INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA CATARINENSE – CAMPUS CAMBORIÚ
CURSO DE SISTEMAS DE INFORMAÇÃO
(BACHARELADO)

FERNANDO LUIZ MORO

APRENDIZADO DE MÁQUINA NO PROCESSO DE DETECÇÃO DE ATAQUES EM REDE DEFINIDA POR SOFTWARE

CAMBORIÚ (SC)

2019

FERNANDO LUIZ MORO

APRENDIZADO DE MÁQUINA NO PROCESSO DE DETECÇÃO DE ATAQUES EM REDE DEFINIDA POR SOFTWARE

Trabalho de Conclusão de Curso submetido ao Instituto Federal Catarinense – Campus Camboriú, para obtenção dos créditos de disciplina com nome equivalente no curso de Sistemas de Informação – Bacharelado.

Orientação: Prof. Alexandre de Aguiar Amaral, Dr.

CAMBORIÚ (SC)

2019

FERNANDO LUIZ MORO

APRENDIZADO DE MÁQUINA NO PROCESSO DE DETECÇÃO DE ATAQUES EM REDE DEFINIDA POR SOFTWARE

ESTE RELATÓRIO, DO TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS DA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO, DO 8º. SEMESTRE, OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM SISTEMAS DE INFORMAÇÃO

Camboriú (SC), 11 de dezembro de 2019

Prof. Alexandre de Aguiar Amaral, Dr.
Orientador

BANCA EXAMINADORA:

Prof. Aujor Tadeu Cavalca Andrade, Dr.
IFC-CAMBORIÚ

Prof. Rodrigo de Paula e Silva Ribeiro, Bel
IFC-CAMBORIÚ

Prof. Daniel Fernando Anderle, Dr.
Professor de TCC

Prof. Daniel Fernando Anderle, Dr.
Coordenador de Curso

Dedico este trabalho para minha amada esposa Camila Pagelkopf Moro.

Agradeço a Deus, minha esposa, minha família, meu orientador, meus professores e meus amigos por todo o apoio, incentivo e paciência nesta jornada.

“Difícil de ver. Sempre em movimento está
o futuro.”

(Mestre Yoda)

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ACK - *Acknowledgment*
ACL - *Access Control List*
ACM - *Association for Computing Machinery*
API - *Application Programming Interface*
AUC - *Area Under the receiver operating characteristic Curve*
BPP - *Bytes per Packet*
BPS - *Byte per Second*
BYOD - *Bring Your Own Device*
BYT - *Byte*
CLI - *Command Line Interface*
CSS - *Cascading Style Sheets*
CSV - *Comma-Separated Values*
DA - *Destination Address*
DDoS - *Distributed Denial of Service*
DoS - *Denial of Service*
DP - *Destination Port*
DT - *Decision Tree*
FIN - *Finish*
FLG - *Flag*
FLW - *Flow*
GNB - *Gaussian Naive Bayes*
HTML - *HyperText Markup Language*
HTTP - *HyperText Transfer Protocol*
ICMP - *Internet Control Message Protocol*
IDS - *Intrusion Detection System*
IEEE - *Institute of Electrical and Electronics Engineers*
INIT - Initialize
IPS - *Intrusion Prevention System*
IoET - *Internet of Evil Things*
IoT - *Internet of Things*
IP - *Internet Protocol*
IPFIX - *IP Flow Information Export*
KNN - *K-Nearest Neighbors*
MAC - *Media Access Control*

MLP - *Multi-Layer Perceptron*
MVC - *Model-View-Controller*
NDC - *Network Dataset Creation*
NDP - *Number of Destination Port*
NSP - *Number of Source Port*
ORM - *Object-Relational Mapper*
PKT - *Packet*
PPS - *Packet per Second*
PR - *Protocol*
REST - *Representational State Transfer*
SA - *Source Address*
SCTP - *Stream Control Transmission Protocol*
SDN - *Software-Defined Networking*
SFEP - *Static Flow Entry Pusher*
SNMP - *Simple Network Management Protocol*
SP - *Source Port*
SVM - *Support Vector Machine*
SYN - *Synchronization*
TCP - *Transport Control Protocol*
TD - *Total duration*
TLS - *Transport Layer Security*
TTM - *Time to Market*
UDP - *User Datagram Protocol*

