UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MARCOS AURÉLIO RIBEIRO

DETECTANDO E MITIGANDO ATAQUES DDOS COM A ABORDAGEM MTD COM BASE NA CLASSIFICAÇÃO DE FLUXO AUTOMATIZADA EM REDES SDN.

CURITIBA

2023

MARCOS AURÉLIO RIBEIRO

DETECTANDO E MITIGANDO ATAQUES DDOS COM A ABORDAGEM MTD COM BASE NA CLASSIFICAÇÃO DE FLUXO AUTOMATIZADA EM REDES SDN.

Detecting and Mitigating DDoS Attacks with MTD approach based on automated flow classification in SDN networks.

Dissertação apresentada como requisito para obtenção do título de Mestre em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Mauro Sergio Pereira Fonseca

Coorientador: Profª. Drª. Juliana de Santi

CURITIBA

2023

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

MARCOS AURELIO RIBEIRO

DETECTANDO E MITIGANDO ATAQUES DDOS COM A ABORDAGEM MTD COM BASE NA CLASSIFICAÇÃO DE FLUXO AUTOMATIZADA EM REDES SDN.

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Computação Aplicada da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Sistemas Computacionais.

Data de aprovação: 15 de Dezembro de 2022

Dr. Mauro Sergio Pereira Fonseca, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Ana Cristina Barreiras Kochem Vendramin, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Carlos Alberto Maziero, Doutorado - Universidade Federal do Paraná (Ufpr)

Dr. Daniel Fernando Pigatto, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Juliana De Santi, Doutorado - Universidade Tecnológica Federal do Paraná

Dedico este trabalho de pesquisa a minha esposa, a meus pais e a meu irmão, por todos os momentos que estive ausente.

AGRADECIMENTOS

Nesses anos de mestrado, de muito estudo, esforço e empenho, permeada por inúmeros desafios, tristeza, incerteza, alegria e diversos percalços pelo caminho. Trilhar essa caminhada só é possível com apoio, energia, bondade e compreensão de todos que acompanharam e foram fundamentais para a realização de mais uma etapa da minha vida, a quem dedico especialmente este projeto.

Primeiramente, gostaria de agradecer ao maravilhoso Deus por ter me fornecido os instrumentos necessários para minha jornada até aqui, e por toda a persistência para não desistir diante das dificuldades encontradas nesta caminhada.

À minha amada esposa, Márcia, pelo seu cuidado e dedicação, que nos momentos mais difíceis trouxe alento ao cansaço e solidão. Você é o presente de Deus para minha vida.

Aos meus pais, Nair e Antonio, amo-os incondicionalmente, presente em todos os momentos da minha vida, por me ensinarem a lutar e nunca desistir dos meus objetivos.

Ao meu irmão, Márcio, pelo seu amor e carinho, e pelo prazer de compartilhar pensamentos e histórias. Você é uma parte de mim que Deus enviou para completar minha existência.

Agradeço aos meus orientadores, Profº. Drº. Mauro Sergio Pereira Fonseca e Profª. Drª. Juliana de Santi que me orientaram durante toda essa jornada e ajudaram em meu crescimento pessoal e profissional, ao qual tenho uma eterna dívida de gratidão.

Um obrigado especial aos amigos, que sempre estiveram ao meu lado, me apoiando e torcendo, independente se estavam distantes ou próximos.

Enfim, a todos aos que amo, que de alguma forma contribuíram para essa conquista.

A todos, meu sincero agradecimento.

A melhor maneira que o homem dispõe para se
aperfeiçoar é aproximar-se de Deus.
(Pitágoras)

RESUMO

A Negação de Serviço Distribuída (*Distributed Denial of Service* (DDoS)) coordena ataques sincronizados a sistemas na Internet usando um conjunto de *hosts* infectados (*bots*). Os *bots* são programados para atacar um determinado alvo disparando diversas requisições sincronizadas, causando lentidão ou indisponibilidade do serviço. Esse tipo de ataque cresceu recentemente em magnitude, diversidade e custo econômico. Assim, este estudo tem como objetivo apresentar uma arquitetura de detecção e mitigação de DDoS em Redes Definidas por Software (*Software Defined Networking* (SDN)). Neste trabalho, considera-se a abordagem *Moving Target Defense* (MTD), redirecionando inundações maliciosas para servidores descartáveis de baixa capacidade para proteger o servidor principal enquanto desencoraja o invasor. A decisão de redirecionamento é baseada em um sensor, que emprega algoritmos de Aprendizado de Máquina (*Machine Learning* (ML)) para classificação de fluxo. Quando os fluxos maliciosos são detectados, o sensor notifica o controlador SDN para incluí-los nas listas de *hosts* maliciosos e realizar o redirecionamento. A validação e avaliação da arquitetura proposta são realizadas por simulação. Resultados considerando diferentes modelos de classificação (probabilístico, linear, redes neurais e árvores) e tipos de ataque indicam que a arquitetura proposta é eficiente em detectar e mitigar ataques DDoS em aproximadamente 3,00 segundos.

Palavras-chave: negação de serviço distribuído; moving target defense; aprendizado de máquina; redes definidas por software.

ABSTRACT

The Distributed Denial of Service (DDoS) coordinates synchronized attacks on systems on the Internet using a set of infected hosts (bots). Bots are programmed to attack a determined target by firing a lot of synchronized requests, causing slowness or unavailability of the service. This type of attack has recently grown in magnitude, diversity, and economic cost. Thus, this study aims to present a DDoS detection and mitigation architecture on Software Defined Networking (SDN). It considers the Moving Target Defense (MTD) approach, redirecting malicious floods for expendable low-capacity servers to protect the main server while discouraging the attacker. The redirecting decision is based on a sensor, that employs Machine Learning (ML) algorithms for flow classification. When malicious flows are detected, the sensor notifies the SDN controller to include them in the malicious lists and to realize the redirection. The validation and evaluation of the proposed architecture are conducted by simulation. Results considering different classification models (probabilistic, linear model, neural networks, and trees) and attack types indicate that the proposed architecture is efficient in detecting and mitigating DDoS attacks in approximately 3.00 seconds.

Keywords: distributed denial of service; moving target defense; machine learning; software defined networking.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	Application Programming Interface
CSV	*Comma-separated values*
DDoS	*Distributed Denial of Service*
DoS	*Denial of Service*
DRL	*Deep Reinforcement Learning*
GNB	*Gaussian Naive Bayes*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
MAC	*Media Access Control*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
MTD	*Moving Target Defense*
NAT	*Network Address Translation*
NBI	*Northbound Interface*
OF	*OpenFlow*
ONF	*Open Networking Foundation*
OVS	*OpenvSwitch*
RF	*Random Forest*
RTT	Round-trip time
SBI	*Southbound Interface*
SDN	*Software Defined Networking*
SNMP	*Simple Network Management Protocol*

SVM	*Support Vector Machine*
TCP	*Transmission Control Protocol*
vIP	IP Virtuais
WSGI	*Web Server Gateway Interface*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos estamos vivenciando uma mudança de paradigma na prestação de serviços de tecnologia. Com crescimento exponencial, a adoção de serviços on-line impactou diretamente na evolução das empresas, as quais migraram de parques tecnológicos próprios para computação em nuvem (SINGH; BEHAL, 2020; BHARDWAJ *et al.*, 2021). Aproximadamente 90% das organizações estão migrando seus recursos de trabalho para ambientes virtualizados em nuvem (RAJAKUMARAN; VENKATARAMAN, 2021). A Gartner, Inc (2022), prevê crescimento de gastos mundiais para usuários finais em nuvens de 20,4% em 2022, e crescimento com infraestrutura como serviço em 30,6%. Deste modo, as empresas modernas têm como pilar os serviços on-line e, consequentemente, seus negócios são projetados sobre essas plataformas. Na outra extremidade estão os provedores de serviços, que diante da necessidade de diferenciação de serviços forçaram as empresas a elevarem seus níveis de disponibilidade e segurança, uma vez que sobre seus ombros estão os negócios de seus clientes (SINGH; BHANDARI, 2020).

Junto com esta mudança de paradigma, os ataques cibernéticos nas plataformas on-line também evoluíram, tornando-se mais efetivos/destrutivos, impactando maior número de serviços/usuários e ganhando maior visibilidade. Neste contexto, um dos ataques de maior incidência nos últimos anos é o ataque de Negação de Serviço Distribuído (*Distributed Denial of Service* (DDoS)) (CIL; YILDIZ; BULDU, 2021).

O ataque DDoS é um dos ataques mais destrutivos da Internet moderna devido à sua forma de operação (LIU *et al.*, 2018). Para executar o ataque, o invasor usa várias máquinas infectadas para enviar tráfego indevido ao servidor da vítima, esgotar seus recursos e torná-lo indisponível para seus clientes (LIU *et al.*, 2018; ZHOU *et al.*, 2022). O *Modus operandi* do ataque causa enormes despesas financeiras às empresas e instituições. Além dos efeitos diretos da interrupção do serviço, existem efeitos colaterais como reputação comercial, redundância de equipamentos, custos de dimensionamento e aumento do consumo de energia (CIL; YILDIZ; BULDU, 2021; BHARDWAJ *et al.*, 2021).

Além das tradicionais motivações (vandalismo, rivalidade comercial, etc.), os ataques de DDoS atuais tem sido impulsionados por disputas políticas e guerras cibernéticas. Nestes casos, o objetivo é deixar inoperante sistemas de defesa e infraestruturas de países, inviabilizando serviços básicos e essenciais com impacto direto para sua população (BHARDWAJ *et al.*, 2021).

Vários estudos presentes na literatura têm o objetivo de detectar e bloquear o atacante ou impedi-lo de acessar determinado recurso (MOHANAPRIYA; SHALINIE, 2017; MOHAMMED *et al.*, 2018). Entretanto, o bloqueio do atacante ou suas atividades, muitas vezes, não se torna uma ação de defesa efetiva, uma vez que o ataque é desenvolvido a partir de inúmeros *hosts* distribuídos na Internet.

Considerando o contexto exposto, este trabalho propõe uma arquitetura modular e flexível para detecção e mitigação do DDoS em tempo real nas Redes Definidas por Software (*Software Defined Networking* (SDN)). A arquitetura proposta tem como pilares o *Moving Target Defense* (MTD), o Aprendizado de Máquina (*Machine Learning* (ML)) e a flexibilidade provida pelo paradigma SDN. Levando-se em consideração a ideia do *Moving Target Defense*, que provê implementações, configurações de redes e caminhos alternativos com o intuito de aumentar a imprevisibilidade para o atacante, na estrutura proposta, um controlador SDN é responsável pelo balanceamento de fluxos entre um servidor principal e um servidor secundário. Com esta estratégia, permite-se que o atacante continue o ataque, mas direcionando-o para um servidor secundário controlado e com poucos recursos. O atacante ao identificar a indisponibilidade ou lentidão de resposta, tem a percepção de que a investida contra a vítima foi bem sucedida, enquanto os recursos do servidor principal são utilizados para prover serviços para usuários legítimos. Desta forma, a segurança é melhorada na medida em que se aumenta a resiliência e a disponibilidade dos serviços (YUNGAICELA-NAULA *et al.*, 2022a).

O controlador SDN realiza o encaminhamento de fluxos para o servidor primário ou secundário a partir da informação sobre a detecção de ataque gerada pelo sensor proposto na arquitetura. O sensor usa algoritmos de Aprendizado de Máquina sobre fluxos coletados da rede para classificá-los em fluxos benignos ou maliciosos. Para o desenvolvimento do sensor foram testados quatro modelos de classificadores de famílias diferentes: modelos probabilísticos (*Gaussian Naive Bayes* (GNB)), lineares (*Support Vector Machine* (SVM)), árvores (*Random Forest* (RF)) e rede neurais artificiais (*Multilayer Perceptron* (MLP)). A utilização conjunta destes algoritmos (Pilha de Classificadores) também foi usada como sensor. A validação e avaliação da arquitetura proposta foi realizada através de simulação. Os resultados gerados mostram que a arquitetura é capaz de manter a transparência do redirecionamento de fluxos, que o sensor é rápido e eficiente na detecção do ataque independentemente do tipo do ataque e do modelo preditivo empregado e que o sensor é flexível quanto à escolha dos algoritmos de ML a serem usados.

1.1 OBJETIVOS GERAIS

O objetivo deste trabalho é apresentar uma arquitetura de detecção e mitigação de ataque DDoS em tempo real utilizando as técnicas *Moving Target Defense* e Aprendizado de Máquina em Redes Definidas por Software.

1.1.1 Objetivos específicos

Para atingir seu objetivo geral, este estudo também deve concluir as seguintes etapas:

- Desenvolver um controlador SDN capaz de balancear requisições *Web* com regras mutáveis a partir de sensores localizados na rede e redirecionar o fluxo do ataque para um servidor secundário.

- Desenvolver um sensor que utilize algoritmos de Aprendizado de Máquina e consiga detectar ataques DDoS a partir de fluxo de rede.

- Implementar um protótipo que permita executar os sistemas desenvolvidos em ambiente capaz de validar o mostrar o desempenho dos algoritmos de Aprendizado de Máquina e da arquitetura MTD.

1.2 PUBLICAÇÃO

Os resultados dessa dissertação foram documentados no periódico a seguir, o qual se encontra em processo de submissão:

- RIBEIRO, Marcos Aurélio; FONSECA, Mauro Sergio Pereira; SANTI, Juliana. Detecting and Mitigating DDoS Attacks with Moving Target Defense approach based on automated flow classification in SDN networks. **Expert Systems with Applications**[1]**.**

1.3 ESTRUTURA DO DOCUMENTO

Este documento está organizado da seguinte forma: a Seção 2 aborda o referencial teórico e os trabalhos relacionados. A Seção 3 descreve a arquitetura do projeto e detalha as implementações desenvolvidas. A Seção 4 apresenta os resultados da avaliação da arquitetura proposta. A presente dissertação é concluída na Seção 5.

[1] https://www.journals.elsevier.com/expert-systems-with-applications - acesso em 8/11/2022

2 REVISÃO LITERÁRIA E TRABALHOS RELACIONADOS

A área de segurança da informação evolui apoiada em diversas tecnologias e a partir de esforços de diversos pesquisadores. A seguir são apresentados os principais tópicos relacionados a este estudo, para cada um deles é apresentada sua definição e exemplos de estudos relevantes para o tema desta dissertação.

2.1 *Moving Target Defense*

Modelos/infraestruturas de segurança estáticos são ambientes propícios para ataques cibernéticos (NGUYEN *et al.*, 2022), pois os invasores têm tempo para estudar e planejar ataques nestas infraestruturas. Como consequência, as ferramentas de segurança padrão não são suficientes para protegê-los. Dentro desse contexto nasce a necessidade do *Moving Target Defense* (MTD), um paradigma que visa tornar os sistemas e redes de computadores mais seguros utilizando uma abordagem proativa e adaptativa. Ao introduzir um ambiente de segurança dinâmico e em constante evolução, o MTD aumenta a incerteza e a complexidade para o atacante. Assim, o MTD pode melhorar o desempenho do sistema, torná-lo tolerante e recuperável de falhas, e prevenir ataques cibernéticos (ALHOZAIMY; MENASCé, 2022; NGUYEN *et al.*, 2022).

O MTD pode ser implementado a partir de alterações na topologia da rede, reconfiguração de *hosts* e nos sistemas operacionais, ou alteração a nível de aplicação. Outra estratégia é usar um gatilho temporal (NGUYEN *et al.*, 2022; AZAB; SAMIR; SAMIR, 2022) ou gatilho baseado em eventos identificados como maliciosos (NGUYEN *et al.*, 2022) para modificar o ambiente em que o ataque poderá ocorrer.

A escolha do momento que será realizada a movimentação do alvo e sua periodicidade é crucial na estratégia a ser implementada. Sistemas que sofrem excessivas modificações, podem ter reflexos diretamente no desempenho e na disponibilidade do sistema. Sistemas com poucas modificações podem estar mais suscetíveis a ataques. As estratégias MTD podem ser divididas em: estratégias proativas, reativas e híbridas. A maioria das estratégias MTD utilizam a estratégia proativa, modificando regularmente as configurações utilizadas em intervalos estabelecidos ou aleatórios. Essa estratégia garante que qualquer investida do invasor em reconhecer o sistema torna-se obsoleta nas próximas intervenções do MTD (ALHOZAIMY; MENASCé, 2022). Uma das estratégias proativas comumente utilizadas são as baseadas em tempo. A partir de um gatilho temporal um processo MTD é executado modificando o ambiente em que o ataque poderá ocorrer (NGUYEN *et al.*, 2022; AZAB; SAMIR; SAMIR, 2022). Segundo Alhozaimy e Menascé (2022) e Nguyen *et al.* (2022), a defesa reativa inclui um mecanismo capaz de reconhecer a ocorrência de uma atividade maliciosa, esse mecanismo será utilizado como gatilho para a execução de determinado evento ou alerta para que um processo MTD seja executado. Na defesa híbrida, na concepção de Alhozaimy e Menascé (2022), utilizam-se as abordagens proativa e reativa. Em intervalos de tempo é executada uma operação MTD conectada a even-

tos ou alertas de segurança. Os intervalos de tempo evitam potenciais ameaças não detectadas pelos sistemas de segurança reativos, resultando em uma arquitetura mais resiliente a ataques.

Segundo Alavizadeh *et al.* (2021), Rajakumaran e Venkataraman (2021) e Alhozaimy e Menascé (2022) as técnicas de MTD se dividem em: embaralhamento, redundância e diversidade. As técnicas de embaralhamento são implementadas a partir de randomização e reorganização de configuração de sistemas. A técnica é utilizada em métodos de tolerância a falhas para garantir operações bem-sucedidas, ainda que ocorram falhas em *hardware* ou *software*. O objetivo é confundir e criar incerteza para o atacante. As técnicas de diversidade utilizam várias implementações para garantir a invulnerabilidade do sistema no momento da ocorrência do ataque. Umas das abordagens utilizadas é desenvolver versões alternativas do software que possam ser executadas em vários sistemas, não permitindo que o atacante explore uma vulnerabilidade específica de um *software*. As técnicas de redundância pretendem garantir a confiabilidade e estabilidade do sistema, através da implantação de redundâncias de dispositivos. Assim, se ocorrerem ameaças esses equipamentos paralelos serão comprometidos enquanto os serviços essenciais continuarão em perfeito funcionamento.

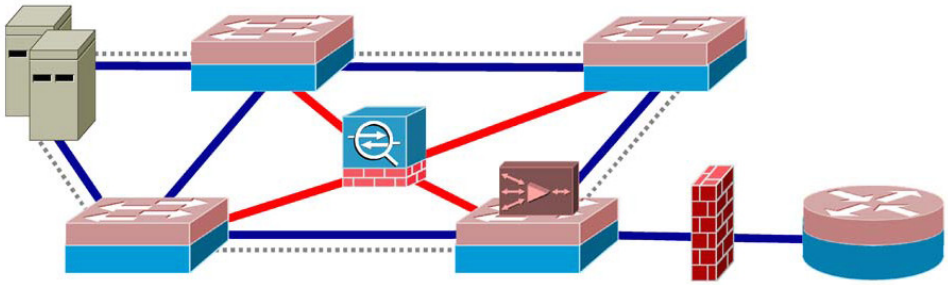
2.2 Redes Definidas por Software

Atualmente o plano de controle e dados são incorporados em dispositivos proprietários, o que traz diversas dificuldades às redes de computadores tradicionais. A adoção de protocolos proprietários e as proteções de patentes de dispositivos reduzem consideravelmente a flexibilidade necessária nas redes de computadores, resultando em custo na implantação de novos serviços e dificuldade de gerenciamento da rede. As Redes Definidas por Software nasceram com o intuito de reduzir essas dificuldades (VALDOVINOS *et al.*, 2021; MAHESHWARI *et al.*, 2022).

A arquitetura propõe o desacoplamento do plano de dados do plano de controle. Deste modo, as funções que no modelo tradicional eram implementadas nos roteadores (Figura 1), nas redes SDN são implementadas de forma centralizada no controlador SDN (Figura 2) (VALDOVINOS *et al.*, 2021; YUNGAICELA-NAULA *et al.*, 2022b). No modelo tradicional, cada roteador é responsável por realizar a função de controle e encaminhamento dos pacotes nas redes de computadores. Na arquitetura SDN, os roteadores obedecem às regras definidas pelo controlador SDN de forma centralizada. Essa abordagem diminui consideravelmente as limitações encontradas nas redes tradicionais e provê maior controle da infraestrutura da rede, uma vez que todo o roteamento e o mecanismo de controle são centralizados no controlador (YUNGAICELA-NAULA *et al.*, 2022b).

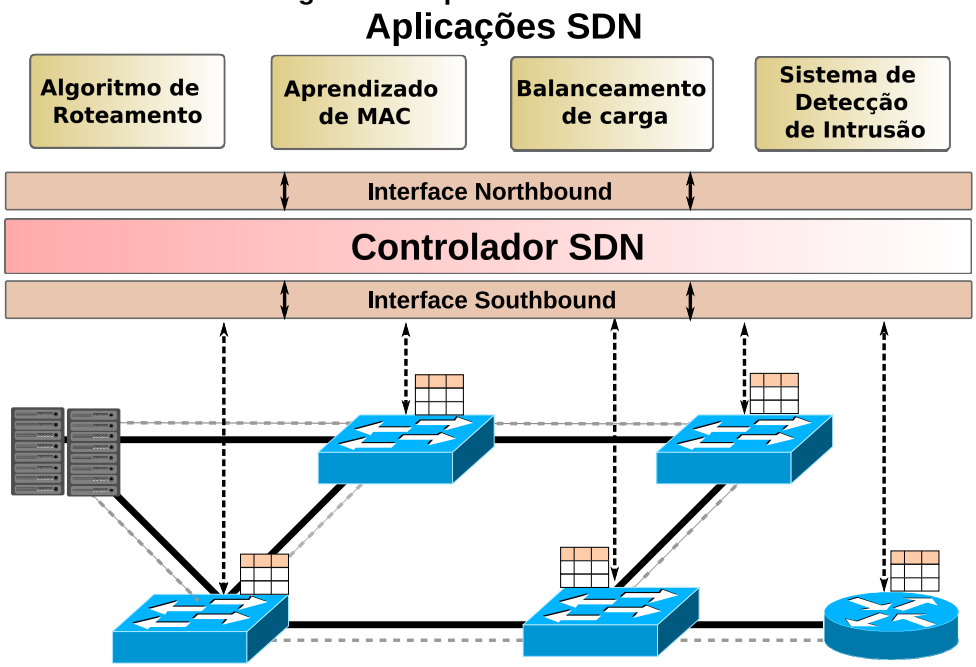
A flexibilidade oferecida nas Redes Definidas por Software permite redes programáveis de fácil gerenciamento e propiciam a capacidade de inovação, que não eram possíveis nas redes tradicionais (BOERO; MARCHESE; ZAPPATORE, 2017; ABHIROOP; BABU; MANOJ, 2018; VALDOVINOS *et al.*, 2021; YUNGAICELA-NAULA *et al.*, 2022b). A arquitetura SDN tem empol-

Figura 1 – Arquitetura de rede tradicional



Fonte: Adaptado de (KREUTZ *et al.*, 2015).

