdn) controller rules during ddos attack. In: SPRINGER. **Proceedings of International Joint Conference on Computational Intelligence**. [S.l.], 2020. p. 49–60.

SHAGHAGHI, A. et al. Towards policy enforcement point as a service (peps). In: IEEE. **2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)**. [S.l.], 2016. p. 50–55.

SHANNON, C. E. Communication in the presence of noise. **Proceedings of the IRE**, IEEE, v. 37, n. 1, p. 10–21, 1949.

SHIN, S. et al. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In: **Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security**. [S.l.: s.n.], 2013. p. 413–424.

SINGH, J.; BEHAL, S. Detection and mitigation of ddos attacks in sdn: A comprehensive review, research challenges and future directions. **Computer Science Review**, Elsevier, v. 37, p. 100279, 2020.

STANCU, A. et al. Enabling sdn application development using a netconf mediator layer simulator. In: IEEE. **2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)**. [S.l.], 2017. p. 658–663.

SUH, D. et al. On performance of opendaylight clustering. In: IEEE. **2016 IEEE NetSoft Conference and Workshops (NetSoft)**. [S.l.], 2016. p. 407–410.

TANG, T. A. et al. Deep learning approach for network intrusion detection in software defined networking. In: IEEE. **2016 international conference on wireless networks and mobile communications (WINCOM)**. [S.l.], 2016. p. 258–263.

TRAJKOVSKA, I. et al. Sdn-based service function chaining mechanism and service prototype implementation in nfv scenario. **Computer Standards & Interfaces**, Elsevier, v. 54, p. 247–265, 2017.

UJJAN, R. M. A. et al. Towards sflow and adaptive polling sampling for deep learning based ddos detection in sdn. **Future Generation Computer Systems**, Elsevier, v. 111, p. 763–779, 2020.

VAUGHAN-NICHOLS, S. J. Openflow: The next generation of the network? **Computer**, IEEE Computer Society, v. 44, n. 08, p. 13–15, 2011.

WANG, Y. et al. Sgs: Safe-guard scheme for protecting control plane against ddos attacks in software-defined networking. **IEEE Access**, IEEE, v. 7, p. 34699–34710, 2019.

WOLPERT, D. H.; MACREADY, W. G. et al. **No free lunch theorems for search**. [S.l.], 1995.

XIAO, Y. et al. Discovery method for distributed denial-of-service attack behavior in sdns using a feature-pattern graph model. **Frontiers of Information Technology & Electronic Engineering**, Springer, v. 20, n. 9, p. 1195–1208, 2019.

YI, X. et al. Deep distributed fusion network for air quality prediction. In: **Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining**. [S.l.: s.n.], 2018. p. 965–973.

YIN, M.; VAUGHAN, J. W.; WALLACH, H. Understanding the effect of accuracy on trust in machine learning models. In: **Proceedings of the 2019 chi conference on human factors in computing systems**. [S.l.: s.n.], 2019. p. 1–12.

YUNGAICELA-NAULA, N. M.; VARGAS-ROSALES, C.; PEREZ-DIAZ, J. A. Sdn-based architecture for transport and application layer ddos attack detection by using machine and deep learning. **IEEE Access**, IEEE, v. 9, p. 108495–108512, 2021.

YUREKTEN, O.; DEMIRCI, M. Sdn-based cyber defense: A survey. **Future Generation Computer Systems**, Elsevier, v. 115, p. 126–149, 2021.