RESUMO

Devido ao aumento no número de ataques de rede lançados constantemente contra indivíduos e organizações, novos paradigmas têm sido explorados pelos sistemas de detecção de intrusão para combatê-los. A utilização do aprendizado de máquina neste cenário provê a habilidade de identificar automaticamente os padrões do tráfego de rede possibilitando a detecção de diferentes tipos de ataques. As atuais redes de computadores tornaram-se heterogêneas devido ao rápido crescimento e a conexão de novos dispositivos, aumentando ainda mais a complexidade nas operações manuais para realizar a mitigação dos ataques. Para prover uma maior flexibilidade e escalabilidade, surgiu a rede definida por software que proporciona uma comunicação homogênea com os dispositivos de rede, facilitando assim, a aplicação de ações de contramedida na ocorrência de ataques. A união destes dois paradigmas possibilita o desenvolvimento de uma solução mais inteligente, automatizada e apta a agir em um ambiente de rede real. Desta maneira, o objetivo deste trabalho é integrar o aprendizado de máquina e a rede definida por software para a detecção e mitigação de ataques de rede minimizando a intervenção humana neste processo. Foi desenvolvido um sistema web composto por dois componentes principais responsáveis por gerar a base de fluxos IP utilizada pelos modelos e por realizar de forma automatizada o monitoramento e combate aos ataques. A solução proposta foi validada em uma rede definida por software simulada onde foram executados diferentes experimentos com ataques de rede reais. Os resultados demonstraram que o sistema proposto neste trabalho foi capaz de detectar e mitigar automaticamente diferentes tipos de ataques.

Palavras-chave:
Segurança cibernética; Aprendizado de máquina; Rede definida por software.

ABSTRACT

Due to the increasing number of network attacks constantly launched against individuals and organizations, new paradigms have been exploited by intrusion detection systems to combat them. Using machine learning in this scenario provides the ability to automatically identify network traffic patterns enabling the detection of different types of attacks. Today's computer networks have become heterogeneous due to the rapid growth and connection of new devices, further increasing the complexity of manual operations to mitigate attacks. In order to provide greater flexibility and scalability, a software-defined networking has emerged that provides a homogeneous communication with the network devices, thus facilitating the application of countermeasure actions in the occurrence of attacks. Bringing these two paradigms together enables the development of a smarter, automated solution that can act in a real network environment. Thus, the objective of this paper is to integrate machine learning and software-defined networking for network attack detection and mitigation, minimizing human intervention in this process. A web system was developed consisting of two main components responsible for generating the IP flows base used by the models and for automatically monitoring and combating attacks. The proposed solution was validated in a simulated software-defined networking where different experiments with real network attacks were performed. The results showed that the system proposed in this work was able to automatically detect and mitigate different types of attacks.

Keywords:
Cybersecurity; Machine learning; Software-defined networking.

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

Cerca de 3.1 GB por minuto foram transmitidos pela Internet em 2018 e estima-se que até 2020, 1,7 MB de dados serão criados por segundo para cada pessoa na terra (AHMAD, 2018). A informação gerada por este grande volume de dados tornou-se um dos principais e mais valiosos ativos da era atual. Consequentemente, ela se tornou alvo de cibercriminosos que ameaçam sua disponibilidade, integridade e confidencialidade. Constantemente novos ataques são criados por "[...] indivíduos e organizações para atacar as redes de computadores com o objetivo de roubar as informações e dados privados" (NAJAFABADI *et al.*, 2015). Os ataques "[...] tendem a aumentar significativamente com a Internet das Coisas (*Internet of Things* - IoT), uma vez que estima-se mais de 80 bilhões de dispositivos interconectados até 2025" (LOBATO; LOPEZ; DUARTE, 2016 *apud* CLAY 2015).

Para se proteger dos inúmeros ataques de rede, medidas e sistemas provenientes da área de segurança cibernética têm sido propostos. Dentre as diversas tecnologias disponíveis surgiu o sistema de detecção de intrusão (*Intrusion Detection System* – IDS) que possui o objetivo de monitorar, detectar e analisar as atividades maliciosas na rede (CARVALHO, 2018). Levando em consideração a escalabilidade e a natureza da infraestrutura de rede, este tipo de mecanismo pode ter o desempenho afetado de acordo com a fonte de dados (*e.g. payload* de pacotes) escolhida. Desta maneira, "uma análise baseada em fluxos IP (*Internet Protocol*), que representa o resumo dos pacotes trafegados na rede, faz com que o volume de dados a serem

processados seja consideravelmente menor” (KAKIHATA *et al.*, 2017).