Figura 2 – Arquitetura de rede SDN



Fonte: Adaptado de (KREUTZ *et al.*, 2015).

gado os pesquisadores de segurança devido ao fato de permitir o refinamento no gerenciamento de fluxo, permitindo a programabilidade e a flexibilidade na criação de aplicações voltadas para áreas de segurança (YUREKTEN; DEMIRCI, 2021).

2.2.1 Protocolo e *Switch OpenFlow*

Para a implementação do paradigma SDN o protocolo padrão é o *OpenFlow* (OF) um protocolo aberto para comunicação entre os *switches* e o controlador SDN (TANG *et al.*, 2016). A comunicação entre o controlador SDN e os *switches* é realizada através de mensagens OF. Os *switches* encapsulam e enviam pacotes de dados para o controlador através de um pacote *OpenFlow*. Do mesmo modo, o controlador pode inserir regras de encaminhamento de fluxo nos *switches* (AL-NAJJAR *et al.*, 2017).

A estrutura lógica de um *switch* OpenFlow é constituída de uma ou mais tabelas de fluxo e uma tabela de grupo, responsáveis pelas pesquisas e encaminhamento de pacotes (NYGREN *et al.*, 2015). Na comunicação com os dispositivos, cada *switch* OF necessita manter o conjunto de tabelas de fluxos atualizadas. Essas tabelas de fluxos são instaladas pelo plano de controle, estabelecendo regras de como serão tratados os fluxos de pacotes que trafegam na rede. Essas regras especificam quais ações um *switch* executa quando um determinado fluxo adentrar o dispositivo. As características do fluxo variam de acordo com o cabeçalho do protocolo OF. Para cada fluxo de rede específico existirá uma tabela de fluxo correspondente. Deste modo, os *switches* OF usam uma regra de correspondência e ação com a tabela de fluxo. Essa tabela de fluxo equivale a uma tabela de roteamento camada 3 ou uma tabela de endereço de *MAC Address* em um *switch* Ethernet camada 2, mas a tabela de fluxo é mais dinâmica (KRONGBARAMEE; SOMCHIT, 2018).

Segundo Krongbaramee e Somchit (2018), nas primeiras versões do protocolo *OpenFlow*, a tabela de fluxo era limitada. As únicas correspondências que poderiam ser usadas da camada 4 era a porta de origem e de destino e ainda existia a limitação de ações que poderiam ser executadas com os campos. Com a evolução do protocolo, vários campos do fluxo foram expandidos na versão 1.5.0 e acima. O Controlador Ryu SDN Framework (Seção 3.2) , utilizado nesse trabalho, é um dos controladores que suportam as versões mais recentes do protocolo *OpenFlow*.

2.2.2 Interface *Northbound* e Interface *Southbound*

A arquitetura das Redes Definidas por Software possui duas interfaces de comunicação definidas: interface para o sul (*Southbound Interface* (SBI)) e a interface para o norte (*Northbound Interface* (NBI)) (ilustradas na Figura 2). A primeira foi padronizada pela *Open Networking Foundation* (ONF) [1] e permite o controle da rede e seus periféricos, como *switches* e roteadores, através da integração com o protocolo *OpenFlow*. A segunda é utilizada para conexão entre o controlador e recursos extras necessários. Esses recursos podem ser aplicativos ou serviços adicionados ao controlador como sistemas de intrusão ou balanceadores de carga. A interface favorece o aperfeiçoamento e a criação de novos recursos nas Redes Definidas por Software (NATANZI; MAJMA, 2017; SULTANA *et al.*, 2019).

2.3 Aprendizado de máquina

Os primeiros estudos na área de segurança de redes de computadores baseavam-se no bloqueio do ataque utilizando limites estatísticos ou em políticas definidas pelos administradores de sistema. Esses métodos resultam muitas vezes em conclusões equivocadas, além de

[1] https://opennetworking.org/ - acesso em 8/11/2022

exigir experiência ao realizar ajustes nas políticas que serão implementadas e que deverão ser revisadas periodicamente (LIU *et al.*, 2022). O principal desafio no diagnóstico do ataque DDoS é diferenciar o tráfego gerado pelos ataques do tráfego legítimo. Essa tarefa se torna mais complexa pela quantidade de tráfego legítimo que transpassam a rede (ZHOU *et al.*, 2022). Para que seja possível diferenciar o fluxo malicioso é necessário que as ferramentas consigam "aprender" sobre o fluxo. Esses foram os principais motivadores das pesquisas utilizando a Inteligência Artificial (DAYAL; SRIVASTAVA, 2018).

Nas visão de McCarthy (2004), Inteligência Artificial é definida como a ciência capaz de criar máquinas e computadores inteligentes, enquanto que para Seshia, Sadigh e Sastry (2022) é a expressão usada quando sistemas computacionais imitam atributos da inteligência humana.

O Aprendizado de Máquina (Machine Learning (ML)) é uma ramificação da Inteligência Artificial em que um computador a partir da observação de um conjunto de dados, cria um modelo com base nos dados de entrada com o objetivo de resolver determinado problema (BADUGE *et al.*, 2022). O Aprendizado de Máquina é um método de análise de dados que automatiza a construção de modelos analíticos, usando algoritmos que possuem a capacidade de aprender interativamente. O ML permite que computadores encontrem relacionamentos implícitos sem serem programados diretamente para procurar (CELESOVA *et al.*, 2019). Assim sendo, para permitir ao algoritmo que ele aprenda é necessário a ajuda de conjunto de dados precisos, para gerar a classificação de tráfego normal em relação ao tráfego gerado pelo ataque. Deste modo, para a implementação de algoritmos, é fundamental determinar as características que serão usadas para treinar o algoritmo a partir dos relacionamentos implícitos. Essas características são basicamente os padrões de tráfego que ajudam o algoritmo a aprender o comportamento da rede de computadores. Somente uma característica não é suficiente para determinar um padrão na rede, é necessário um conjunto apropriado de recursos reunidos para determinar um padrão de ataque com eficiência (DAYAL; SRIVASTAVA, 2018).

A partir do exposto, percebe-se que a chave para o sucesso dos algoritmos de Aprendizado de Máquina é como eles são treinados. Quanto mais precisos forem os dados, melhor será o desempenho do algoritmo. Os tipos de algoritmos de ML diferem em suas abordagens, de acordo com o tipo de dados que eles inserem e produzem, e o tipo de tarefa ou problema que eles pretendem resolver (LIU; XU, 2019). Segundo Sahoo *et al.* (2018) e Celesova *et al.* (2019), existem 3 tipos de técnicas:

- Algoritmo de aprendizado supervisionado (LIU; XU, 2019): nesse modelo, o objetivo do algoritmo é aprender através de uma função que mapeia uma entrada para uma saída utilizando como base exemplos de pares de entrada-saída. A função aprende a partir de dados de treinamento rotulados que compõem um conjunto de exemplos de treinamento. Nesse caso, cada exemplo é um par que se constitui em um objeto de entrada e um valor de saída desejado. A proposta do algoritmo supervisionado é analisar os dados de treinamento e produzir uma função deduzida, que será usada para

mapear novos exemplos. Em resumo, a máquina prevê a classe de dados de entrada com base na amostra de treinamento. Isso é chamado de supervisionado porque a classe de amostra de treinamento é conhecida na fase de aprendizado da máquina e a saída do algoritmo são as classes treinadas.

- Algoritmo de aprendizado não-supervisionado (LIU; XU, 2019): nesse caso, o aprendizado não utiliza dados rotulados, classificados ou categorizados. O algoritmo identifica pontos em comum e age com base na presença ou falta desses pontos em cada novo dado. O modelo trabalha com base em estimativa de densidade estatística. Deste modo, o modelo deve aprender os relacionamentos entre os elementos em um conjunto de dados e classificá-los sem auxílio, diferentemente do algoritmo supervisionado que necessita rotular os dados previamente. Esse algoritmo tem o objetivo de buscar estruturas, padrões ou relacionamentos indiretos para analisar novos dados.

- Algoritmo de aprendizado semi-supervisionado (LIU; XU, 2019): esse modelo utiliza uma quantidade de dados rotulados e não rotulados para treinamento. Normalmente é utilizado uma pequena quantidade de dados rotulados e uma grande quantidade de não rotulados. Esse tipo de aprendizagem é um modelo intermediário entre o aprendizado supervisionado e o não-supervisionado. Algumas pesquisas mostraram que a utilização em conjunto dos dois modelos produz melhores resultados de aprendizagem do algoritmo, sem o tempo e o custo que a aprendizagem supervisionada despendia. Geralmente, o modelo primeiro analisa os dados rotulados e o resultado é usado para extrair dados não rotulados.

Nesta dissertação são testados cinco algoritmos de Aprendizado de Máquina com relevância em modelos: probabilísticos, lineares, árvores e redes neurais artificiais.

***Gaussian Naive Bayes* (GNB):** o algoritmo é fundamentado no teorema de Bayes. Ele faz parte do grupo de classificadores probabilístico de Aprendizado de Máquina. Deste modo, ele utiliza a probabilidade anterior conforme as amostras de classes fornecidas. Desta maneira, quando uma nova amostra é inserida, o algoritmo calcula a probabilidade de pertencer a alguma das classes anteriormente treinadas (BISHOP, 2006; SAHOO *et al.*, 2018; OCCHIPINTI; ROGERS; ANGIONE, 2022).

***Support Vector Machine* (SVM)**: o classificador é caracterizado dentro dos modelos lineares. O algoritmo se baseia em encontrar o hiperplano que oferece a distância máxima de separação entre as classes, que seja capaz de classificar claramente os pontos de dados (KOUTROUMBAS; THEODORIDIS, 2008; WANG *et al.*, 2016; SAHOO *et al.*, 2018; OCCHIPINTI; ROGERS; ANGIONE, 2022).

***Random Forest* (RF):** o algoritmo consiste na combinação de várias árvores de decisão em que cada classificador é construído a partir de dados fornecidos e os resultados são combinados para a realização das previsões. O modelo está contido dentro dos classificadores

que utilizam as técnicas de grafos comumente chamados de árvores (BISHOP, 2006; DENNIS, 2018; SAHOO *et al.*, 2018).

***Multilayer Perceptron* (MLP):** o algoritmo é um modelo de Redes Neural Artificiais. O MLP consiste em várias camadas conectadas por neurônios e que cada camada está conectada à próxima, mapeando um conjunto de entrada em um conjunto de saída (OCCHIPINTI; ROGERS; ANGIONE, 2022).

Pilha de Classificadores (PC): o modelo permite agrupar vários classificadores que serão utilizados como base de treinamento do modelo. O modelo utiliza um algoritmo de meta-aprendizagem para aprender as melhores combinações de previsões entre vários algoritmos. O objetivo do algoritmo é combinar as melhores previsões de entrada para fazer a melhor previsão de saída (GANAIE *et al.*, 2022). Para o modelo proposto para este estudo, os algoritmos utilizados na técnica de empilhamento foram: *Gaussian Naive Bayes*, *Support Vector Machine*, *Random Forest* e *Multilayer Perceptron*.

O Aprendizado de Máquina tem sido implementado com sucesso em várias áreas da ciência da computação tais como: reconhecimento facial e de voz, para calcular valor de ações, padrões climáticos e engenharia de tráfego (TANG *et al.*, 2016; RASOOL *et al.*, 2019). Nas redes de computadores, podem ser usados para maximizar as possibilidades existentes nas Redes Definidas por Software. Dentro dos estudos de segurança cibernética com foco em SDN e ataques DDoS, Yurekten e Demirci (2021) mencionam que a arquitetura SDN possui vantagens quando utilizada com Aprendizado de Máquina. A arquitetura SDN fornece diversas possibilidades de extração e manipulação de pacotes e permite a execução de algoritmos de ML para classificar os fluxos de dados. Essa abordagem, na visão dos pesquisadores, permite que a arquitetura reaja rapidamente aos ataques, controlando o fluxo de pacotes dos *switches*, dado que o controlador possui uma visão ampla da rede de computadores.

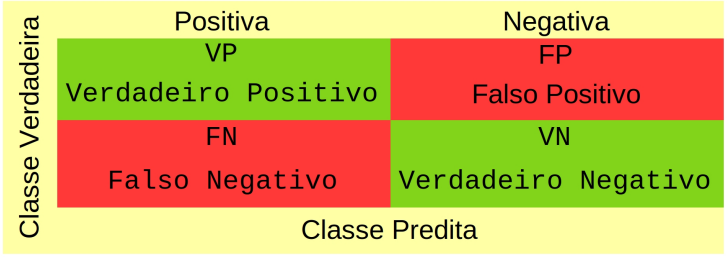
A arquitetura SDN proporciona um ambiente adequado para que o ML possa ser utilizado e seus algoritmos são perfeitamente adaptados para a detecção de intrusões conhecidas e na predição de novos ataques (SCHUELLER *et al.*, 2018; CHENG *et al.*, 2018).

2.3.1 Métricas de avaliação

Matriz de confusão é uma matriz N x N em que N é o número de classes. A matriz de confusão fornece o número de resultados previstos corretos e incorretos do modelo (LIU *et al.*, 2022; GAURAV; GUPTA; PANIGRAHI, 2022). A partir dela são calculadas a acurácia, a precisão e a revocação (SAHOO *et al.*, 2018; DEY; RAHMAN; UDDIN, 2018). Os estudos de Sahoo *et al.* (2018), Dey, Rahman e Uddin (2018), Zeng *et al.* (2022) também utilizaram as mesmas métricas para avaliar seus modelos. A Figura 3 mostra a forma básica da matriz de confusão.

- Verdadeiros positivos (VP): são os valores positivos de ataques de DDoS que foram previstos corretamente.

Figura 3 – Forma genérica da matriz de confusão



Fonte: Adaptado de (SAHOO *et al.*, 2018).

- Falsos negativos (FN): são os ataques de DDoS que foram classificados falsamente como tráfego legítimo.
- Falsos positivos (FP): são os ataques de DDoS que foram classificados verdadeiramente como tráfego malicioso.
- Verdadeiros negativos (VN): são valores negativos de ataques DDoS que foram previstos corretamente.

Acurácia (A): é definida como a percentagem de resultados precisos retornados por um classificador (Equação 1) (SAHOO *et al.*, 2018; DEY; RAHMAN; UDDIN, 2018; ZENG *et al.*, 2022).

$$A = \frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

Precisão (P): tem o objetivo de medir a precisão do modelo em classificar uma amostra como positiva (Equação 2) (SAHOO *et al.*, 2018; DEY; RAHMAN; UDDIN, 2018; ZENG *et al.*, 2022).

$$P = \frac{VP}{VP + FP} \tag{2}$$

Revocação (R): o objetivo da métrica é medir a capacidade do modelo de detectar amostras positivas (Equação 3) (SAHOO *et al.*, 2018; DEY; RAHMAN; UDDIN, 2018; ZENG *et al.*, 2022).

$$R = \frac{VP}{VP + FN} \tag{3}$$

É possível avaliar a eficácia dos modelos preditivos com um conjunto de teste. Deste modo, separa-se uma parte dos dados para efetuar a testagem e a partir dessa técnica pode-se aferir o conjunto utilizado para o treinamento. Em muitos casos, a utilização de treino e teste com percentuais de divisão limitado ou incorretos podem acarretar em modelos superestimados ou subestimados. Uma alternativa viável é a utilização da validação cruzada. No método de validação cruzada, os dados disponíveis são divididos em κ número de amostras. Deste modo,

é definido um conjunto de treinamento e um conjunto de teste para a avaliação (BISHOP, 2006; JAMES *et al.*, 2013; OCCHIPINTI; ROGERS; ANGIONE, 2022).

2.4 Ataque DDoS

O ataque DDoS é definido como o esforço conjunto de vários dispositivos para ocasionar a interrupção de um determinado serviço da Internet (LIU *et al.*, 2018). A fase inicial do ataque se concentra no sequestro de vários dispositivos que estão distribuídos na Internet. Esse sequestro ocorre através da infecção de *malwares* ou vírus de computador e após essas infecções eles são denominados *bots* e estão sob total controle do coordenador do ataque. Um conjunto de *bots* organizados para orquestrar um ataque formam uma rede de *bots* denominada *Botnet*. Os ataques, que são iniciados através de instruções enviadas aos dispositivos contaminados, são constantemente atualizados na medida em que dispositivos de segurança identificam as características e ferramentas de ataque (MAHESHWARI *et al.*, 2022). Essa ação permite que os invasores atualizem seus métodos de invasão, uma vez que possuem acesso às máquinas infectadas, permitindo a constante manutenção da *Botnet* (OO; KAMOLPHIWONG; KAMOLPHIWONG, 2017).

Os ataques DDoS são divididos em 3 categorias (ZHOU *et al.*, 2022): ataques baseados na camada de aplicação, ataques volumétricos e ataques de esgotamento de estado do *Transmission Control Protocol* (TCP). O primeiro explora a camada de aplicação atuando sobre as portas de serviços específicas abertas. No segundo, dispara-se uma quantidade excessiva de pacotes contra a camada de rede da infraestrutura da vítima, objetivando a inundação do servidor com solicitações falsas. No último ataque, utiliza-se a fragilidade do protocolo TCP atuando sobre o aperto de mão de três vias (*three-way handshake*) e o envio de pacotes com campos mal-formatados para sobrecarregar ou bloquear a resposta da vítima.

Embora existam exaustivas pesquisas acadêmicas sobre a mitigação e bloqueio de ataques DDoS, na visão de Nascimento *et al.* (2021), os ataques DDoS não podem ser totalmente evitados. O autor argumenta que a utilização do MTD seria a melhor forma de diminuir o impacto do ataque e lançar uma defesa efetiva contra ele, enquanto Yungaicela-Naula *et al.* (2022a) considera o MTD um dos métodos de defesa proativa mais proeminente.

2.5 Trabalhos Relacionados

Motivado pelo crescente número de interrupção de serviços decorrentes de ataques DDoS e seu impacto técnico e econômico, estratégias visando a operação segura de redes de computadores tem atraído a atenção da comunidade científica. Os estudos em Zhang e Thing (2021), Zheng *et al.* (2021), Alhozaimy e Menascé (2022) apresentam uma revisão literária do método de defesa a ataques *Moving Target Defense* (detalhado na Seção 2.1). Zhai e Vam-

voudakis (2021), Azab, Samir e Samir (2022), Babadi e Doustmohammadi (2022), Potteiger *et al.* (2022) investigam o uso de MTD na área de *cyber–physical system*. Azab, Samir e Samir (2022) apresentam um Sistema de Detecção de Intrusão com controladores SDN (detalhado na Seção 2.2) e MTD. Através de controladores virtualizados é realizada a migração em tempo real entre controladores com o objetivo de evitar ataques cibernéticos ou falhas do sistema operacional do *host*. Uma revisão literária sobre *Cyber-Resilient Mechanism* baseado em *Reinforcement Learning* é apresentada em Huang, Huang e Zhu (2022). Este estudo apresenta vários métodos de defesa, como o MTD e o *Defensive Cyber Deception*, e diversos tipos de ataques, tais como ataque *Denial of Service* (DoS), *spoofing* e injeção.

Usando uma combinação de técnicas MTD, o projeto desenvolvido por Alavizadeh *et al.* (2021) apresenta uma arquitetura composta por um conjunto de *hosts* com o objetivo de tornar a topologia imprevisível e confundir o atacante. Foram considerados vários sistemas operacionais com vulnerabilidades pré-definidas. Definem-se 3 *subnets* com controle de tráfego entre as zonas realizado através de *firewall*. O foco do projeto é a proteção do banco de dados que se encontra dentro de uma das zonas e é acessível apenas por *hosts* específicos da rede. Os autores realizam análise de riscos e confiabilidade do sistema. Entretanto, considera-se apenas um atacante que está fora da rede e com foco exclusivo em detecção de invasão a banco de dados. Além disso, o ataque DDoS não é contemplado nos experimentos. Com uma abordagem MTD híbrida baseada na combinação de embaralhamento, redundância e diversidade, Rajakumaran e Venkataraman (2021) visam dificultar a escolha do sistema alvo e a identificação do ambiente local de forma a reduzir a probabilidade do ataque. O ambiente de teste apresentado foi configurado sobre o Amazon Web Services (AWS). Foram utilizadas instâncias do Amazon Elastic Compute Cloud (EC2) para representar servidores *proxy* e servidores *Web*. O encaminhamento das requisições aos servidores *Web* é realizado pelos servidores *proxy*. Esses servidores devem distinguir ataques DoS de solicitações normais. Monitoram-se desvios no tráfego de entrada a partir das variáveis contadoras do protocolo *Simple Network Management Protocol* (SNMP).

Visando o desenvolvimento de estratégias para aumentar a segurança contra espionagem, as pesquisas de Xu *et al.* (2022) usam a randomização de roteamento utilizando técnicas de Aprendizado de Máquina, SDN e MTD. O uso de SDN tem como único objetivo a separação de encaminhamento e controle e favorecer o gerenciamento centralizado. O mapa de randomização de roteamento é gerado a partir de *Deep Reinforcement Learning* (DRL).

Uma arquitetura SDN utilizando a abordagem MTD baseada em tempo é apresentada em Nguyen *et al.* (2022). O controlador SDN é programado para gerenciar o IP (*Internet Protocol* (IP)) dos servidores e alterar a configuração de acordo com políticas de segurança pré-definidas. Considerando um relógio MTD como gatilho, as estratégias de MTD são baseadas em técnicas de embaralhamento (*shuffling*) e redundância de IP de servidores SDN (*redundancy*). O conceito de IP virtual é usado para realizar a alternância entre os servidores.

Diferentemente dos estudos citados, o presente trabalho tem como foco servidores *Web*, considerando-se variações de ataque DDoS, sua detecção e mitigação. Para tal, tem-se como base o MTD e a potencialização da utilização conjunta de controladores SDN e sensores baseados em Aprendizado de Máquina. Pretende-se criar um ambiente dinâmico, capaz de confundir e controlar as ações do atacante e, assim, aumentar a segurança mediante ataques DDoS.

Para facilitar a comparação dos estudos descritos nesta seção e a arquitetura proposta neste trabalho, a Tabela 1 sumariza as investigações que consideram *Moving Target Defense* e como elas se relacionam com ataques DDoS, servidores *WEB*, SDN, sensores e Aprendizado de Máquina.

Tabela 1 – Estudos sobre *Moving Targed Defense* e suas áreas de estudo.

Ano	Literatura	Áreas de Estudo					
		MTD	SDN	Aprendizado de Máquina	Ataque DDoS	Sensor	Servidor Web
2021	Data-based and secure switched cyber–physical systems (ZHAI; VAMVOUDAKIS, 2021)	✓					
2021	Three decades of deception techniques in active cyber defense - Retrospect and outlook (ZHANG; THING, 2021)	✓					
2021	Dynamic defenses in cyber security: Techniques, methods and challenges (ZHENG *et al.*, 2021)	✓					
2021	Evaluating the effectiveness of shuffle and redundancy MTD techniques in the cloud (ALAVIZADEH *et al.*, 2021)	✓	✓				✓
2021	Performance assessment of hybrid MTD for DoS mitigation in public cloud (RAJAKUMARAN; VENKATARAMAN, 2021)	✓			✓		✓
2022	'MystifY": A proactive Moving-Target Defense for a resilient SDN controllerin Software Defined CPS (AZAB; SAMIR; SAMIR, 2022)	✓	✓				
2022	A moving target defence approach for detecting deception attacks on cyber-physical systems (BABADI; DOUSTMOHAMMADI, 2022)	✓					
2022	A formal analysis of performance-security tradeoffs under frequent task reconfigurations (ALHOZAIMY; MENASCé, 2022)	✓					
2022	Moving target defense for the security and resilience of mixed time and event triggered cyber–physical systems (POTTEIGER *et al.*, 2022)	✓		✓			
2022	Reinforcement Learning for feedback-enabled cyber resilience (HUANG; HUANG; ZHU, 2022)	✓		✓			
2022	Moving target defense of routing randomization with deep reinforcement learning against eavesdropping attack (XU *et al.*, 2022)	✓	✓	✓			
2022	Performability evaluation of switch-over Moving Target Defence mechanisms in a Software Defined Networking using stochastic reward nets (NGUYEN *et al.*, 2022)	✓	✓		✓		✓
-	Este trabalho de pesquisa	✓	✓	✓	✓	✓	✓

Fonte: Autoria própria.

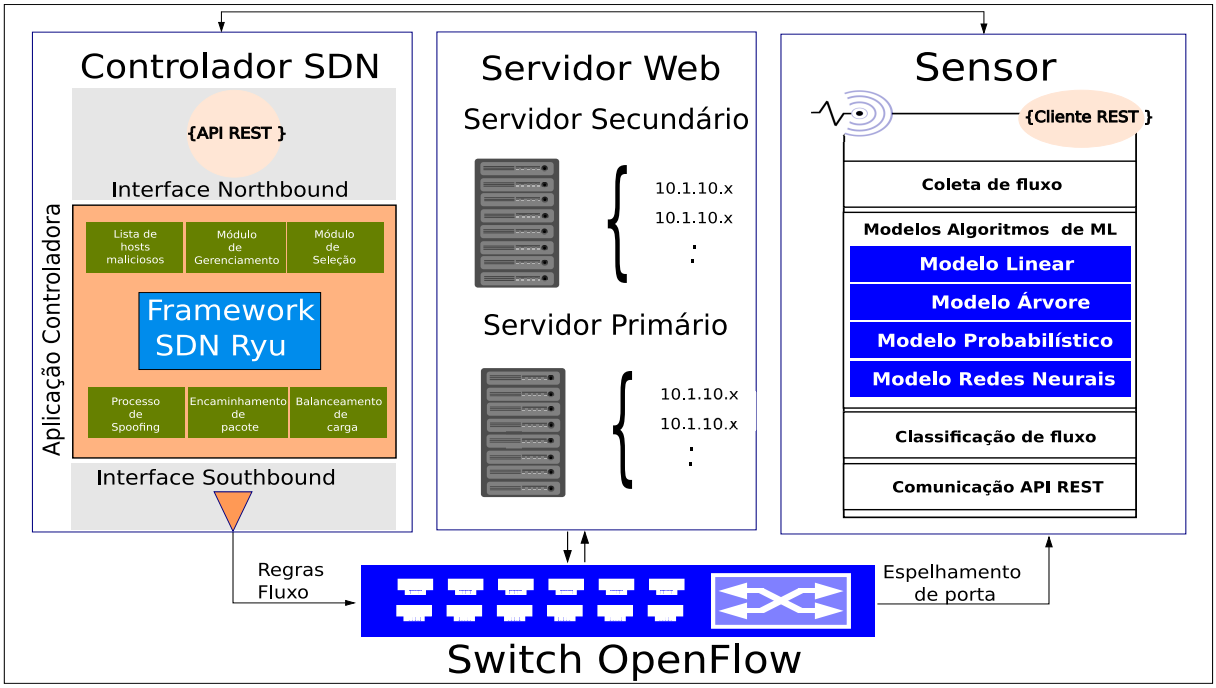
3 ARQUITETURA PROPOSTA

Nesta seção, apresenta-se a arquitetura proposta para detecção e mitigação de ataques DDoS. Inicialmente, apresenta-se uma visão geral da arquitetura proposta (Seção 3.1). Na sequência, o projeto do controlador (Seção 3.2) e do sensor (Seção 3.3) são detalhados.

3.1 Visão geral

A arquitetura proposta (Figura 4) compreende o controlador SDN (Seção 3.2), servidores *Web* primários e secundários, o sensor (Seção 3.3) e o *switch OpenFlow*.

Figura 4 – Arquitetura proposta



Fonte: Autoria própria.

O controlador SDN se comunica com a rede e seus periféricos através da interface *Southbound*, por meio da integração com o protocolo *OpenFlow*. A interface *Northbound* permite a conexão entre o controlador e novos aplicativos ou serviços adicionados ao controlador (NATANZI; MAJMA, 2017; SULTANA *et al.*, 2019).

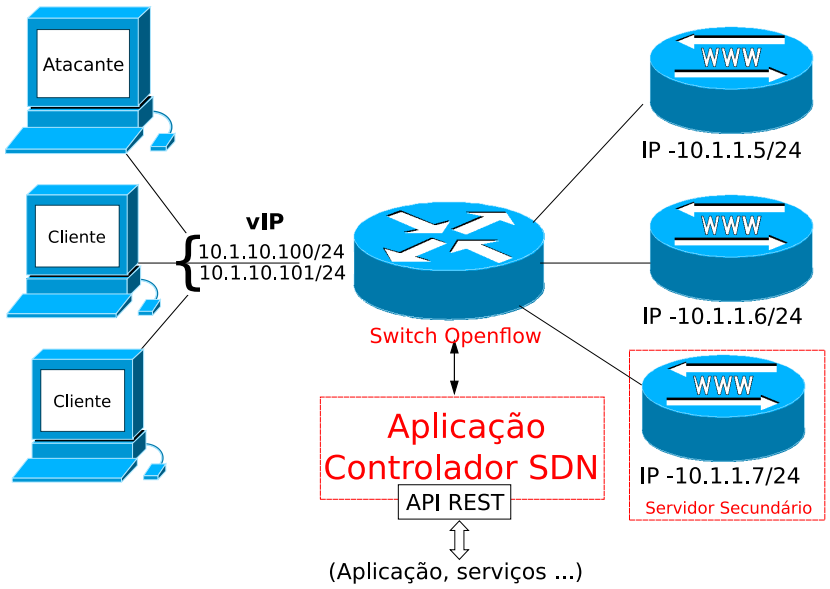
O controlador SDN é responsável por gerenciar o roteamento e o redirecionamento de requisições (Fig. 4 - Aplicação Encaminhamento de pacote) aos servidores *Web* disponíveis (servidores primários e servidores secundários). O redirecionamento de pacotes do atacante para o servidor secundário é baseado no conceito de redundância do *Moving Target Defense*. O controlador possui uma *Application Programming Interface (API)* REST que recebe solicitações de sensores na rede e uma interface virtual de entrada padrão para todas as solicitações aos servidores *Web*. Os sensores são responsáveis por classificar os fluxos de dados obtidos

do *switch OpenFlow* usando algoritmos de aprendizado de máquina. Caso o fluxo seja classificado como malicioso, o sensor solicita ao controlador SDN que inclua o IP de origem do solicitante na lista de *hosts* potencialmente maliciosos, criando uma regra no *switch OF* responsável pelo encaminhamento do fluxo até determinado servidor secundário. No projeto foi utilizado o *OpenvSwitch* (OVS) [1], um *switch OpenFlow* de código aberto, projetado principalmente para funcionar como um comutador virtual em ambientes virtualizados (GORJA; KURAPATI, 2014). O modo de operação configurado no OVS é o modo seguro. Neste caso, se ocorrer um erro no controlador, novos fluxos não serão configurados, nem os existentes serão excluídos. Todos os pacotes serão descartados.

3.2 Controlador SDN

A interação entre o controlador SDN, os clientes e os servidores *Web* é ilustrada na Figura 5. Os clientes *Hypertext Transfer Protocol* (HTTP) acessam os servidores por meio de endereços IP Virtuais (vIP), que o controlador SDN cria em tempo de execução a partir do arquivo de configuração do aplicativo. Além disso, o controlador é responsável pelo balanceamento (Fig. 4 - Aplicação Balanceamento de carga) entre os servidores principais e os secundários.

Figura 5 – Fluxo de acesso aos servidores *Web*



Fonte: Autoria própria.

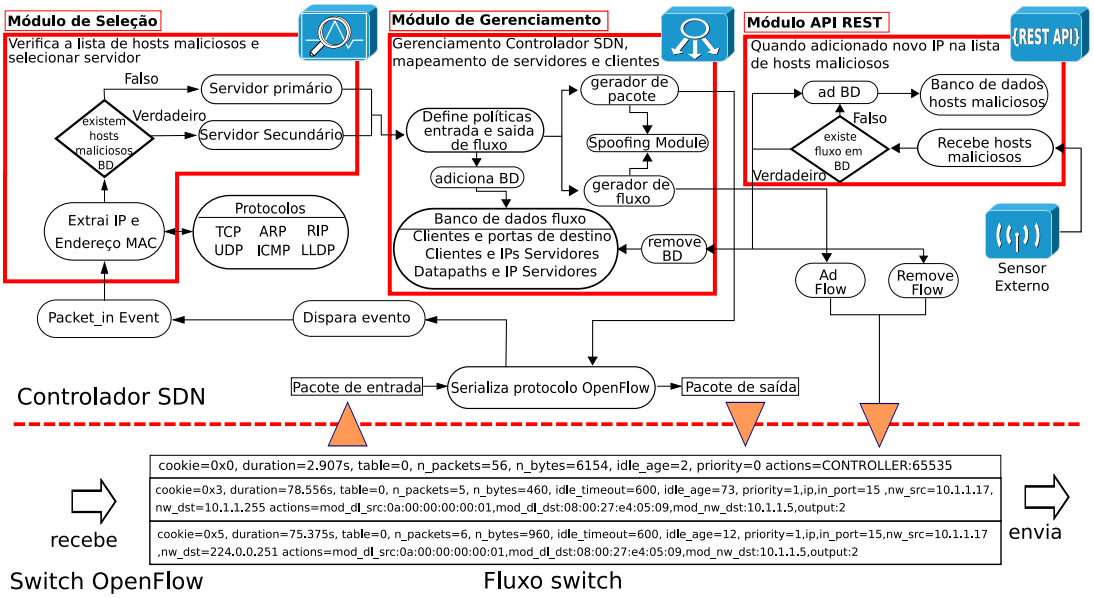
O controlador proposto foi desenvolvido utilizando como base o núcleo do Ryu SDN Framework (Nippon Telegraph and Telephone Corporation, 2012)[2]. O Ryu oferece uma estrutura SDN baseado em componentes, possibilitando alteração, extensão e desenvolvimento de aplicativos controladores personalizados (ALHIJAWI *et al.*, 2022). Para o controlador proposto

[1] Versão 2.4.0. http://www.openvswitch.org/

[2] Versão 4.34 com Apache *License* (https://www.apache.org/) 2.0

foram desenvolvidos 3 módulos sobre os componentes padrões da arquitetura SDN (Fig. 6): Módulo de Seleção (Fig. 4 - Aplicação Módulo de Seleção), Módulo de Gerenciamento (Fig. 4 - Aplicação Módulo de Gerenciamento) e Módulo API REST.

Figura 6 – Arquitetura e fluxo do controlador SDN



Fonte: Autoria própria.

No processo de mitigação, quando novos pacotes chegam ao *switch OpenFlow* (destacado na parte inferior da Figura 6) e não há regras de entrada e saída referenciadas na tabela de fluxo, estes pacotes são redirecionados para o controlador SDN. O evento *Packet_in_Event* é um gatilho nativo do *framework* Ryu acionado quando o controlador recebe uma mensagem OpenFlow *packet_in*. Após o evento a mensagem é encaminhada ao Módulo de Seleção.

3.2.1 Módulo de Seleção

No módulo de seleção (destacado na parte esquerda da Figura 6), inicialmente são coletados os IPs, endereço de controle de acesso ao meio (*Media Access Control* (MAC) *Address*) e a carga útil (*payload*) dos clientes HTTP. Em seguida, verifica-se se o cliente está inscrito na lista de clientes maliciosos (Fig. 4 - Aplicação Lista de *hosts* maliciosos) alimentados por sensores de rede (Seção 3.3). Caso seja classificado como benigno, o fluxo do cliente é encaminhado para um servidor *Web* primário selecionado de forma aleatória. Caso seja classificado como malicioso, o fluxo é encaminhado para um dos servidores secundários escolhido de forma aleatória. Posteriormente, o fluxo é encaminhado para o Módulo de Gerenciamento.

3.2.2 Módulo de Gerenciamento

No Módulo de Gerenciamento (destacado na parte central da Figura 6), define-se, com base nos dados coletados no Módulo de Seleção, as políticas de entrada e saída a serem utilizadas no fluxo e encaminha-se os dados para o processo de *spoofing* (Fig. 4 - Aplicação Processo de *Spoofing*). Neste processo um novo pacote é criado tomando como referência o IP de origem e IP de destino, e o *MAC Address* de origem e destino do pacote. Ao modificar o endereço de origem/destino, oculta-se a identidade do remetente. Processo similar ao *Network Address Translation* (NAT). Então, o pacote criado no *spoofing* é redirecionado ao *switch* OF com a porta e servidor de destino definidos no Módulo de Seleção. Paralelamente, regras de entrada e de saída para o fluxo são criadas, conforme as políticas definidas, e entregues ao *switch OpenFlow*. As regras são criadas na tabela de fluxo do *switch* e adotadas nas próximas requisições do cliente, não necessitando novas interações com o controlador SDN. Uma regra permanece no *switch* OF até atingir o *timeout* do fluxo ou até que seja solicitado a sua exclusão/alteração pelo controlador. O *timeout* padrão do fluxo foi definido em 600 segundos, o padrão do controlador utilizado, e pode ser alterado na inicialização do controlador SDN. Todas as informações sobre os fluxos e regras criadas são armazenadas no banco de dados de fluxos (Figura 6 - Banco de dados fluxos) do Módulo de Gerenciamento para serem manipulados durante a execução do controlador.

3.2.3 Módulo API REST

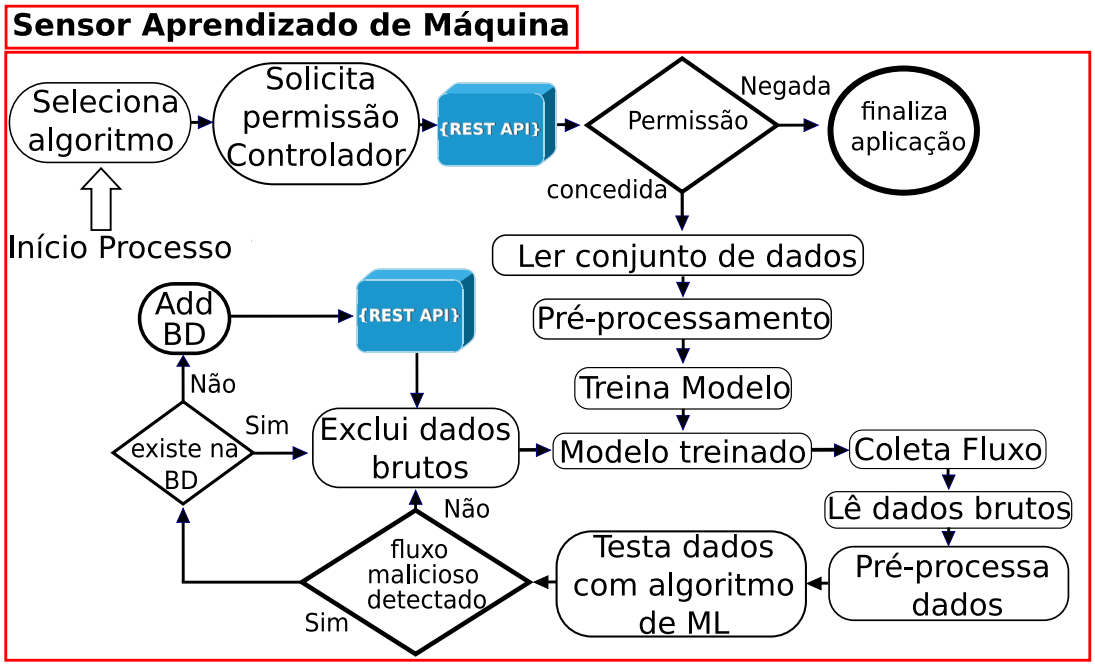
Seguindo recomendações de Sultana *et al.* (2019) sobre modelagem de serviços em redes SDN, a integração entre serviços e o controlador é realizada através da arquitetura REST. O *framework* Ryu possui um serviço *Web* compatível com *Web Server Gateway Interface* (WSGI) que possibilita a vinculação com outros serviços ou aplicações (TEAM, 2021). Partindo desse princípio, foi desenvolvido o Módulo API REST (destacado na parte direita da Figura 6), permitindo que clientes REST acessem os serviços disponíveis no controlador SDN.

Ao classificar um usuário como malicioso, o sensor solicita ao controlador SDN a inclusão deste cliente na lista de *hosts* maliciosos. Esta solicitação é realizada pelos sensores via Módulo API REST, que terá a função de tratar a solicitação e comunicar o Módulo de Gerenciamento. Caso este cliente tenha sido anteriormente classificado como benigno, o Módulo de Gerenciamento solicitará a exclusão das regras anteriormente criadas na tabela de fluxo do *switch OpenFlow* e forçará o processo de criação de novas regras para o fluxo do usuário, agora definido como usuário malicioso.

3.3 Sensor

O sensor (direita na Fig. 4) é responsável por realizar a classificação dos dados e notificar o controlador SDN da rede caso um cliente seja classificado como possível ameaça. O sensor foi desenvolvido na linguagem Python 3.10 e seu funcionamento é detalhado na Figura 7. No desenvolvimento, diversas técnicas e ferramentas foram incorporadas ao fluxo interno da aplicação.

Figura 7 – Diagrama do sensor de classificação de fluxo



Fonte: Autoria própria.

Inicialmente, utilizando a API REST via interface *Northbound*, o sensor solicita ao controlador SDN parâmetros para realizar a comunicação entre o sensor e o controlador (Fig. 7 - Solicita permissão Controlador).

Os sensores são configurados previamente no controlador SDN com login (ID) e *Mac Address*. Em tempo de execução são geradas senhas aleatórias para os sensores, com número de caracteres customizáveis. No momento da requisição, o sensor solicitará autenticação remota com login e senha, juntamente enviará o *hostname* do sensor, *Mac Address* e o campo de contador de pacote iniciado com 0 (zero). Serão validados no controlador as solicitações de autenticação, *Mac Address* e contador de pacotes no controlador que armazenará o *hostname* do sensor. O controlador responderá com ID, DPID (Datapaths ID [3]) e contador de pacotes incrementado de 1 (um). O sensor armazenará os dados para utilização nas próximas conexões. Nas próximas solicitações do sensor serão realizadas novas autenticação no controlador SDN com o login e senha, o ID do sensor, *hostname*, *Mac Address*, DPID e o controle de sequên-

[3] Datapaths ID é uma identificação exclusiva da conexão do *swith Openflow* com o controlador SDN (CHAVES; GARCIA; MADEIRA, 2016).

cia do contador de pacote. Na mensagem constará IP do usuário suspeito para a inclusão na lista de *hosts* maliciosos. Em caso de divergência na validação dos dados do sensor, ele é automaticamente desabilitado.

Algoritmos de Aprendizado de Máquina previamente selecionados (Seção 2.3) são usados para classificar o fluxo de dados como benigno ou malicioso. O treinamento dos algoritmos de Aprendizado de Máquina é realizado utilizando o *dataset* CICDDOS2019 (SHARAFALDIN *et al.*, 2019). Segundo Zeng *et al.* (2022), o CICDDOS2019 é um conjunto de dados que reduz as limitações encontrados em outros *datasets*, refletindo os padrões reais considerando integralmente os aspectos do ataque DDoS em relação a estrutura da rede. Estes dados compreendem diferentes tipos de ataques DDoS (PortMap, NetBIOS, LDAP, MSSQL, TCP, UDP, SYN, NTP, DNS e SNMP) e rotulados considerando 80 características. Os dados disponibilizados por Sharafaldin *et al.* (2019) foram compilados em um conjunto de dados único, sendo utilizados 12.794.627 registros, balanceados em fluxos benignos e maliciosos (DDoS).

A coleta de dados (Fig. 7 - Coleta fluxo) é realizada através da ferramenta Tcpdump [4] através do espelhamento de porta *default gateway* do *switch OpenFlow*.

A partir dos dados coletados, as características existentes no conjunto de dados de treino são pré-processadas utilizando a iniciativa CICFlowMeter (DRAPER-GIL *et al.*, 2016; LASHKARI *et al.*, 2017). Os dados pré-processados são entregues aos algoritmos de Aprendizado de Máquina para análise. Caso o usuário seja classificado como malicioso é solicitado ao controlador SDN a inclusão na lista de usuários maliciosos. Para reduzir a carga de processamento no controlador SDN e o tráfego da rede de computadores, os usuários diagnosticados como maliciosos são armazenados na base de dados do sensor. Deste modo, somente o primeiro diagnóstico é comunicado ao controlador, os próximos são somente contabilizados, reduzindo o desperdício de recursos com envio de informações redundantes.