A detecção realizada por um IDS pode utilizar uma variedade de métodos, tais como entropia, qui-quadrado, análise de correlação, assinaturas estáticas, entre outros, sendo que nos últimos anos, soluções baseadas em aprendizado de máquina tem se popularizado (HAMID; SUGUMARAN; JOURNAUX, 2016). O aumento da utilização dos algoritmos de aprendizado de máquina nesse contexto é devido às limitações encontradas nas técnicas tradicionais, tais como firewall e mecanismos de controle de acesso, para detectar ataques mais sofisticados (NAJAFABADI *et al.*, 2015 *apud* ZAMANI e MOVAHEDI, 2013). Este paradigma utiliza uma abordagem automatizada para a aquisição do conhecimento de fontes não humanas como textos, planilhas, base de dados, entre outras (ABEL; FIORINI, 2013).

Com o crescimento da Internet, novos dispositivos de diferentes fabricantes com sistemas operacionais e interfaces proprietários foram incorporados na rede, tornando-a heterogênea (PANDIKUMAR; ATKILT; HASSEN, 2017). Como resultado, o gerenciamento da rede foi dificultado tornando a detecção e mitigação de ataques ações custosas. Com o surgimento da rede definida por software (*Software-Defined Networking* – SDN) esta questão foi contornada, possibilitando a comunicação homogênea entre um IDS e os dispositivos de rede para a aplicação de ações de contramedida a um ataque detectado. Este procedimento é possível porque na arquitetura SDN os planos de controle e encaminhamento são separados permitindo que o controle da rede seja gerenciado por um elemento centralizado (MOUSAVI, 2014).

Na literatura diferentes técnicas para detectar a ocorrência de ataques são empregadas em um ambiente SDN, tais como em Pandikumar *et al.* (2017), Singh *et al.* (2015) e Wang *et al.* (2015). Embora tais técnicas realizem a detecção baseadas em limiares, a cada mudança no ambiente monitorado haverá a intervenção humana para redefinir os valores deles. Autores como Belouch *et al.* (2018), Hamid *et al.* (2016) e Najafabadi *et al*. (2015) buscam automatizar o processo de detecção de um IDS através do aprendizado de máquina. Em casos como Najafabadi *et al.* (2015), um método de agregação dos fluxos de rede é aplicado com o objetivo de gerar

características mais discriminativas para melhorar o aprendizado dos modelos. Contudo, não é realizada uma comparação com os fluxos sem agregação para avaliar os benefícios do procedimento realizado. Bhunia e Gurusamy (2017), Meti *et al.* (2017) e Mohammed *et al.* (2018) integram o aprendizado de máquina com SDN por meio de mecanismos automatizados, porém, não exploram o desenvolvimento de uma ferramenta web para facilitar os procedimentos realizados pelo administrador de rede.

Propõem-se neste trabalho de conclusão de curso a aplicação do aprendizado de máquina supervisionado para a detecção de ataques no contexto de uma rede definida por software. Buscou-se reduzir a intervenção humana neste processo para que os ataques de rede fossem detectados antecipadamente e mitigados através da execução de ações automatizadas por meio do controlador SDN. O objetivo destes procedimentos consiste em minimizar a indisponibilidade da infraestrutura de rede e dos serviços prestados por ela durante a ocorrência de um ataque. Uma base de dados com os fluxos IP agregados foi gerada para que os algoritmos de aprendizado de máquina obtivessem um melhor desempenho no tempo consumido durante o treinamento e nas classificações realizadas. Com o intuito de permitir a integração desses diferentes paradigmas foi desenvolvido um sistema web que possibilita o administrador de rede criar bases de fluxos IP e realizar a segurança em tempo real do ambiente de rede monitorado. Para validar a proposta foram realizados diferentes experimentos com ataques reais em uma topologia SDN virtual.

1.2 Descrição do problema

O número de pessoas utilizando a Internet aumentou consideravelmente nos últimos anos atingido 4.388 bilhões de pessoas conectadas das 7.676 bilhões existentes no mundo (KEMP, 2019). A facilidade de acesso às novas tecnologias como os dispositivos IoT tem contribuído com o aumento do número de usuários online. A IoT tem se popularizado cada vez mais na vida cotidiana da população conectando “[...] *wearables*, câmeras, luminárias, eletrodomésticos e outros dispositivos inteligentes na Internet formando um ecossistema rico [...]” e diversificado (BHUNIA; GURUSAMY,

2017). Segundo Gartner (2018) “14.2 bilhões de coisas conectadas estarão em uso em 2019, e que chegará a 25 bilhões até 2021”.