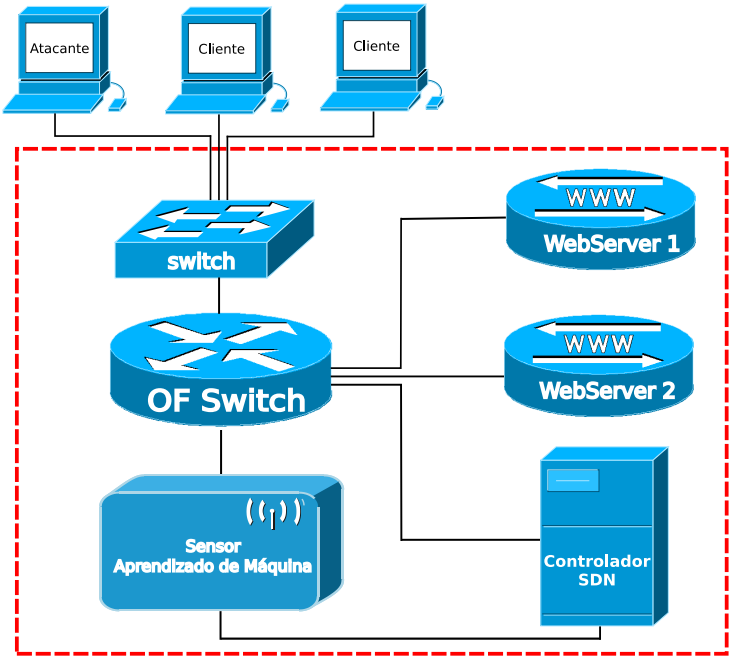
[4] https://www.tcpdump.org/

4 AVALIAÇÃO DA ARQUITETURA

Para avaliar a arquitetura proposta foram realizados experimentos baseados em simulação, possibilitando a verificação da capacidade de detecção e mitigação de ataques DDoS em servidores *Web* nas Redes Definidas por Software.

Para os experimentos foi utilizado um computador com Core i7 4.90 GHz com 32Gb de memória e sistema operacional Linux 5.0.0-38. A topologia de rede (Figura 8) foi criada utilizando a ferramenta para emular rede GNS3 [1], utilizando máquinas virtuais. Para o controlador SDN foi utilizado o sistema operacional Linux 4.14.0-128 com 1024 MB. Foram instalados dois servidores *Web* para serem alvos de ataques *WebServer 1* (servidor principal) e *WebServer 2* (servidor secundário). O primeiro possui 2048 MB de memória e dois núcleos de processamento, o segundo possui 1024 MB de memória e um núcleo de processamento, ambos com sistema operacional Linux 5.3.0-28 e servidor Web Apache[2] 2.4.38. O sensor possui 1024 MB de memória e sistema operacional Linux 4.15.0-132. O código fonte das aplicações desenvolvidas para este estudo pode ser verificado no repositório GitHub [3].

Figura 8 – Topologia usada na avaliação de desempenho



Fonte: Autoria própria.

Os resultados da avaliação são apresentados considerando $i)$ a capacidade em manter a transparência no processo de direcionamento MTD (Seção 4.1), $ii)$ o comparativo de desempenho entre diferentes algoritmos de ML usados (Seção 4.2), e $iii)$ a capacidade de detecção

[1] https://gns3.com/
[2] https://httpd.apache.org/
[3] https://github.com/marcostiribeiro/MovingTargetDefenseProject/

e mitigação de ataques DDoS, observando-se os tempos de resposta e o impacto nos recursos utilizados (Seção 4.3).

4.1 Verificação da manutenção de transparência no processo MTD

Na implementação do MTD, é necessário considerar o desempenho e a qualidade dos serviços apresentados e a eficácia dos modelos desenvolvidos (NGUYEN *et al.*, 2022). O redirecionamento de pacotes entre clientes HTTP e servidores *web* deve ocorrer de forma transparente sem que o invasor possa detectá-los (FAN; FERNANDEZ, 2017). Caso o atacante verifique que o ataque está ocorrendo em um ambiente indefinido, ele irá encerrar o ataque e ocultar as evidências. Deste modo, o processo de *spoofing* adquire importância fundamental nos processos de redirecionamentos do MTD.

Para validar o modelo com foco na qualidade e eficiência foram realizados testes com a ferramenta de diagnóstico NMAP [4] no servidor *Web* para averiguar a efetividade do processo de *spoofing* durante o balanceamento entre o servidor principal e o servidor secundário.

A Tabela 2 apresenta um comparativo entre o servidor principal e o servidor secundário. Os dados mostram os parâmetros do servidor físico. A coleta foi realizada antes e após o redirecionamento para o servidor secundário. Os resultados mostram que o processo ocultou com sucesso os servidores *Web* dos clientes, expondo somente os vIPs com suas respectivas portas de serviço abertas. O endereço IP examinado respondeu com latência e saltos (*hops*) igual nos dois servidores. Foi apresentados somente a porta padrão de acesso ao servidor *Web* (80/tcp http) e 99 portas apresentaram o estado fechado (*closed*) em ambos os servidores. As métricas apresentadas mostram que após o redirecionamento realizado pelo processo do MTD, os usuários tiveram a percepção que estavam consumindo os serviços do servidor primário, enquanto que o conteúdo era disponibilizado pelo servidor secundário.

Tabela 2 – Parâmetros dos servidores físicos

Métricas Coletadas	Servidor Primário	Servidor Secundário
IP Examinado	10.1.10.100	10.1.10.100
Latência	0.0011s	0.0011s
Portas	99 closed	99 closed
Distância da rede (Saltos)	1 hop	1 hop
Média Saltos Round-trip time (RTT)	1.00 ms	1.15 ms
Portas Abertas	80/tcp http	80/tcp http

Fonte: Autoria própria.

[4] https://nmap.org/

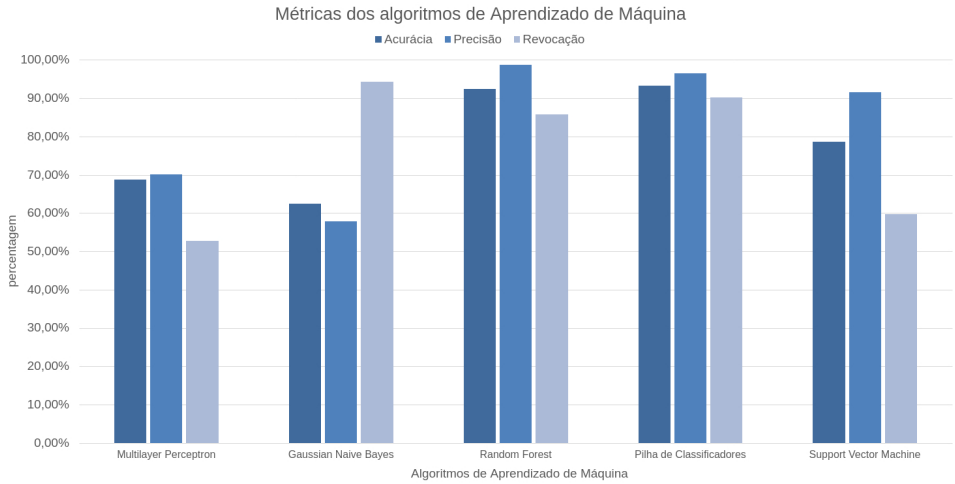
4.2 Desempenho dos algoritmos de Aprendizado de Máquina

Para avaliar o desempenho dos algoritmos de Aprendizado de Máquina utilizados nos sensores foram usadas três métricas: acurácia, precisão e revocação (Seção 2.3.1).

A validação cruzada foi a técnica utilizada para avaliar a eficácia dos modelos preditivos usados no sensor proposto neste trabalho. Neste trabalho, o número de amostras utilizadas nos testes foi definido em $\kappa = 5$, o valor padrão.

O Figura 9 apresenta os resultados de acurácia, precisão e revocação para os algoritmos usados.

Figura 9 – Resultado das métricas dos algoritmos de Aprendizado de Máquina



Fonte: Autoria própria.

Na métrica Acurácia, o algoritmo que conseguiu o melhor resultado foi a Pilha de Classificadores com 93,31% seguido pelo classificador *Random Forest* com 92,36%. Os algoritmos *Support Vector Machine*, *Multilayer Perceptron* e *Gaussian Naive Bayes* obtiveram, respectivamente, 78,57%, 68,69% e 62,42% de acurácia.

Na métrica Precisão, os algoritmos que obtiveram os piores resultados foram *Gaussian Naive Bayes* com 57,90% e o *Multilayer Perceptron* com 70,07%. O S*upport Vector Machine* obteve 91,5%, a Pilha de Classificadores alcançou 96,43% e o *Random Forest* com 98,62% com o melhor desempenho.

Na métrica Revocação, o classificador Pilha de Classificadores atingiu 90,11% nessa métrica, melhor que o classificador *Random Forest* que obteve 85,81%. Os que obtiveram os piores resultados foram os classificadores *Multilayer Perceptron* (52,78%) e *Support Vector Machine* (59,8%). O *Gaussian Naive Bayes* teve o melhor resultado entre todos os classificadores com 94,31% de revocação.

4.3 Detecção e mitigação de DDoS

A avaliação do sistema proposto foi realizada considerando os ataques listados na Tabela 3. Para gerar ataques baseados em fluxo, em protocolos e na camada de aplicação foram usadas as ferramentas, Slowhttptest [5], Scapy [6] e Hping [7]. Estas ferramentas possuem a capacidade de manipular pacotes, explorar vulnerabilidades de protocolos e estressar conexões.

Tabela 3 – Lista de ataques realizados

Nome do ataque	Tipo de vetor	Ferramenta utilizada
Bad TCP *flags* (*All Flags Set*)	*Bad Header* - TCP	Scapy
FIN *Only* Set	*Bad Header* - TCP	Scapy
SYN and FIN *Set*	*Bad Header* - TCP	Scapy
TCP SYN *Flood*	*Flood*	Scapy/Hping
Slow HTTP *POST*	HTTP *Protocol*	Slowhttptest

Fonte: Autoria própria.

O tempo de resposta da arquitetura proposta foi aferido de forma a verificar a eficiência do sensor em detectar o ataque, comunicar o controlador SDN e mitigar o ataque. O tempo médio para cada sequência de teste é definido conforme a Equação (4):

$$\overline{x} = \frac{1}{N} \sum_{n=1}^{N} (mitigTime - initAttack) \tag{4}$$

onde N=20 é número de amostras coletadas para o ataque, *initAttack* é o instante de tempo em que o ataque é iniciado, *mitigTime* é o instante de tempo em que o ataque é mitigado, e $\overline{x}$ é a média de tempo dos ataques realizados.

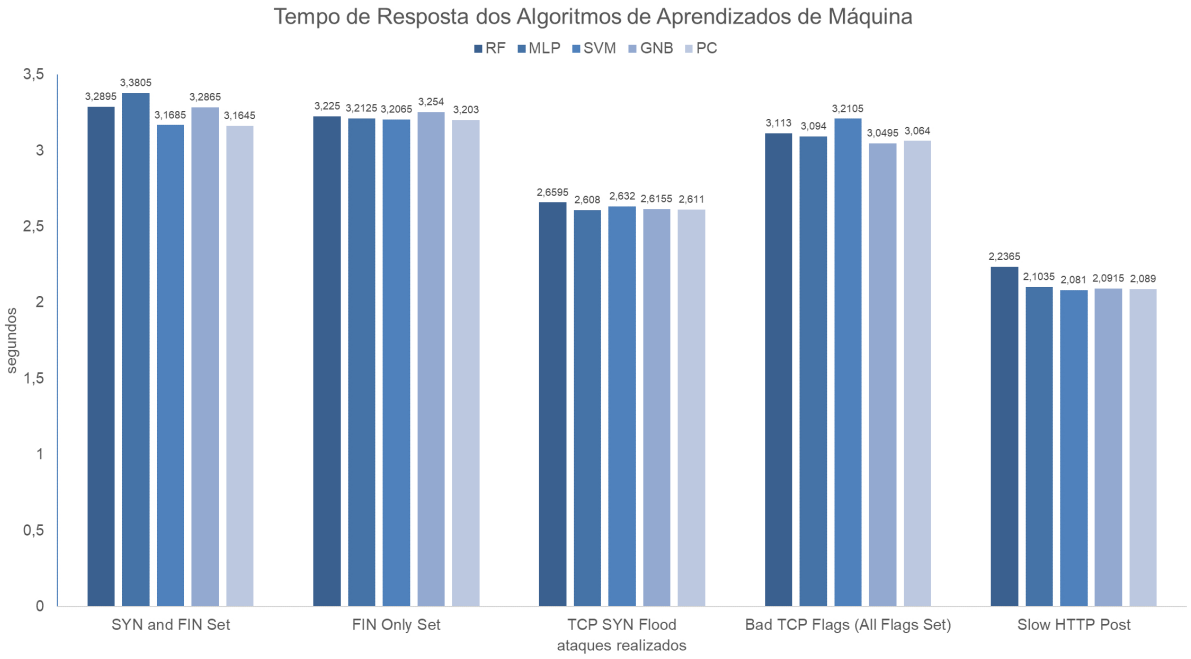
O tempo médio de resposta (em segundos) do sensor, considerando diferentes ataques (Tabela 3) e algoritmos ML (Seção 2.3), é apresentado na Figura 10.

O tempo médio para detecção de ataques considerando todos os algoritmos foi 2,8660 segundos (Figura 10). O nível de confiança usado é de 95% para os resultados apresentados. O modelo Pilha de Classificadores (PC) foi o que apresentou maior velocidade na detecção de ataques (tempo médio de resposta de 2.8263 segundos), seguido por *Gaussian Naive Bayes* (2,8594 segundos), *Support Vector Machine* (2,8597 segundos), *Multilayer Perceptron* (NN) (2,8797 segundos) e *Random Forest* (2,9047 segundos). A proximidade nos tempos de resposta gerados mostra que a utilização de algoritmos de famílias distintas não interfere na agilidade da classificação dos sensores.

Em relação à detecção dos diferentes tipos de ataques considerados, o modelo proposto apresentou os menores tempos de resposta na detecção de ataques *Slow* HTTP *Post* e TCP SYN *Flood* (Figura 10). Os ataques baseados em inundação (por exemplo, o TCP SYN *Flood*)

[5] https://github.com/shekyan/slowhttptest/
[6] https://scapy.net/
[7] http://www.hping.org/

Figura 10 – Tempo médio de resposta (em segundos) dos sensores com diferentes classificadores



Fonte: Autoria própria.

são os tipos mais frequente de ataque DDoS utilizado contra grandes companhias e o que possui maior efetividade (CIL; YILDIZ; BULDU, 2021). Por sua vez, o ataque *Slow* HTTP *Post* é de difícil diagnóstico devido sua semelhança com fluxos de usuários legítimos (HONG *et al.*, 2017). Assim, os resultados gerados evidenciam a capacidade e agilidade do modelo proposto em detectar e mitigar ataques de alta incidência e de difícil detecção.

Por questões de semelhança nos gráficos gerados, alguns resultados são apresentados nesta seção, e os demais resultados podem ser verificados nos apêndices deste documento. Foram realizados testes alternados entre os sensores e os ataques propostos (Tabela 3). Os arquivos fontes (arquivos *Comma-separated values* (CSV)) dos gráficos podem encontrados na repositório GitHub [8] incluindo versão dos gráficos na língua inglesa.

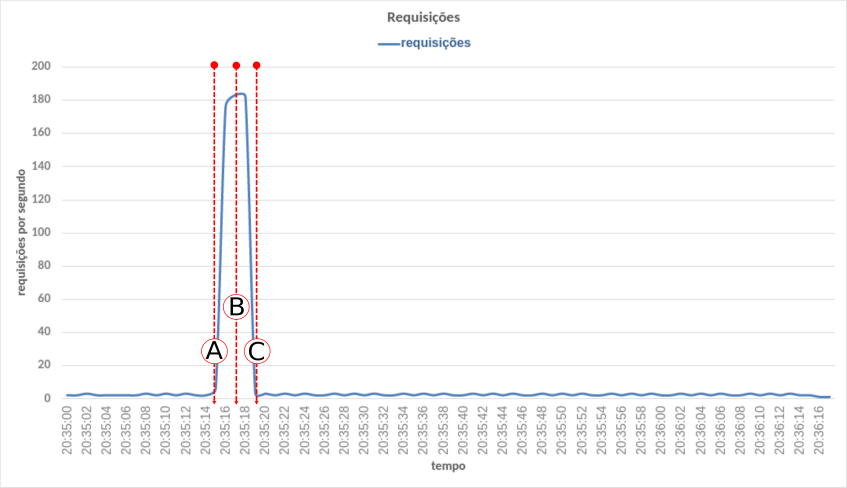
As Figuras 11 e 12 mostram resultados gerados considerando o ataque *Slow* HTTP *POST* e sensor com o classificador *Random Forest*.

Na Figura 11.a, apresenta-se o número de requisições por segundo enviadas ao servidor principal na linha de tempo. O início do ataque ocorre no ponto A e no ponto B ocorre o pico de acessos (183 requisições/segundo), momento em que ocorre a reação do sensor e o início do redirecionamento do fluxo para o servidor secundário. O tempo entre o início do ataque e sua detecção é 2 segundos. Entre o início da reação do classificador e a mitigação do ataque (ponto C) no servidor principal passaram-se 2 segundos.

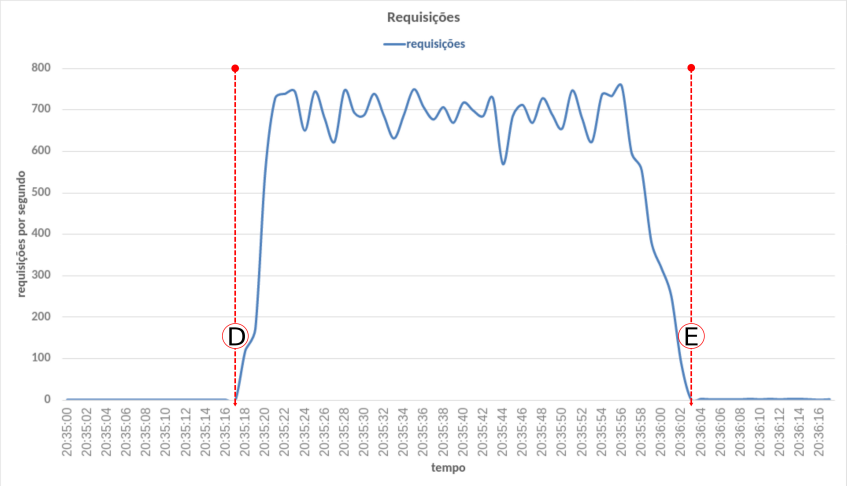
[8] https://marcostiribeiro.github.io/MovingTargetDefenseProject/

Figura 11 – Comportamento dos servidores durante o ataque *Slow* HTTP *POST* e classificador *Random Forest*

(a) Requisições ao servidor primário



(b) Requisições ao servidor secundário



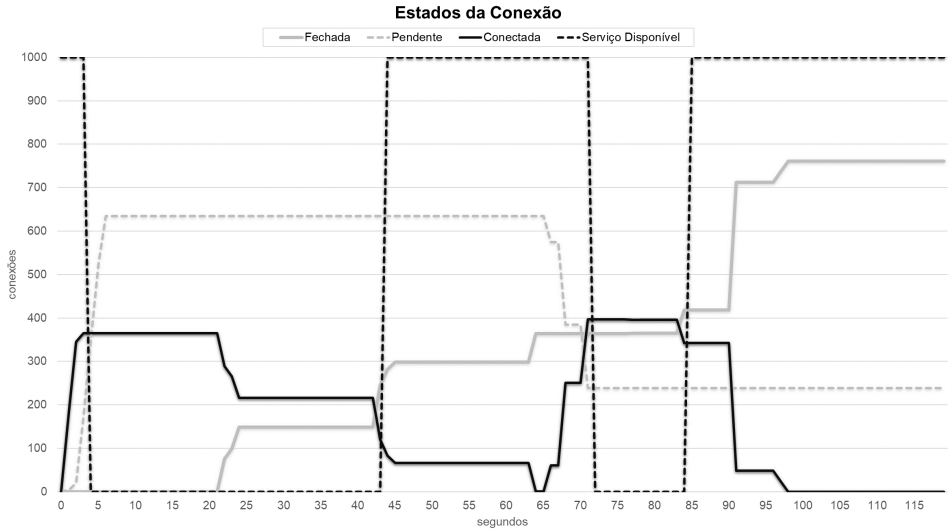
Fonte: Autoria própria.

A Figura 11.b mostra (no ponto D) o momento que ocorre o início e a evolução do redirecionamento de requisições para o servidor secundário. No ápice do ataque o número de requisições ao servidor secundário alcançou 759,9 requisições por segundo. O ataque é concluído 46 segundos após o início do redirecionamento para o servidor secundário (ponto E).

A Figura 12 apresenta o resultado do ataque realizado ao servidor, ou seja, o comportamento da disponibilidade de serviço e os diferentes estados das conexões no decorrer do ataque. Percebe-se que o serviço esteve indisponível para o usuário malicioso em dois momentos: entre 4 e 43 segundos, e entre o instante 72 e o instante 84 segundos comprovando a efetividade da ataque realizado contra o servidor. Em outros momentos as conexões com o servidor ficaram aguardando resposta (Pendente) por determinado tempo, atingindo picos de 635 conexões pendentes entre o instante 7 e o instante 68. O servidor também apresentou estado Fechado, ou seja, recusando conexões. No instante 21 ocorreram 145 conexões fechadas e o

pico ocorreu no instante 117 com 761 conexões. Na visão do atacante, a partir dos resultados apresentados, o servidor *Web* foi comprometido e o ataque foi bem sucedido, contribuindo com a estratégia proposta de enganar o atacante sobre a efetividade do ataque.

Figura 12 – Estados do serviço e das conexões durante o ataque *Slow* HTTP *POST* com classificador *Random Forest*.



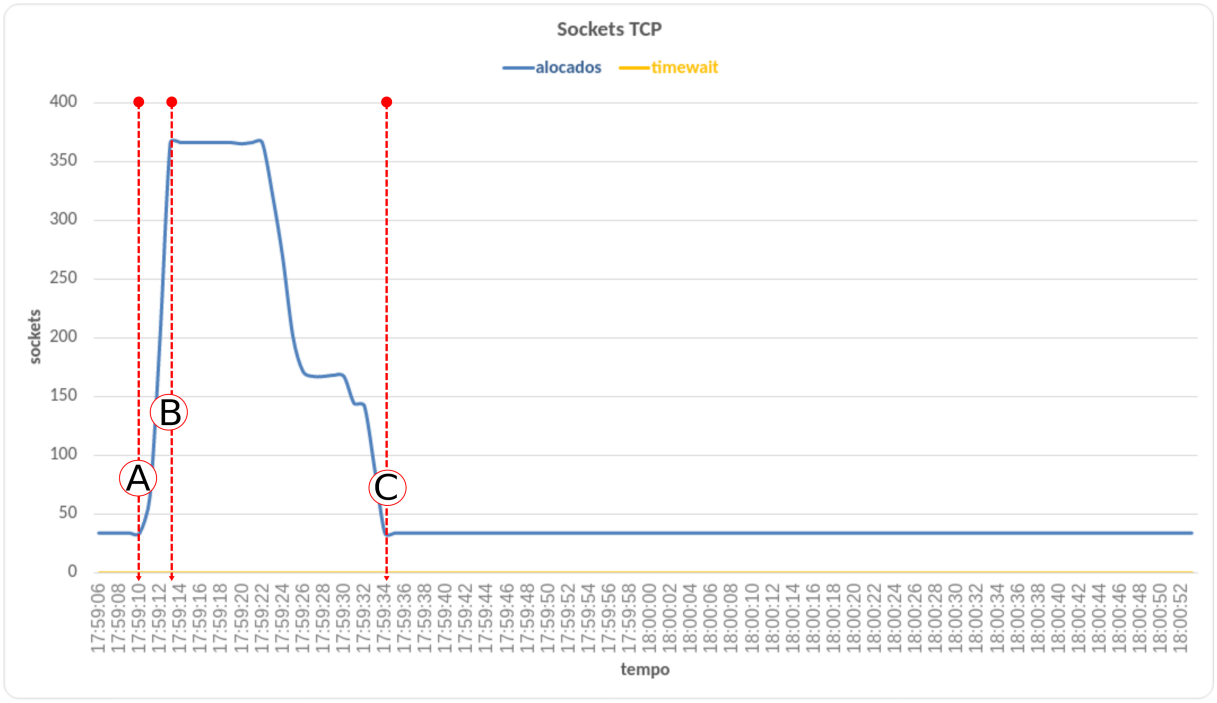
Fonte: Autoria própria.

As Figuras 13 e 14 apresentam o comportamento dos servidores principal (Figuras 13.a e 14.a) e secundário (Figuras 13.b e 14.b) e a reação dos sensores. A coleta dos dados foi realizada simultaneamente nos dois servidores.

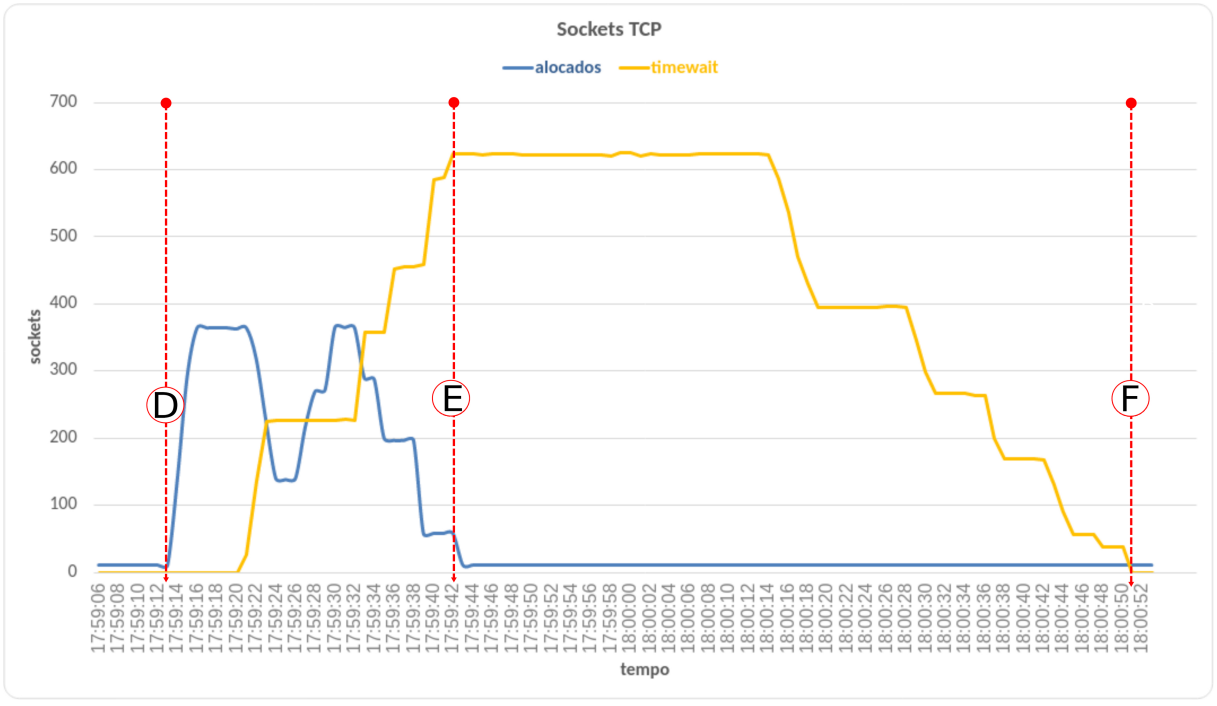
As Figuras 13.a e a Figura 13.b mostram a quantidade de sockets alocados e com estado *timewait* durante a ação do ataque. Estes resultados foram gerados considerando o sensor com classificador SVM e ataque *Slow* HTTP *POST*. Para o servidor principal (Figura 13.a), o intervalo de tempo entre o início do ataque (ponto A) e o instante de pico (ponto B, com 356 *sockets* alocados), quando o sensor identifica o ataque e direciona o fluxo para o servidor secundário, é de 3 segundos. Além disso, 21 segundos após o início do redirecionamento (ponto B), o fluxo de ataque ao servidor principal é encerrado (ponto C). Observa-se, ainda, que não há *sockets* sinalizados com *timewait* no servidor principal, indicando que todos os *sockets* alocados foram consumidos. O servidor secundário (Figura 13.b) recebe o fluxo 3 segundos após o início do ataque (ponto D) culminando com o crescimento exponencial de *sockets*. Constatou-se que o servidor secundário atingiu 621 *sockets* (ponto E) sinalizados com *timewait* no momento de maior volume do ataque, percebe-se a incapacidade do servidor secundário de responder as requisições. O ataque foi totalmente encerrado no ponto F.

As Figuras 14.a e 14.b mostram a quantidade de erros gerados pelo ataque TCP SYN *Flood* nos servidores. O classificador utilizado no sensor foi o *Gaussian Naive Bayes*. O servidor principal sofre ataque, com duração de dois (4) segundos, no período de tempo entre A e C (Figura 14.a). No instante de pico do ataque (ponto B, com 154 requisições/s) o sensor reconhece o ataque e o redireciona ao servidor secundário. Neste instante, o servidor secundário (Figura

Figura 13 – Comportamento do servidor (por *sockets*) durante o ataque *Slow* HTTP *POST*.
(a) *Sockets* alocados e com *timewait* no servidor primário.



(b) *Sockets* alocados e com *timewait* no servidor secundário.

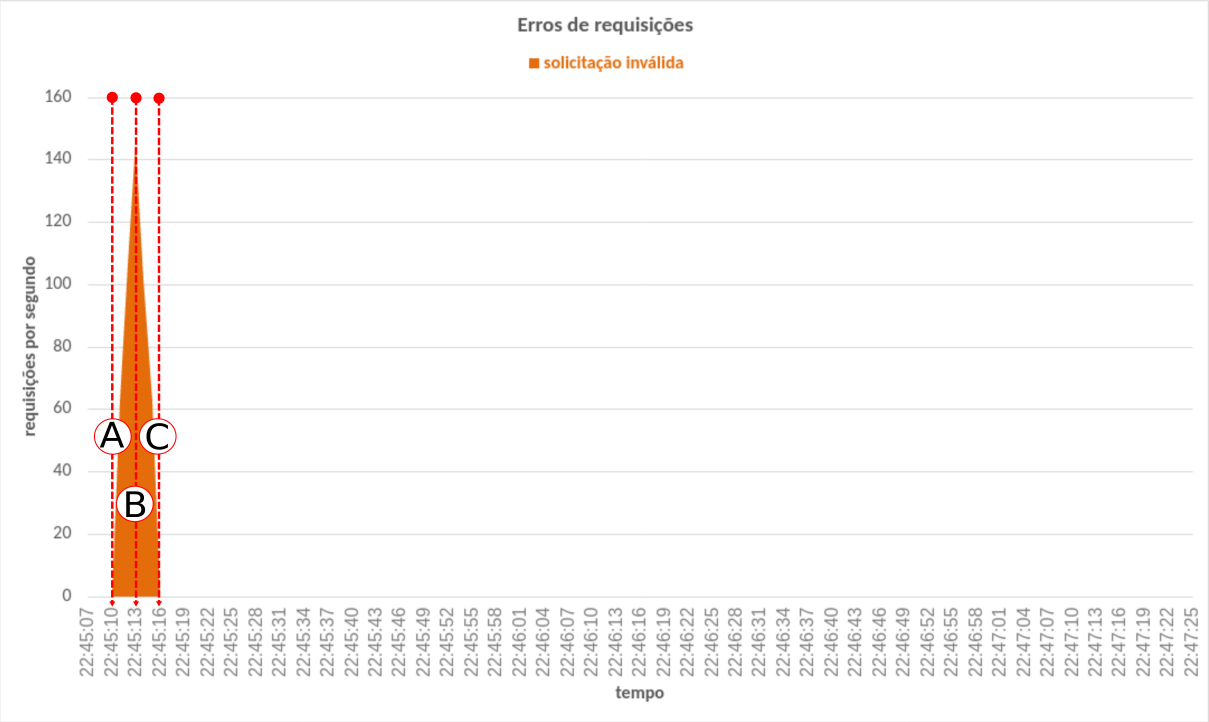


Fonte: Autoria própria.

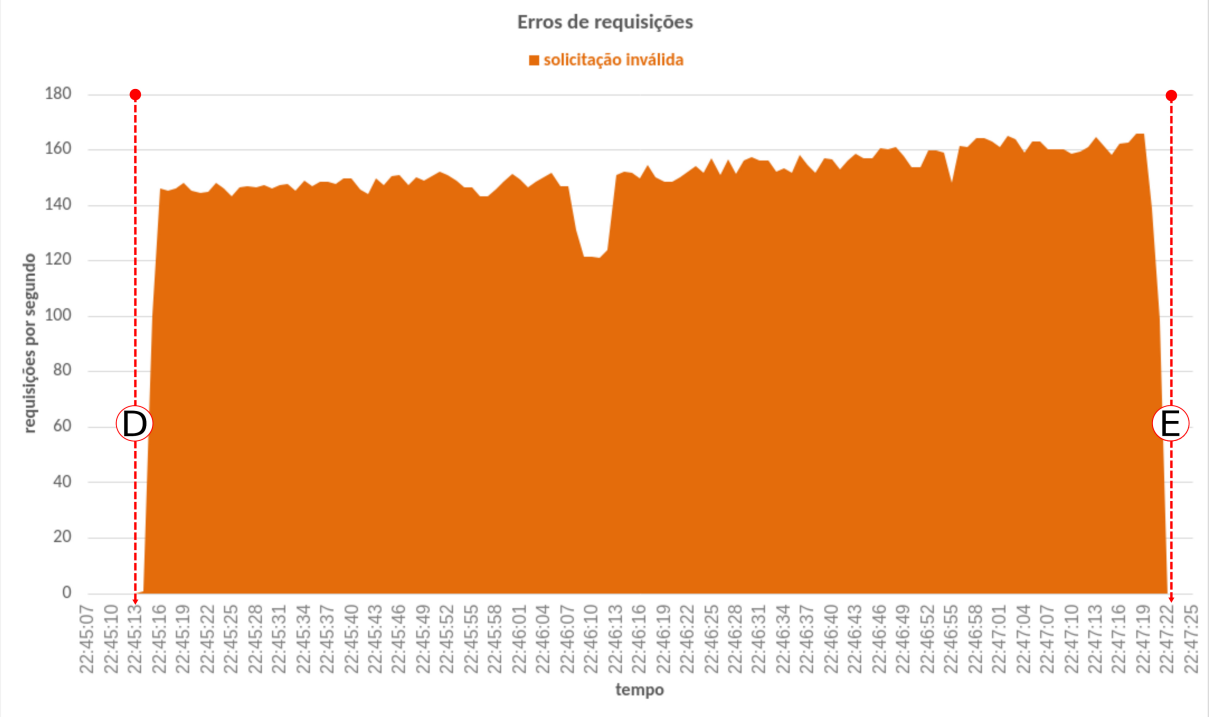
14.b) começa a receber o fluxo do ataque (ponto D), o qual é finalizado 2,1 minutos após o início do ataque (ponto E).

Figura 14 – Comportamento dos servidores durante o ataque TCP SYN *Flood* e o sensor *Gaussian Naive Bayes*

(a) Erros ocorridos no servidor primário



(b) Erros ocorridos no servidor secundário



Fonte: Autoria própria.

4.4 Discussão

Os resultados gerados com o uso da arquitetura proposta mostram que os ataques DDoS podem ser detectados classificando, através de sensores, as solicitações recebidas na rede.

A utilização da técnica MTD mostrou-se transparente para os usuários (Seção 4.1). Através da avaliação de Acurácia, Precisão e Revocação (Seção 4.2) verificou-se que os algoritmos de ML usados pelos sensores mostraram-se eficientes e precisos na detecção de diferentes tipos de ataques de DDoS. Além disso, a diferença no tempo de resposta (Seção 4.3) destes algoritmos é muito pequena, e não impacta no desempenho da arquitetura. De forma geral, o modelo Pilha de Classificadores apresenta os melhores resultados na velocidade de detecção do ataque, o algoritmo *Random Forest* possui melhores resultados na métrica Precisão. Nessa perspectiva, o modelo classificaria mais ataques como falsos positivos, o que resultaria no aumento de usuário encaminhados para o servidor secundário. Outra estratégia seria a utilização do algoritmo *Gaussian Naive Bayes* que obteve os melhores resultados na métrica Revocação, essa métrica evidencia a redução dos falsos negativos. A escolha do algoritmo, conforme as métricas apresentadas, permite que o administrador gerencie a sensibilidade dos modelos no diagnóstico do ataque, impactando diretamente no gerenciamento de recursos que serão utilizados no servidor principal e no secundário.

5 CONCLUSÃO E TRABALHOS FUTUROS

O crescente número de ataques DDoS em servidores de conteúdo e serviços da Internet e o alto custo monetário e de confiança imposto por estes ataques levam à necessidade de mecanismo com a capacidade de detectá-los e mitigá-los de forma rápida e eficiente.

Esta dissertação de mestrado apresenta uma arquitetura modularizada para detectar e mitigar ataques DDoS em tempo real através da utilização conjunta da técnica MTD e sensores baseados em Aprendizado de Máquina em redes SDN. Esta modularização e capacidade de adaptação das tecnologias utilizadas provêm flexibilidade, inclusive para outros ataques de segurança, o que é essencial para ambientes que demandam respostas rápidas a mudanças. Além disso, a experiência do usuário é melhorada, pois caso o usuário tenha um padrão considerado suspeito, ele será redirecionado para um servidor secundário, não evitando o acesso a determinado serviço, mas transferindo-o para um servidor com ambiente controlado. Os resultados dos experimentos mostram que o modelo proposto é $i)$ eficiente em manter a transparência no redirecionamento de fluxos no MTD, $ii)$ que o sensor é rápido e eficiente na classificação de fluxos independentemente do modelo preditivo utilizado e do tipo de ataque realizado, $iii)$ que o sensor é flexível no sentido de que pode usar apenas um classificador, um grupo de classificadores ou outros classificadores diferentes daqueles usados neste trabalho, e que esta é uma decisão do administrador da rede. Evidencia-se, assim, que o modelo proposto é capaz de prover segurança contra ataques DDoS ao servidor principal e no gerenciamento de recursos. Como trabalhos futuros, pretendemos verificar o comportamento da arquitetura proposta considerando outros tipos de ataques, outros tipos de técnicas MTD, outras bases de treinamento, outros algoritmos de predição e aumentar o número de servidores principais e secundários utilizados de forma a verificar a capacidade de escalabilidade da arquitetura.

REFERÊNCIAS

ABHIROOP, T.; BABU, S.; MANOJ, B. S. A machine learning approach for detecting dos attacks in sdn switches. *In*: **2018 Twenty Fourth National Conference on Communications (NCC)**. [*S.l.*: *s.n.*], 2018. p. 1–6.

AL-NAJJAR, A. *et al.* Flow-based load balancing of web traffic using openflow. *In*: IEEE. **2017 27th International Telecommunication Networks and Applications Conference (ITNAC)**. [*S.l.*], 2017. p. 1–6.

ALAVIZADEH, H. *et al.* Evaluating the effectiveness of shuffle and redundancy mtd techniques in the cloud. **Computers Security**, v. 102, p. 102091, 2021. ISSN 0167-4048. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167404820303643.

ALHIJAWI, B. *et al.* A survey on dos/ddos mitigation techniques in sdns: Classification, comparison, solutions, testing tools and datasets. **Computers and Electrical Engineering**, v. 99, p. 107706, 2022. ISSN 0045-7906. Disponível em: https://www.sciencedirect.com/science/article/pii/S0045790622000234.

ALHOZAIMY, S.; MENASCé, D. A. A formal analysis of performance-security tradeoffs under frequent task reconfigurations. **Future Generation Computer Systems**, v. 127, p. 252–262, 2022. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X21003514.

AZAB, M.; SAMIR, M.; SAMIR, E. “mystify”: A proactive moving-target defense for a resilient sdn controller in software defined cps. **Computer Communications**, v. 189, p. 205–220, 2022. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366422000949.

BABADI, N.; DOUSTMOHAMMADI, A. A moving target defence approach for detecting deception attacks on cyber-physical systems. **Computers and Electrical Engineering**, v. 100, p. 107931, 2022. ISSN 0045-7906. Disponível em: https://www.sciencedirect.com/science/article/pii/S0045790622002129.

BADUGE, S. K. *et al.* Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications. **Automation in Construction**, v. 141, p. 104440, 2022. ISSN 0926-5805. Disponível em: https://www.sciencedirect.com/science/article/pii/S0926580522003132.

BHARDWAJ, A. *et al.* Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions. **Computer Science Review**, v. 39, p. 100332, 2021. ISSN 1574-0137. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574013720304329.

BISHOP, C. M. **Pattern recognition and machine learning**. [*S.l.*]: springer, 2006. ISBN 978-0-387-31073-2.

BOERO, L.; MARCHESE, M.; ZAPPATORE, S. Support vector machine meets software defined networking in ids domain. *In*: **2017 29th International Teletraffic Congress (ITC 29)**. [*S.l.*: *s.n.*], 2017. v. 3, p. 25–30.

CELESOVA, B. *et al.* Enhancing security of sdn focusing on control plane and data plane. *In*: **2019 7th International Symposium on Digital Forensics and Security (ISDFS)**. [*S.l.*: *s.n.*], 2019. p. 1–6.

CHAVES, L. J.; GARCIA, I. C.; MADEIRA, E. R. M. Ofswitch13: Enhancing ns-3 with openflow 1.3 support. *In*: **Proceedings of the Workshop on ns-3**. [*S.l.*: *s.n.*], 2016. p. 33–40.

CHENG, Q. *et al.* Guarding the perimeter of cloud-based enterprise networks: An intelligent sdn firewall. *In*: **2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)**. [*S.l.*: *s.n.*], 2018. p. 897–902.

CIL, A. E.; YILDIZ, K.; BULDU, A. Detection of ddos attacks with feed forward based deep neural network model. **Expert Systems with Applications**, v. 169, p. 114520, 2021. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417420311647.

DAYAL, N.; SRIVASTAVA, S. An rbf-pso based approach for early detection of ddos attacks in sdn. *In*: **2018 10th International Conference on Communication Systems Networks (COMSNETS)**. [*S.l.*: *s.n.*], 2018. p. 17–24. ISSN 2155-2509.

DENNIS, M. J. R. Machine-learning and statistical methods for ddos attack detection and defense system in software defined networks. 2018.

DEY, S. K.; RAHMAN, M. M.; UDDIN, M. R. Detection of flow based anomaly in openflow controller: Machine learning approach in software defined networking. *In*: **2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT)**. [*S.l.*: *s.n.*], 2018. p. 416–421.

DRAPER-GIL, G. *et al.* Characterization of encrypted and VPN traffic using time-related features. *In*: CAMP, O.; FURNELL, S.; MORI, P. (Ed.). **Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016**. SciTePress, 2016. p. 407–414. Disponível em: https://doi.org/10.5220/0005740704070414.

FAN, W.; FERNANDEZ, D. A novel sdn based stealthy tcp connection handover mechanism for hybrid honeypot systems. *In*: **2017 IEEE Conference on Network Softwarization (NetSoft)**. [*S.l.*: *s.n.*], 2017. p. 1–9.

GANAIE, M. *et al.* Ensemble deep learning: A review. **Engineering Applications of Artificial Intelligence**, v. 115, p. 105151, 2022. ISSN 0952-1976. Disponível em: https://www.sciencedirect.com/science/article/pii/S095219762200269X.

Gartner, Inc. **Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach Nearly $500 Billion in 2022**. 2022. Last accessed 03 november 2022. Disponível em: https://www.gartner.com/en/newsroom/press-releases/2022-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-500-billion-in-20

GAURAV, A.; GUPTA, B. B.; PANIGRAHI, P. K. A novel approach for ddos attacks detection in covid-19 scenario for small entrepreneurs. **Technological Forecasting and Social Change**, v. 177, p. 121554, 2022. ISSN 0040-1625.

GORJA, P.; KURAPATI, R. Extending open vswitch to l4-l7 service aware openflow switch. *In*: **2014 IEEE International Advance Computing Conference (IACC)**. [*S.l.*: *s.n.*], 2014. p. 343–347.

HONG, K. *et al.* Sdn-assisted slow http ddos attack defense method. **IEEE Communications Letters**, IEEE, v. 22, n. 4, p. 688–691, 2017.

HUANG, Y.; HUANG, L.; ZHU, Q. Reinforcement learning for feedback-enabled cyber resilience. **Annual Reviews in Control**, v. 53, p. 273–295, 2022. ISSN 1367-5788. Disponível em: https://www.sciencedirect.com/science/article/pii/S1367578822000013.

JAMES, G. *et al.* **An introduction to statistical learning**. [*S.l.*]: Springer, 2013. v. 112. ISBN 978-1-4614-7137-0.

KOUTROUMBAS, K.; THEODORIDIS, S. **Pattern Recognition**. [*S.l.*]: Elsevier Science, 2008. ISBN 9780080949123.

KREUTZ, D. *et al.* Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, Jan 2015. ISSN 1558-2256.

KRONGBARAMEE, P.; SOMCHIT, Y. Implementation of sdn stateful firewall on data plane using open vswitch. *In*: IEEE. **2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)**. [*S.l.*], 2018. p. 1–5.

LASHKARI, A. H. *et al.* Characterization of tor traffic using time based features. *In*: MORI, P.; CAMP, S. F. andOlivier (Ed.). **Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISSP 2017, Porto, Portugal, February 19-21,2017**. SciTePress, 2017. p. 253–262. Disponível em: https://doi.org/10.5220/0006105602530262.

LIU, J.; XU, Q. Machine learning in software defined network. *In*: **2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)**. [*S.l.*: *s.n.*], 2019. p. 1114–1120.

LIU, Y. *et al.* Deep reinforcement learning based smart mitigation of ddos flooding in software-defined networks. *In*: **2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)**. [*S.l.*: *s.n.*], 2018. p. 1–6. ISSN 2378-4873.

LIU, Y. *et al.* Software-defined ddos detection with information entropy analysis and optimized deep learning. **Future Generation Computer Systems**, v. 129, p. 99–114, 2022. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X21004386.

MAHESHWARI, A. *et al.* An optimized weighted voting based ensemble model for ddos attack detection and mitigation in sdn environment. **Microprocessors and Microsystems**, v. 89, p. 104412, 2022. ISSN 0141-9331. Disponível em: https://www.sciencedirect.com/science/article/pii/S0141933121005585.

MCCARTHY, J. What is artificial intelligence. **URL: http://www-formal. stanford. edu/jmc/whatisai. html**, 2004. Last accessed 08 november 2022.

MOHAMMED, S. S. *et al.* A new machine learning-based collaborative ddos mitigation mechanism in software-defined network. *In*: **2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)**. [*S.l.*: *s.n.*], 2018. p. 1–8. ISSN 2160-4886.

MOHANAPRIYA, P.; SHALINIE, S. M. Restricted boltzmann machine based detection system for ddos attack in software defined networks. *In*: **2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)**. [*S.l.*: *s.n.*], 2017. p. 1–6.

NASCIMENTO, P. P. do *et al.* A methodology for selecting hardware performance counters for supporting non-intrusive diagnostic of flood ddos attacks on web servers.

Computers Security, v. 110, p. 102434, 2021. ISSN 0167-4048. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167404821002583.

NATANZI, S. B. H.; MAJMA, M. R. Secure northbound interface for sdn applications with ntru public key infrastructure. *In*: **2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)**. [*S.l.*: *s.n.*], 2017. p. 0452–0458.

NGUYEN, T. A. *et al.* Performability evaluation of switch-over moving target defence mechanisms in a software defined networking using stochastic reward nets. **Journal of Network and Computer Applications**, v. 199, p. 103267, 2022. ISSN 1084-8045. Disponível em: https://www.sciencedirect.com/science/article/pii/S1084804521002642.

Nippon Telegraph and Telephone Corporation. **RYU Network Operating System**. 2012. Last accessed 03 november 2022. Disponível em: https://github.com/faucetsdn/ryu.

NYGREN, A. *et al.* Openflow switch specification version 1.5. 1. **Open Networking Foundation, Tech. Rep**, 2015.

OCCHIPINTI, A.; ROGERS, L.; ANGIONE, C. A pipeline and comparative study of 12 machine learning models for text classification. **Expert Systems with Applications**, v. 201, p. 117193, 2022. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417422005802.

OO, M. M.; KAMOLPHIWONG, S.; KAMOLPHIWONG, T. The design of sdn based detection for distributed denial of service (ddos) attack. *In*: **2017 21st International Computer Science and Engineering Conference (ICSEC)**. [*S.l.*: *s.n.*], 2017. p. 1–5.

POTTEIGER, B. *et al.* Moving target defense for the security and resilience of mixed time and event triggered cyber–physical systems. **Journal of Systems Architecture**, v. 125, p. 102420, 2022. ISSN 1383-7621. Disponível em: https://www.sciencedirect.com/science/article/pii/S1383762122000212.

RAJAKUMARAN, G.; VENKATARAMAN, N. Performance assessment of hybrid mtd for dos mitigation in public cloud. **International Journal of Intelligent Networks**, v. 2, p. 140–147, 2021. ISSN 2666-6030. Disponível em: https://www.sciencedirect.com/science/article/pii/S2666603021000191.

RASOOL, R. U. *et al.* Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks. **IEEE Access**, v. 7, p. 34885–34899, 2019. ISSN 2169-3536.

SAHOO, K. S. *et al.* A machine learning approach for predicting ddos traffic in software defined networks. *In*: **2018 International Conference on Information Technology (ICIT)**. [*S.l.*: *s.n.*], 2018. p. 199–203.

SCHUELLER, Q. *et al.* A hierarchical intrusion detection system using support vector machine for sdn network in cloud data center. *In*: **2018 28th International Telecommunication Networks and Applications Conference (ITNAC)**. [*S.l.*: *s.n.*], 2018. p. 1–6. ISSN 2474-154X.

SESHIA, S. A.; SADIGH, D.; SASTRY, S. S. Toward verified artificial intelligence. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 65, n. 7, p. 46–55, jun 2022. ISSN 0001-0782. Disponível em: https://doi.org/10.1145/3503914.

SHARAFALDIN, I. *et al.* Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. *In*: **2019 International Carnahan Conference on Security Technology (ICCST)**. [*S.l.*: *s.n.*], 2019. p. 1–8. ISSN 2153-0742.

SINGH, J.; BEHAL, S. Detection and mitigation of ddos attacks in sdn: A comprehensive review, research challenges and future directions. **Computer Science Review**, v. 37, p. 100279, 2020. ISSN 1574-0137. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574013720301647.

SINGH, M. P.; BHANDARI, A. New-flow based ddos attacks in sdn: Taxonomy, rationales, and research challenges. **Computer Communications**, v. 154, p. 509–527, 2020. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366419313830.

SULTANA, N. *et al.* Survey on sdn based network intrusion detection system using machine learning approaches. **Peer-to-Peer Networking and Applications**, Springer, v. 12, n. 2, p. 493–501, 2019.

TANG, T. A. *et al.* Deep learning approach for network intrusion detection in software defined networking. *In*: **2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)**. [*S.l.*: *s.n.*], 2016. p. 258–263.

TEAM, R. **Ryu Documentation Release 4.34**. RYU project team, 2021. (Release 4.34). Last accessed 03 november 2022. Disponível em: Available:https://ryu.readthedocs.io/_/downloads/en/latest/pdf/.

VALDOVINOS, I. A. *et al.* Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. **Journal of Network and Computer Applications**, v. 187, p. 103093, 2021. ISSN 1084-8045. Disponível em: https://www.sciencedirect.com/science/article/pii/S1084804521001156.

WANG, P. *et al.* An efficient flow control approach for sdn-based network threat detection and migration using support vector machine. *In*: **2016 IEEE 13th International Conference on e-Business Engineering (ICEBE)**. [*S.l.*: *s.n.*], 2016. p. 56–63.

XU, X. *et al.* Moving target defense of routing randomization with deep reinforcement learning against eavesdropping attack. **Digital Communications and Networks**, v. 8, n. 3, p. 373–387, 2022. ISSN 2352-8648. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352864822000037.

YUNGAICELA-NAULA, N. M. *et al.* Towards security automation in software defined networks. **Computer Communications**, v. 183, p. 64–82, 2022. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366421004436.

YUNGAICELA-NAULA, N. M. *et al.* Towards security automation in software defined networks. **Computer Communications**, v. 183, p. 64–82, 2022. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366421004436.

YUREKTEN, O.; DEMIRCI, M. Sdn-based cyber defense: A survey. **Future Generation Computer Systems**, v. 115, p. 126–149, 2021. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X20303277.

ZENG, Z. *et al.* Intrusion detection framework based on causal reasoning for ddos. **Journal of Information Security and Applications**, v. 65, p. 103124, 2022. ISSN 2214-2126.

ZHAI, L.; VAMVOUDAKIS, K. G. Data-based and secure switched cyber–physical systems. **Systems Control Letters**, v. 148, p. 104826, 2021. ISSN 0167-6911. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167691120302085.

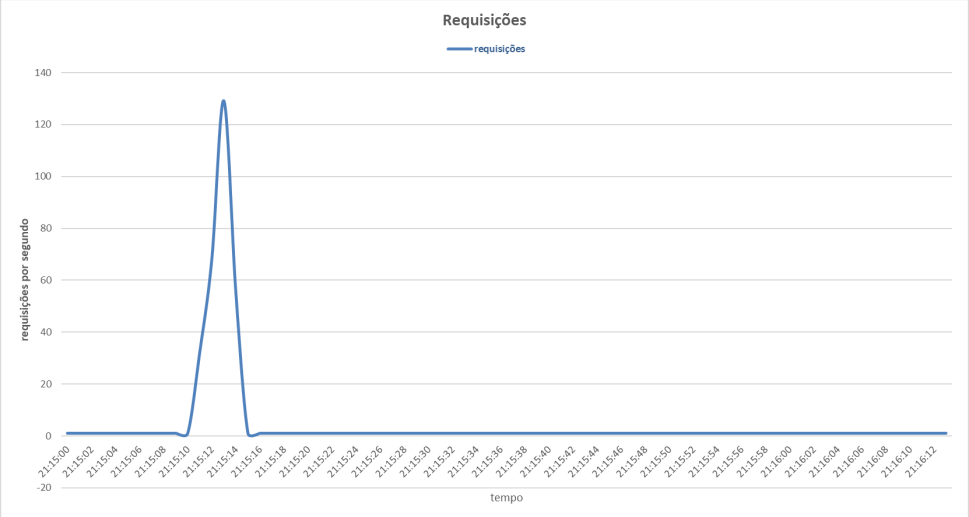
ZHANG, L.; THING, V. Three decades of deception techniques in active cyber defense - retrospect and outlook. **Computers Security**, v. 106, p. 102288, 2021. ISSN 0167-4048. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167404821001127.

ZHENG, Y. *et al.* Dynamic defenses in cyber security: Techniques, methods and challenges. **Digital Communications and Networks**, 2021. ISSN 2352-8648. Disponível em: https://www.sciencedirect.com/science/article/pii/S235286482100047X.

ZHOU, L. *et al.* A feature selection-based method for ddos attack flow classification. **Future Generation Computer Systems**, v. 132, p. 67–79, 2022. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X22000474.

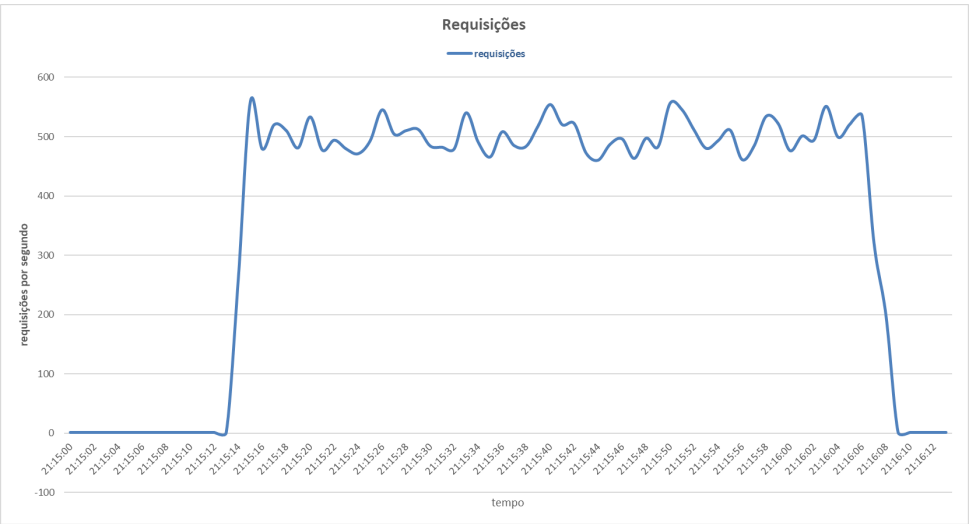
APÊNDICE A – Ataque *Bad* TCP *flags* (*All Flags Set*) com sensor *Random Forest*

Figura 15 – Requisições ao servidor primário durante ataque *Bad* TCP *flags* (All *Flags* Set) com sensor *Random Forest*



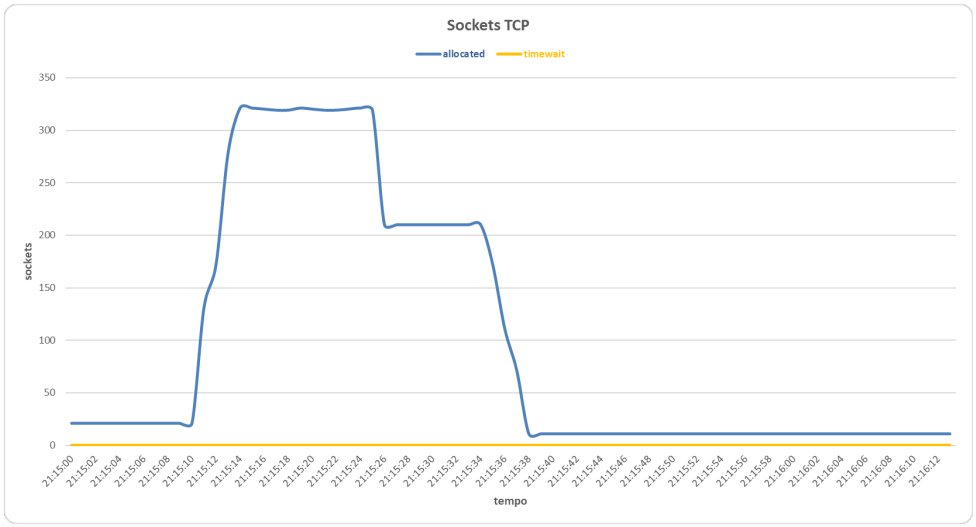
Fonte: Autoria própria.

Figura 16 – Requisições ao servidor secundário durante ataque *Bad* TCP *flags* (*All Flags* Set) com sensor *Random Forest*



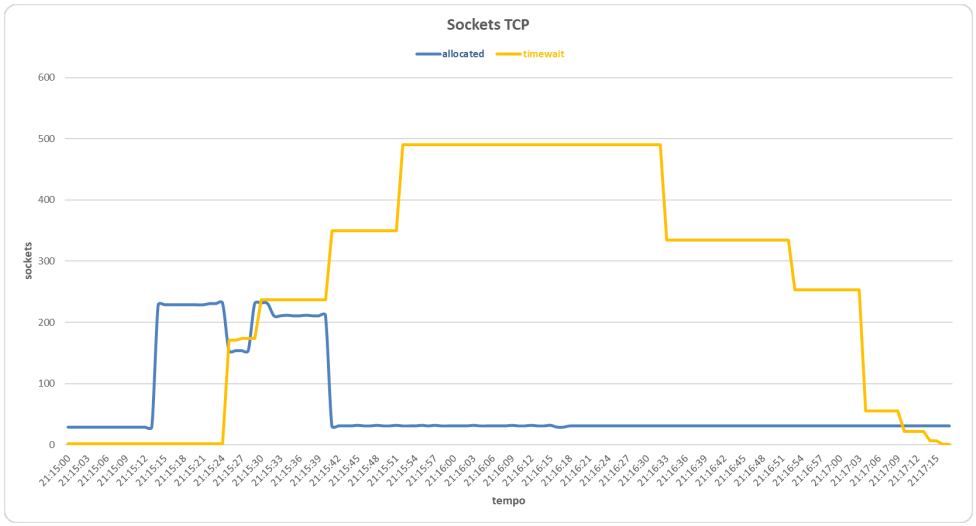
Fonte: Autoria própria.

Figura 17 – *Sockets* alocados e com *timewait* no servidor primário durante ataque *Bad* TCP *flags* (*All Flags* Set) com sensor *Random Forest*



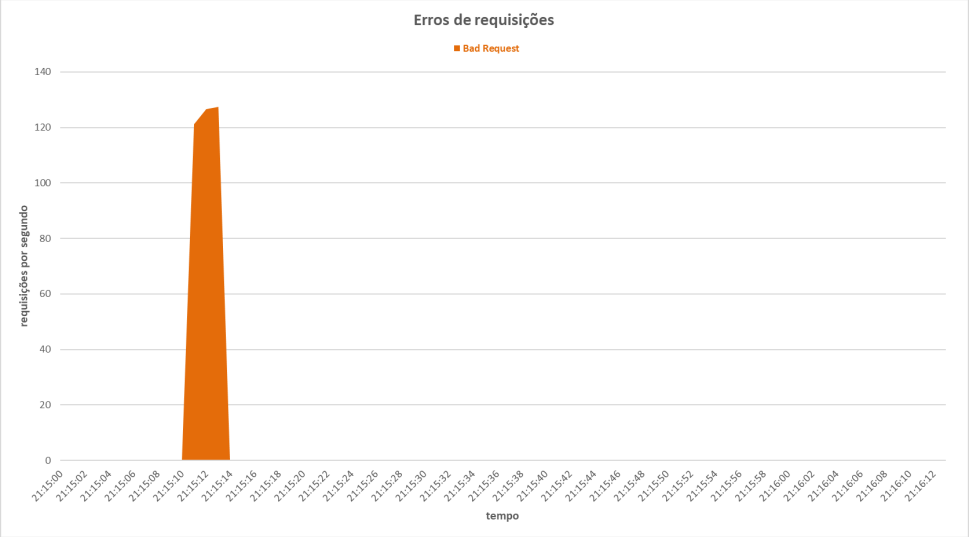
Fonte: Autoria própria.

Figura 18 – *Sockets* alocados e com *timewait* no servidor secundário durante ataque *Bad* TCP *flags* (*All Flags Set*) com sensor *Random Forest*



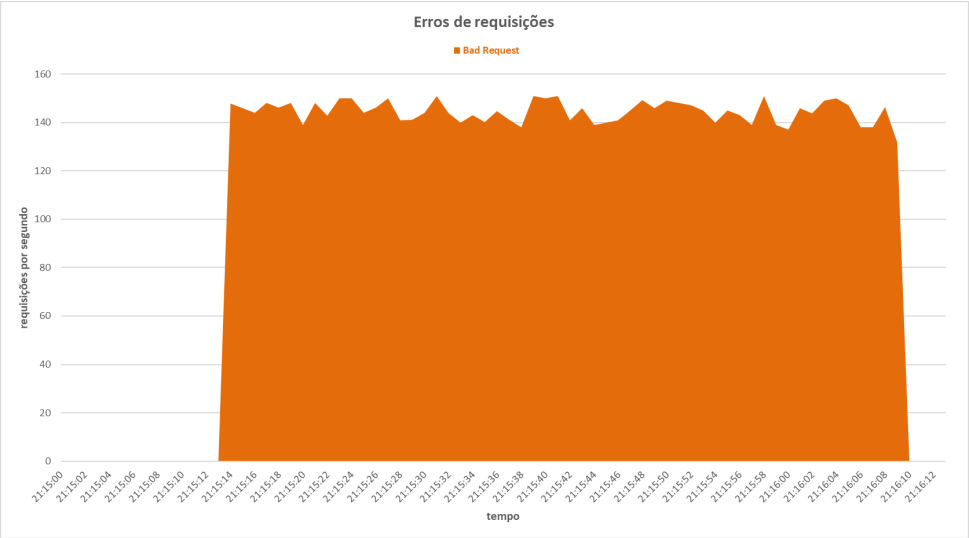
Fonte: Autoria própria.

Figura 19 – Erros de requisições ocorridos no servidor primário durante ataque *Bad* TCP *flags* (*All Flags Set*) com sensor *Random Forest*



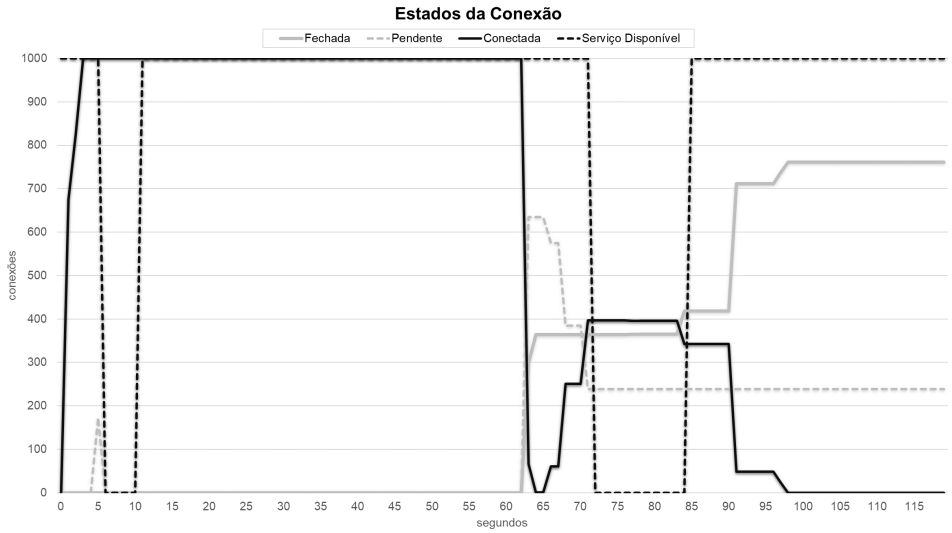
Fonte: Autoria própria.

Figura 20 – Erros de requisições ocorridos no servidor secundário durante ataque *Bad* TCP *flags* (*All Flags Set*) com sensor *Random Forest*



Fonte: Autoria própria.

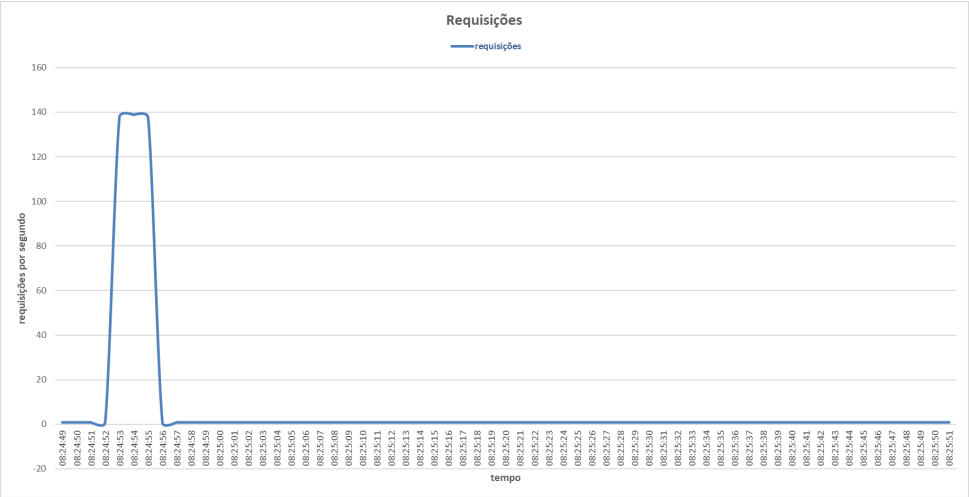
Figura 21 – Estados do serviço e das conexões durante o ataque *Bad* TCP *flags* (*All Flags Set*) com sensor *Random Forest*



Fonte: Autoria própria.

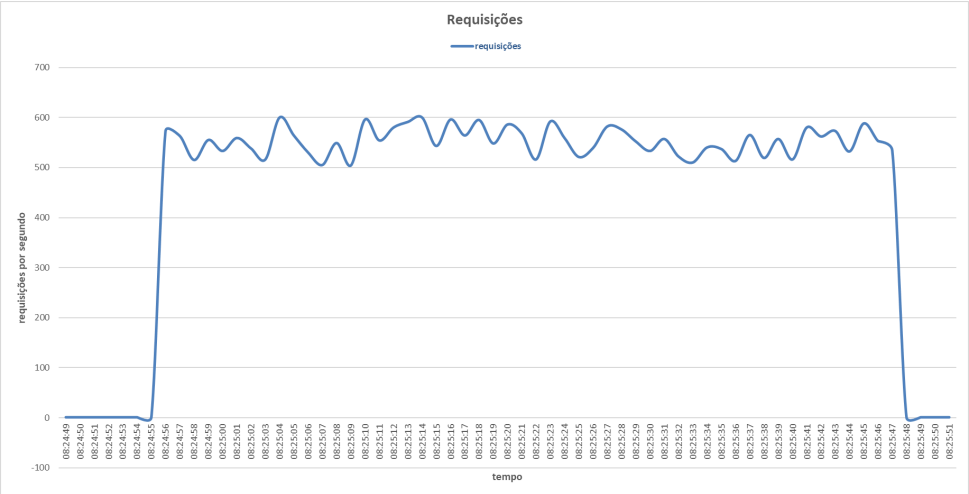
APÊNDICE B – Ataque FIN *Only Set* com sensor *Gaussian Naive Bayes*

Figura 22 – Requisições ao servidor primário durante ataque FIN *Only Set* com sensor *Gaussian Naive Bayes*



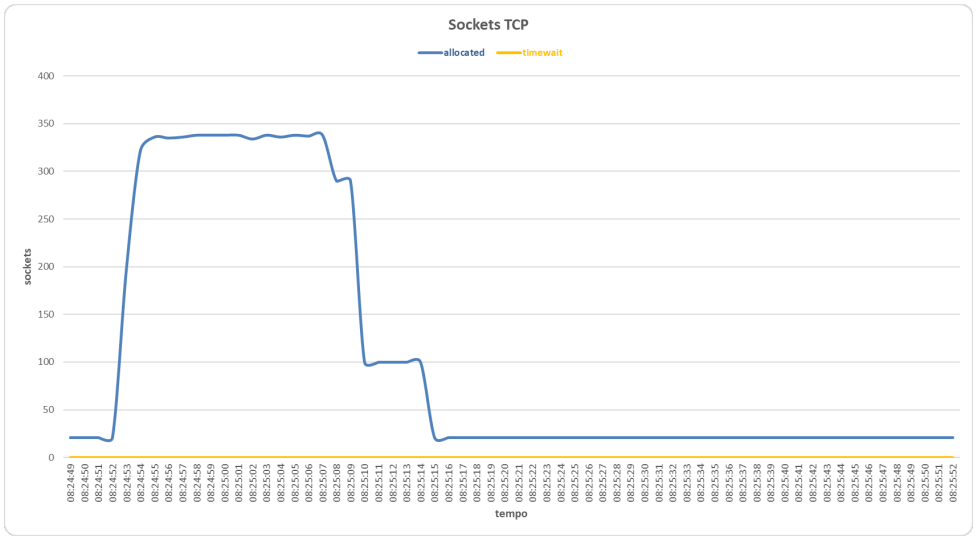
Fonte: Autoria própria.

Figura 23 – Requisições ao servidor secundário durante ataque FIN *Only Set* com sensor *Gaussian Naive Bayes*



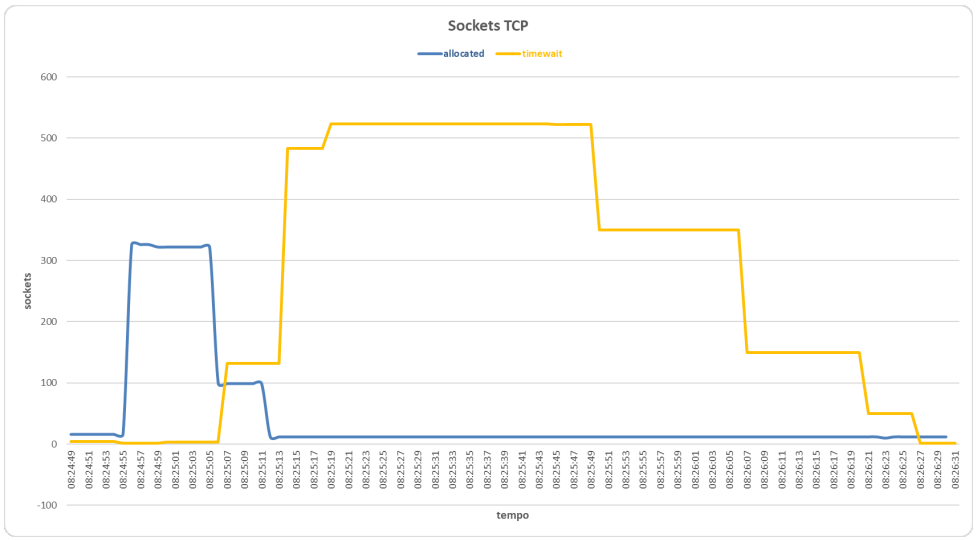
Fonte: Autoria própria.

Figura 24 – *Sockets* alocados e com *timewait* no servidor primário durante ataque FIN *Only Set* com sensor *Gaussian Naive Bayes*



Fonte: Autoria própria.

Figura 25 – *Sockets* alocados e com *timewait* no servidor secundário durante ataque FIN *Only Set* com sensor *Gaussian Naive Bayes*

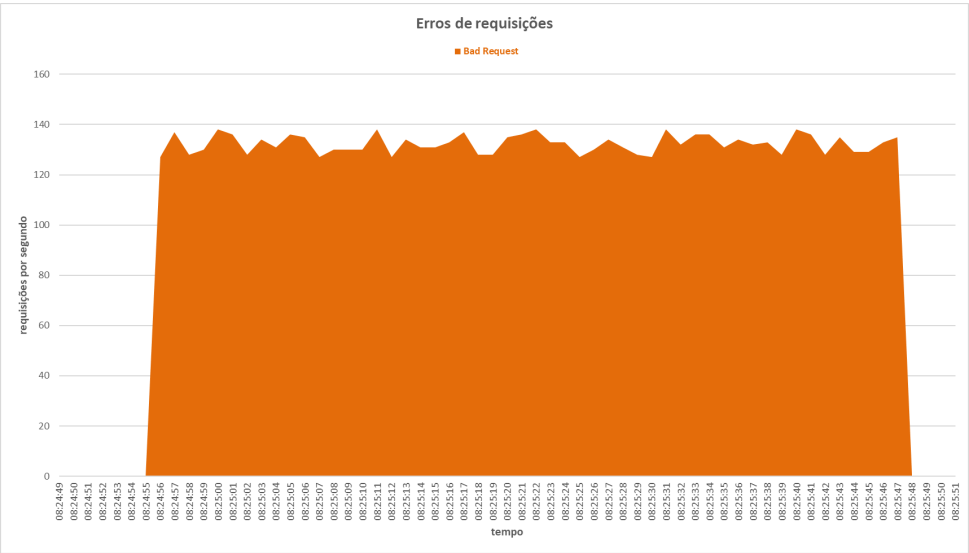


Fonte: Autoria própria.

Figura 26 – Erros de requisições ocorridos no servidor primário durante ataque FIN *Only Set* com sensor *Gaussian Naive Bayes*

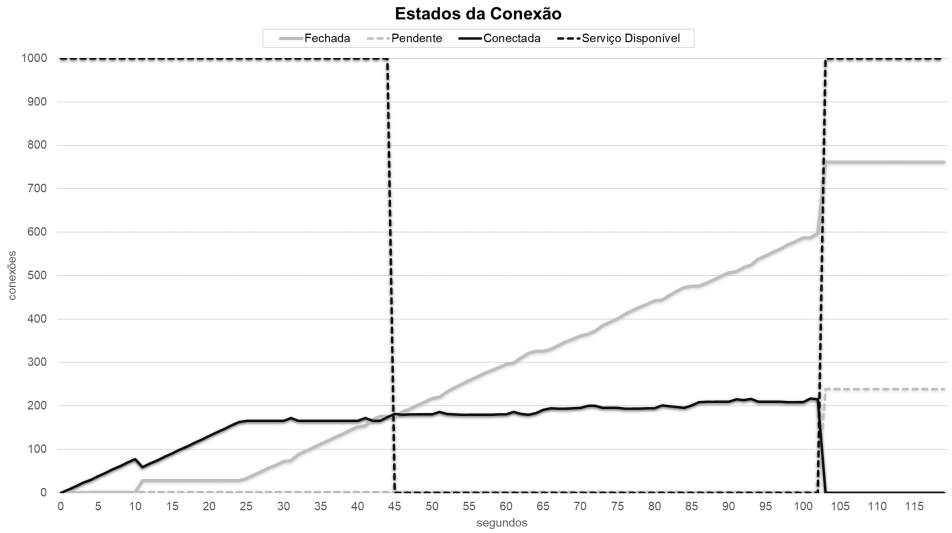
Fonte: Autoria própria.

Figura 27 – Erros de requisições ocorridos no servidor secundário durante ataque *FIN Only Set* com sensor *Gaussian Naive Bayes*



Fonte: Autoria própria.

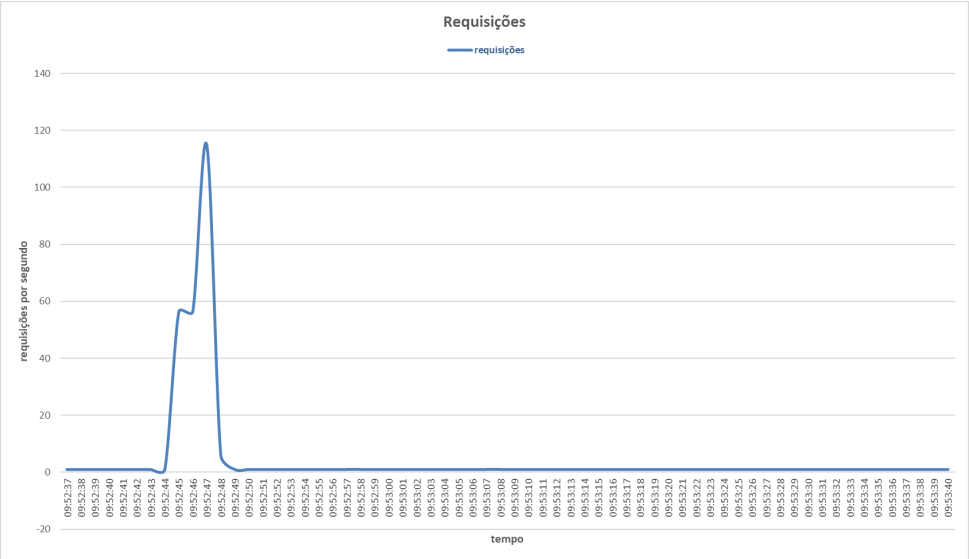
Figura 28 – Estados do serviço e das conexões durante o ataque *FIN Only Set* com sensor *Gaussian Naive Bayes*



Fonte: Autoria própria.

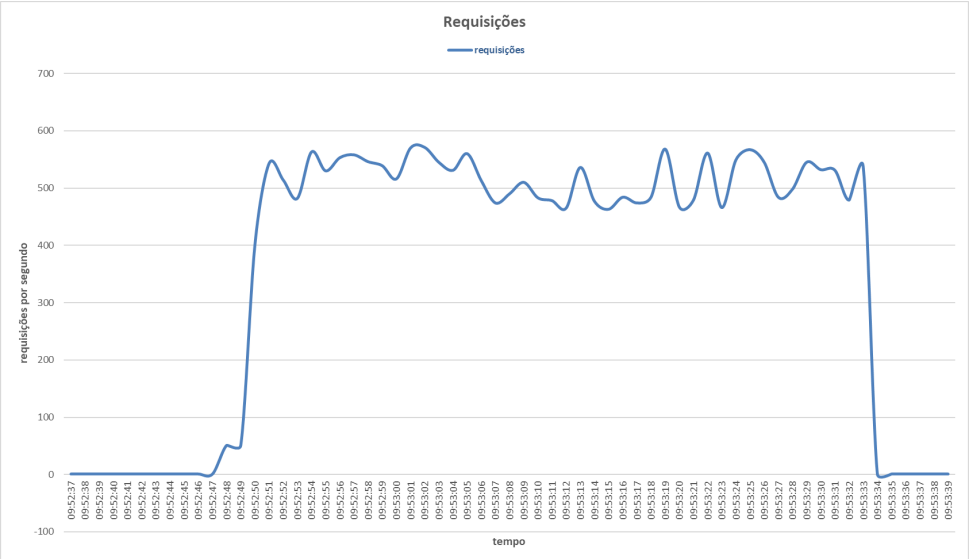
APÊNDICE C – Ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores

Figura 29 – Requisições ao servidor primário durante ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores



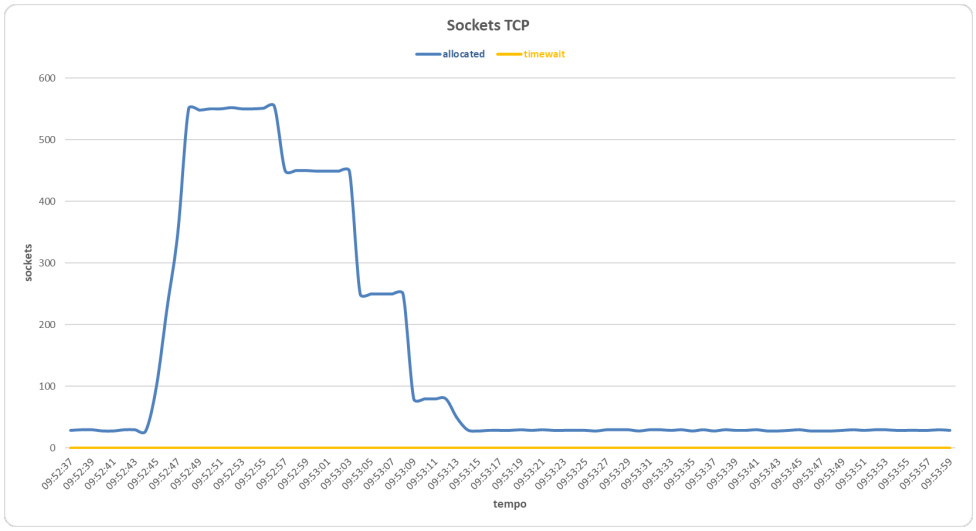
Fonte: Autoria própria.

Figura 30 – Requisições ao servidor secundário durante ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores



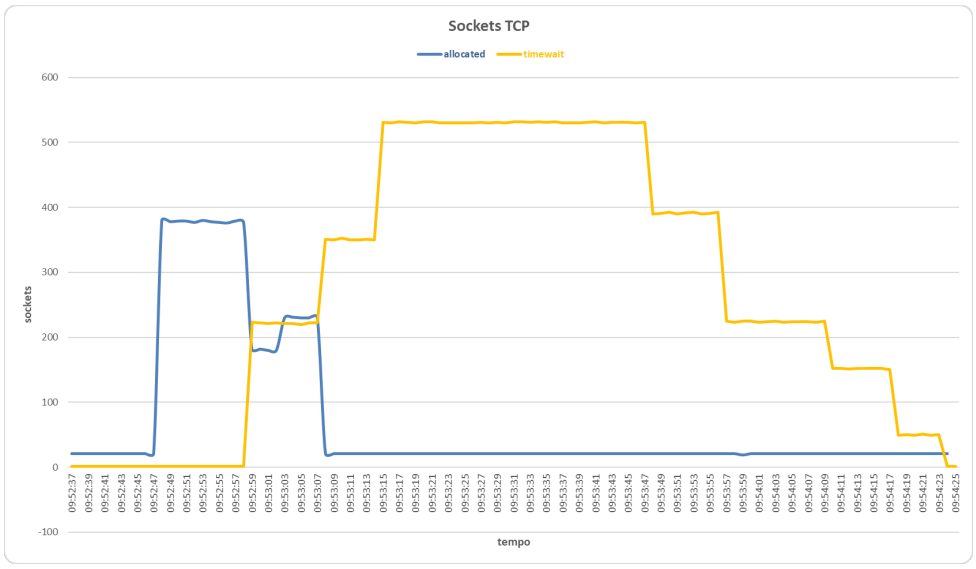
Fonte: Autoria própria.

Figura 31 – *Sockets* alocados e com *timewait* no servidor primário durante ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores



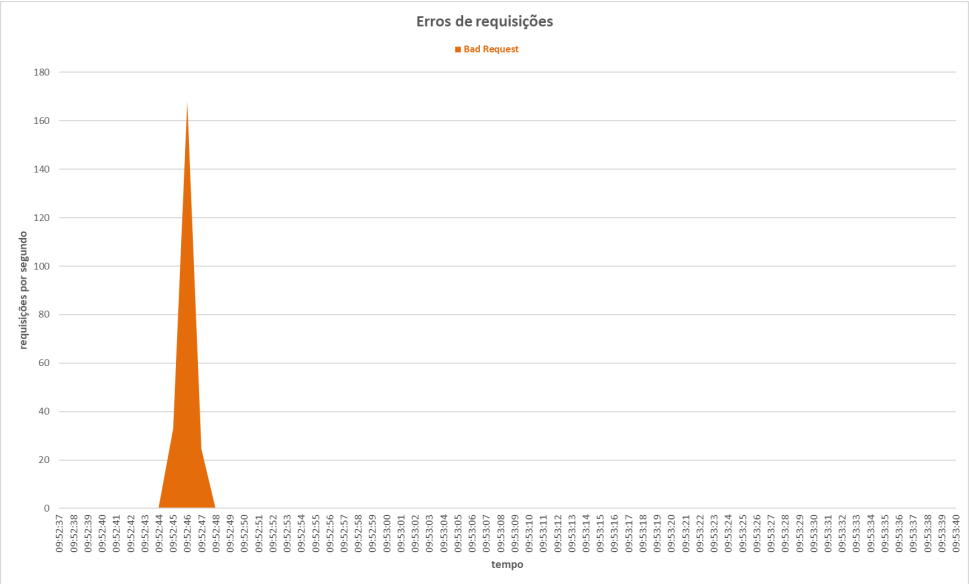
Fonte: Autoria própria.

Figura 32 – *Sockets* alocados e com *timewait* no servidor secundário durante ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores



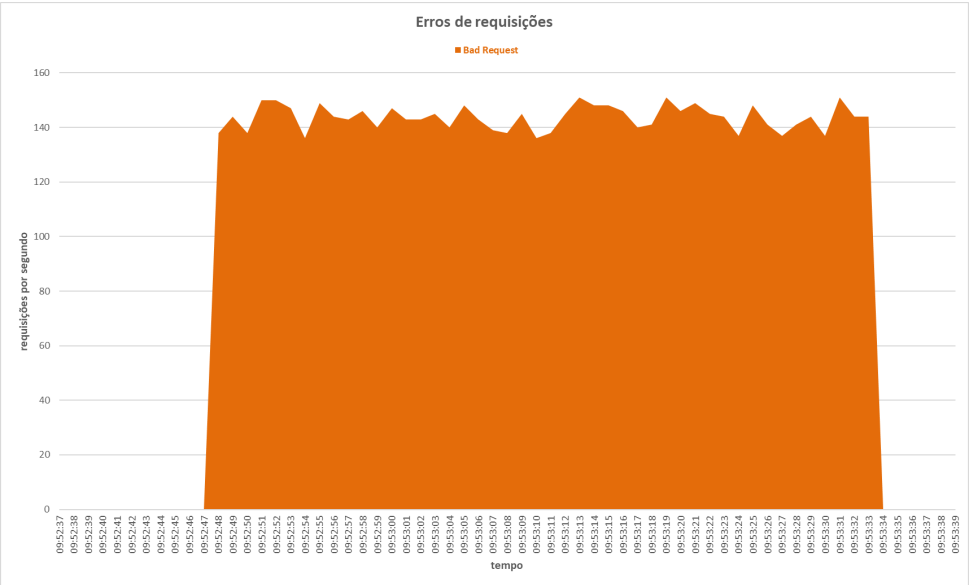
Fonte: Autoria própria.

Figura 33 – Erros de requisições ocorridos no servidor primário durante ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores



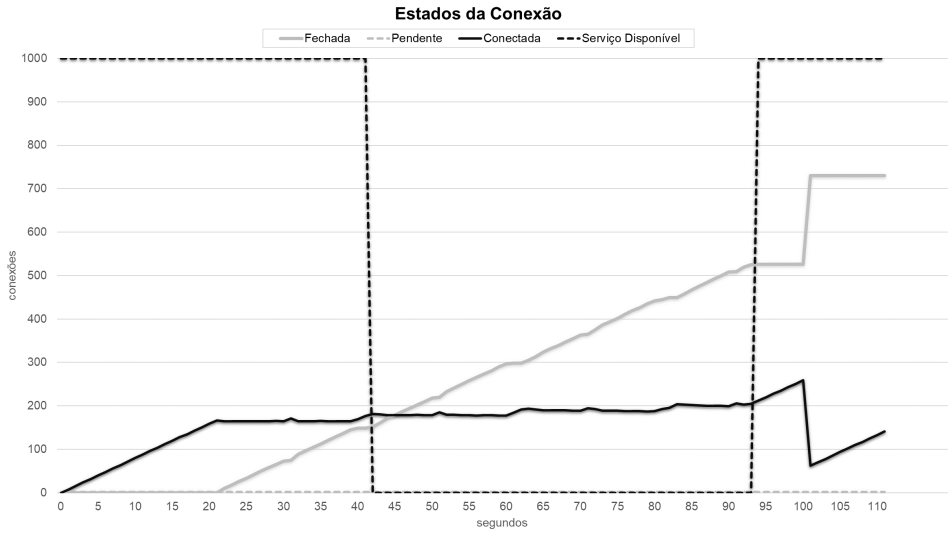
Fonte: Autoria própria.

Figura 34 – Erros de requisições ocorridos no servidor secundário durante ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores



Fonte: Autoria própria.

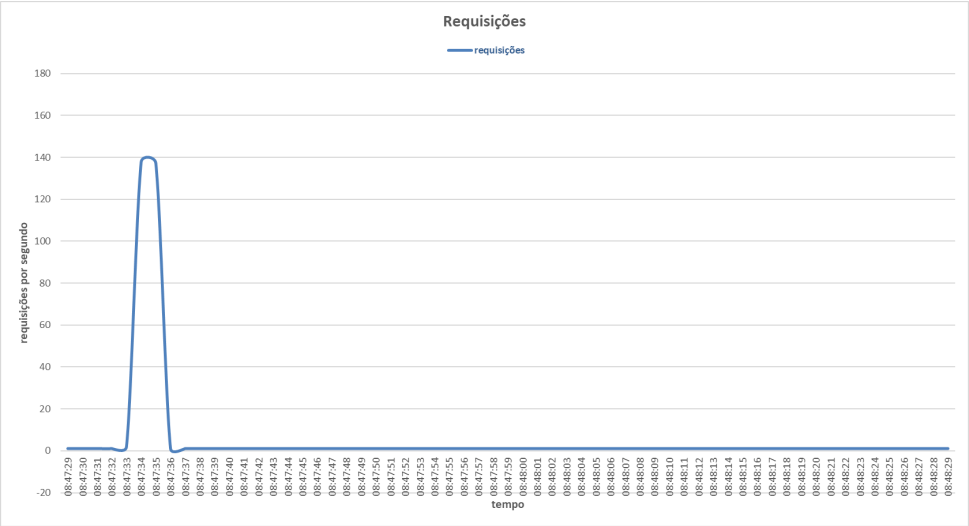
Figura 35 – Estados do serviço e das conexões durante o ataque *Slow* HTTP *POST* com sensor Pilha de Classificadores



Fonte: Autoria própria.

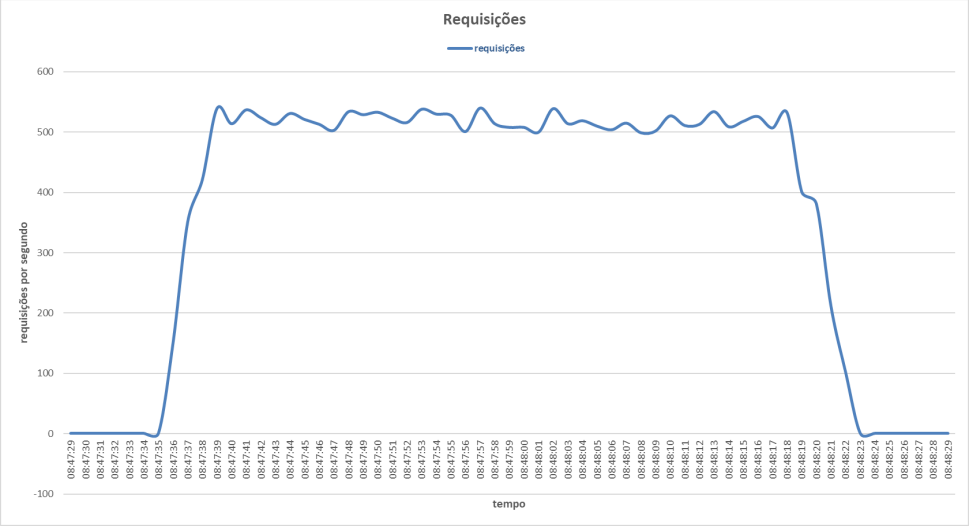
APÊNDICE D – Ataque SYN and FIN Set com sensor *Support Vector Machine*

Figura 36 – Requisições ao servidor primário durante ataque SYN and FIN Set com sensor *Support Vector Machine*



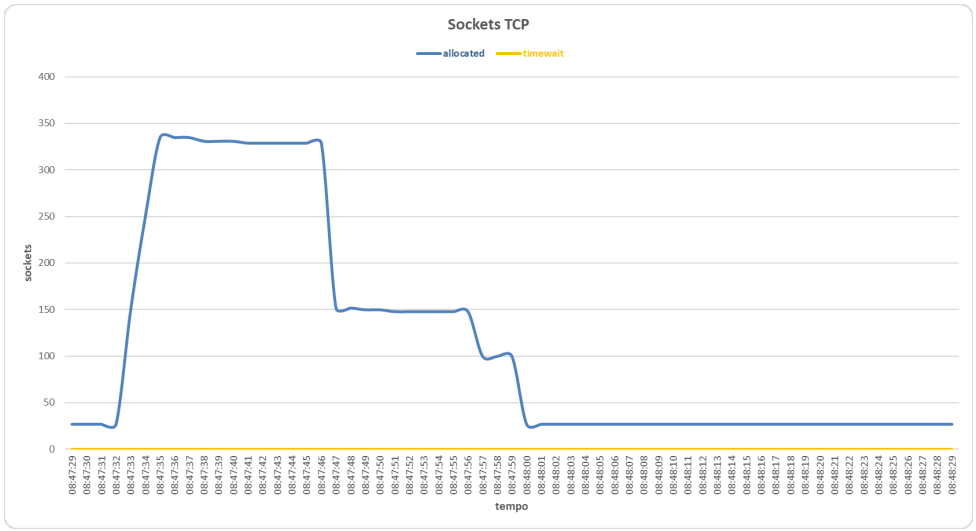
Fonte: Autoria própria.

Figura 37 – Requisições ao servidor secundário durante ataque SYN and FIN Set com sensor *Support Vector Machine*



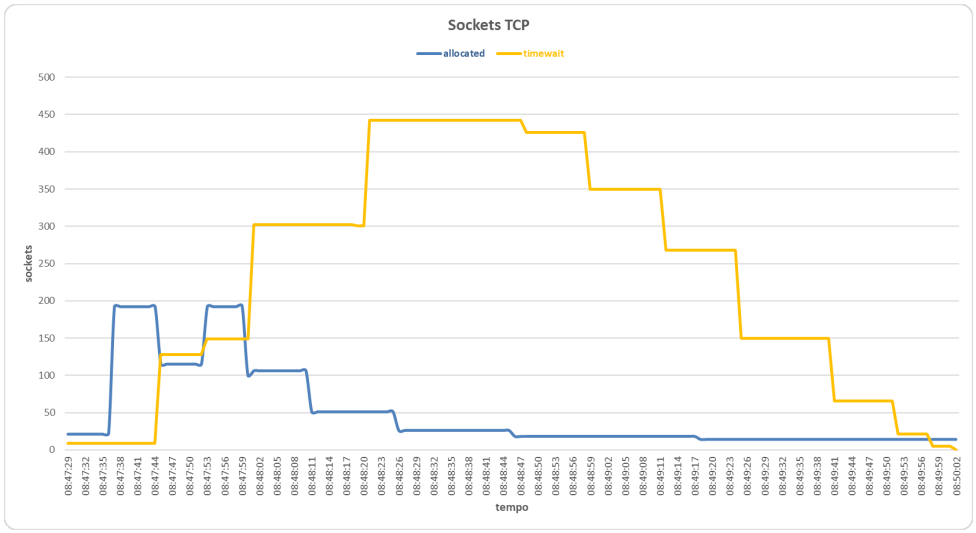
Fonte: Autoria própria.

Figura 38 – *Sockets* alocados e com *timewait* no servidor primário durante ataque SYN and FIN Set com sensor *Support Vector Machine*



Fonte: Autoria própria.

Figura 39 – *Sockets* alocados e com *timewait* no servidor secundário durante ataque SYN and FIN Set com sensor *Support Vector Machine*

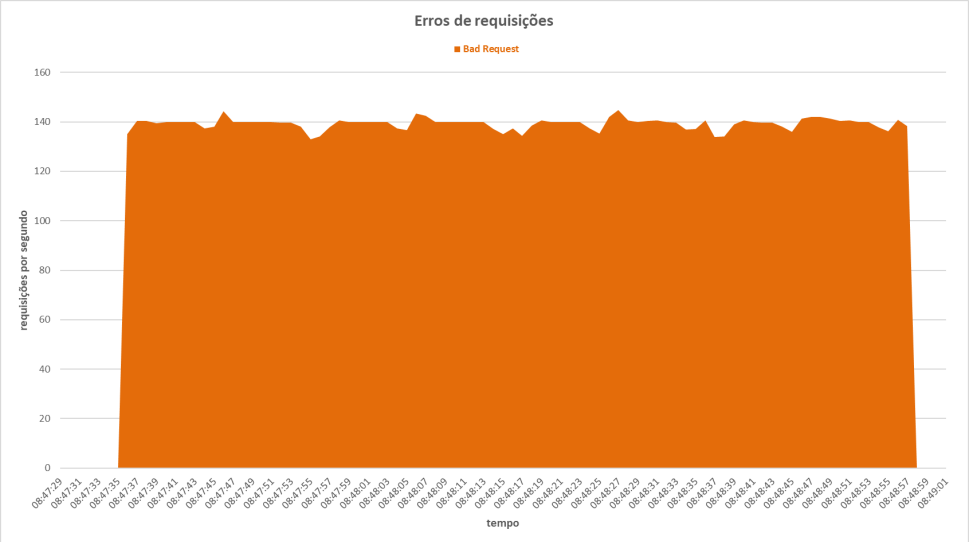


Fonte: Autoria própria.

Figura 40 – Erros de requisições ocorridos no servidor primário durante ataque SYN and FIN Set com sensor *Support Vector Machine*

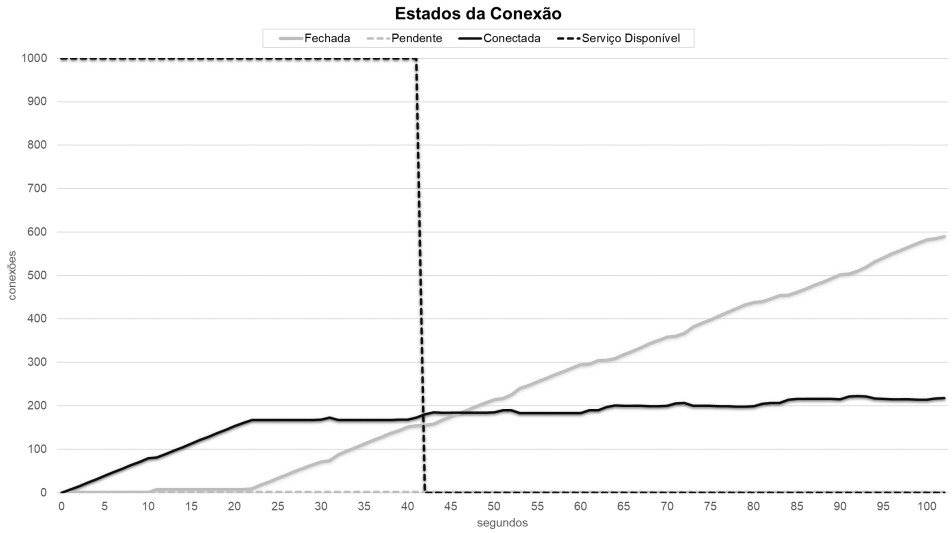
Fonte: Autoria própria.

Figura 41 – Erros de requisições ocorridos no servidor secundário durante ataque SYN and FIN Set com sensor *Support Vector Machine*



Fonte: Autoria própria.

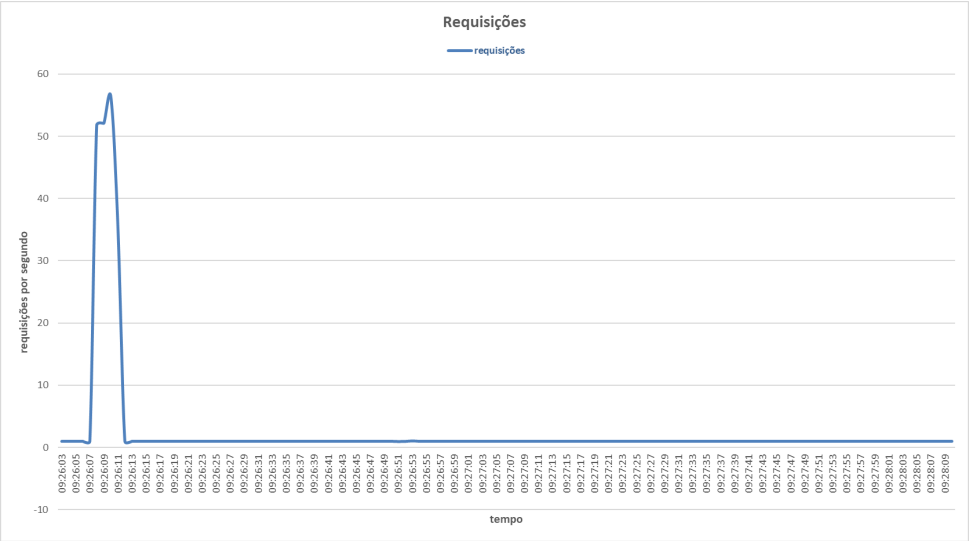
Figura 42 – Estados do serviço e das conexões durante o ataque SYN and FIN Set com sensor *Support Vector Machine*



Fonte: Autoria própria.

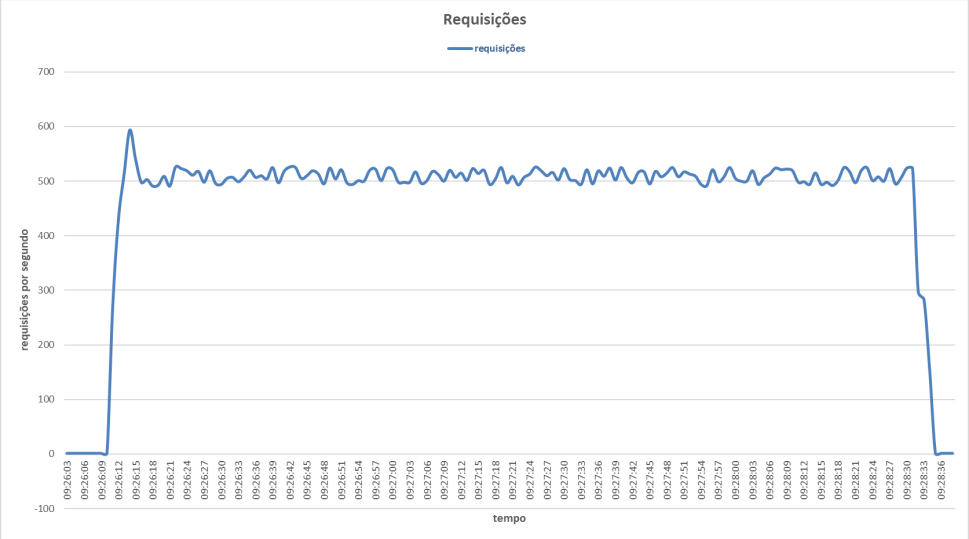
APÊNDICE E – Ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*

Figura 43 – Requisições ao servidor primário durante ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*



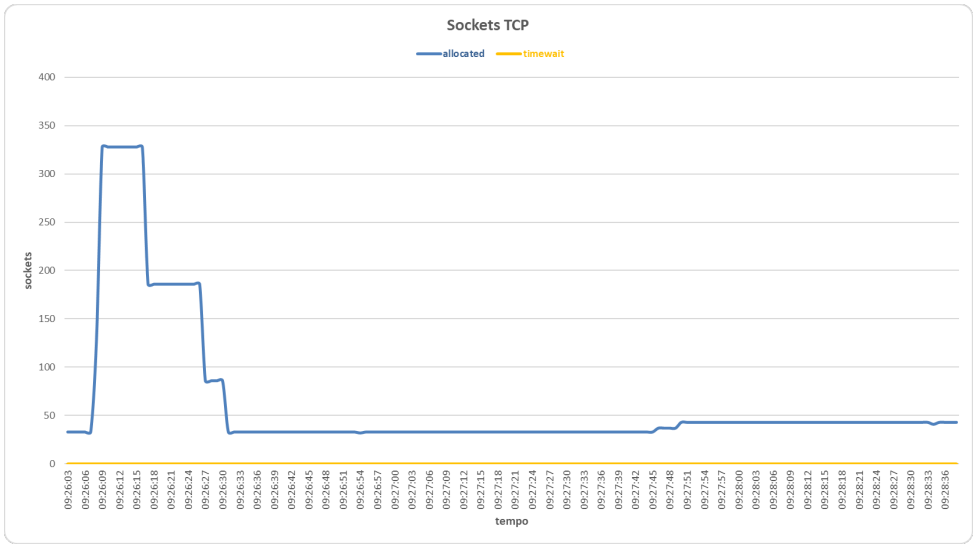
Fonte: Autoria própria.

Figura 44 – Requisições ao servidor secundário durante ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*



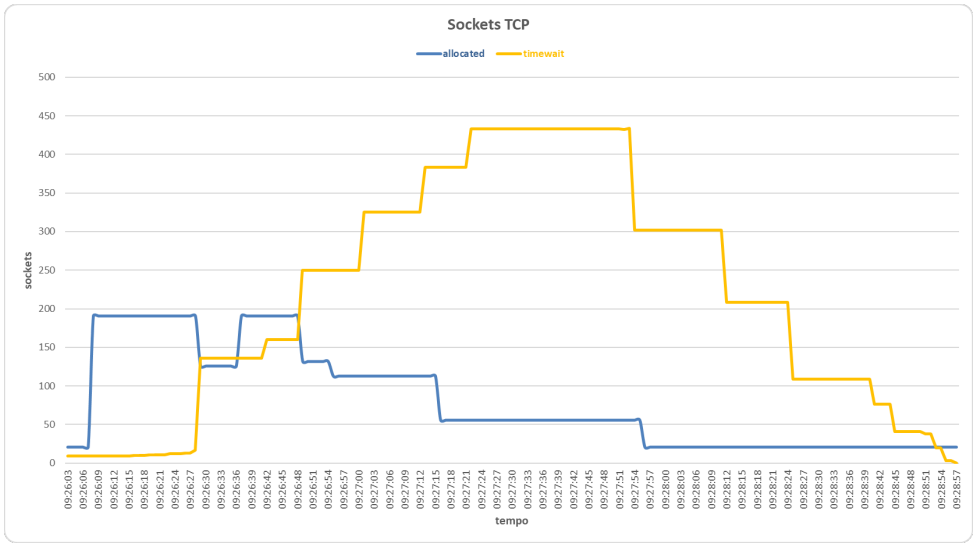
Fonte: Autoria própria.

Figura 45 – *Sockets* alocados e com *timewait* no servidor primário durante ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*



Fonte: Autoria própria.

Figura 46 – *Sockets* alocados e com *timewait* no servidor secundário durante ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*

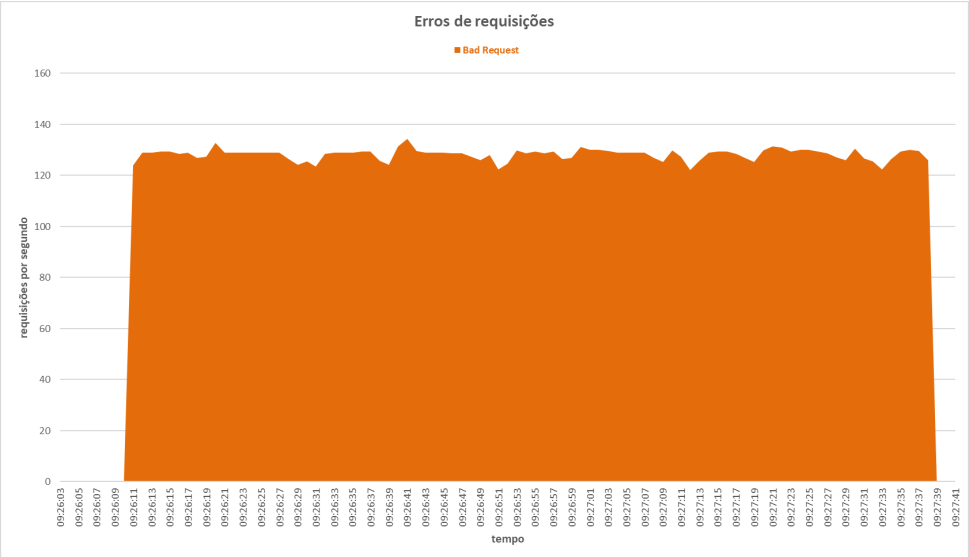


Fonte: Autoria própria.

Figura 47 – Erros de requisições ocorridos no servidor primário durante ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*

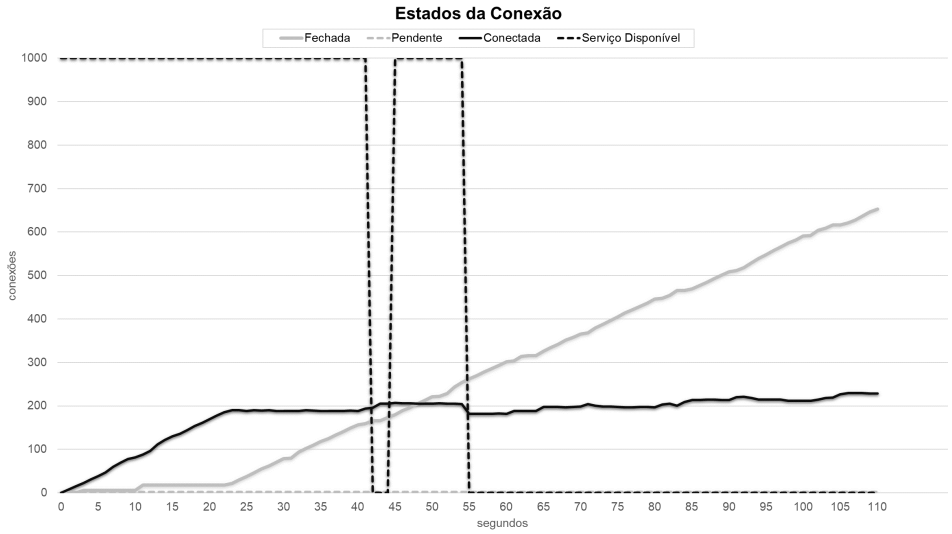
Fonte: Autoria própria.

Figura 48 – Erros de requisições ocorridos no servidor secundário durante ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*



Fonte: Autoria própria.

Figura 49 – Estados do serviço e das conexões durante o ataque TCP SYN *Flood* com sensor *Multilayer Perceptron*



Fonte: Autoria própria.