“Somado a este crescimento em larga escala das redes, está o surgimento constante de novas vulnerabilidades resultantes da pressão do mercado para o lançamento cada vez mais rápido (*Time to Market* - TTM) [...]” de novas tecnologias (AMARAL, 2015). Devido a este fator, novas brechas de segurança são descobertas e exploradas por cibercriminosos munidos com um arsenal de ferramentas com o objetivo de executar uma série de ataques para roubar as informações digitais. Em 2018, uma onda “[...] de ataques cibernéticos e violações de dados atingiram quase todos os setores da indústria [...]”, sendo estes, executados por dispositivos IoT (SARANG, 2018). Um dos fatores preocupantes é que muitos deles são comprometidos para formarem grandes *botnets* (rede de computadores infectados) com o objetivo de executar ataques DDoS (*Distributed Denial of Service*). Este tipo de ataque “[...] cresceu 29% desde o segundo trimestre de 2017, com o tamanho médio dos ataques aumentando cerca de 543%” (ABRAMS, 2018).

Além disso, “novas tendências como BYOD (*Bring Your Own Device*) utilizadas pelas corporações [...]”, têm contribuindo para o agravo da segurança da infraestrutura de rede (AMARAL, 2015). Especialistas têm afirmado que chegou-se a um estado incontrolável no que tange o gerenciamento das redes levando o surgimento de uma nova terminologia para a Internet, a IoET, um acrônimo da *Internet of Evil Things* (PWNIE EXPRESS, 2017). “O fato é que as redes de computadores se tornaram alvo de inúmeros ataques cada vez mais sofisticados, ao passo que mecanismos de defesa tradicionais não têm sido suficientes para acompanhar essa evolução [...]”, pois dependem da intervenção humana no processo de diagnóstico e solução dos ataques de rede (AMARAL, 2015).

Outro fator crítico está relacionado com a complexidade do gerenciamento da rede devido a heterogeneidade da infraestrutura atual. Segundo Costa (2013) às redes de computadores possuem dispositivos e softwares de inúmeros fabricantes denominados “[...] caixas pretas, ou seja, implementações integradas baseadas em software e hardware proprietário”. Tais fatores tornam ainda mais complexa a

aplicação de ações automatizadas para bloquear os ataques requerendo do administrador de rede intervenções manuais que impactam no tempo de resposta para solucioná-los. Assim, “[...] as mesmas razões que permitiram a evolução da Internet são vistas como as barreiras para o seu desenvolvimento, ameaçando a sua capacidade de suprir as demandas futuras da sociedade” (COSTA, 2013).

1.3 Justificativa

Mediante a problemática apresentada o desenvolvimento de um sistema de detecção de intrusão automatizado baseado no aprendizado de máquina “[...] pode ajudar os defensores a superar determinadas lacunas, tornando-os mais eficazes na identificação e resposta às ameaças conhecidas e emergentes” (CISCO, 2018). O aprendizado de máquina possibilita identificar automaticamente os padrões do tráfego de rede distinguindo os fluxos que são normais dos anômalos possibilitando a detecção dos ataques. Assim, a intervenção humana é reduzida resultando em uma maior agilidade no processo de detecção, pois todo o aprendizado será realizado por algoritmos especializados que necessitam ser configurados apenas antes do treinamento.

A rede definida por software é utilizada neste trabalho pois permite a aplicação automatizada de regras de bloqueio para interromper os ataques detectados, fator este, que é complexo de ser realizado nas atuais redes de computadores devido a sua estrutura rígida. “A arquitetura SDN surge como uma alternativa promissora permitindo a flexibilidade e escalabilidade sem precedentes, tanto na configuração quanto na implantação de serviços” (CARVALHO, 2018). Desta maneira, ao integrar o aprendizado de máquina e a SDN é possível o desenvolvimento de uma solução mais inteligente, automatizada e apta a agir em um ambiente de rede real onde inúmeros ataques cada vez mais complexos têm sido criados.

Para facilitar o gerenciamento do administrador de rede e possibilitar a integração dos paradigmas apresentados, outra contribuição deste trabalho é um sistema web composto por dois componentes principais. A ideia é que através deste

sistema a base de fluxos IP possa ser criada para o aprendizado dos algoritmos para que com a formação dos modelos, o sistema realize a detecção em tempo real e aplique ações de contramedidas por meio do controlador SDN. Todo este procedimento não requer a intervenção do administrador de rede permitindo que ele através da interface web obtenha uma visão holística do estado de segurança da rede sendo informado sobre os ataques detectados.

1.4 Objetivo geral

Aplicar o aprendizado de máquina supervisionado no processo de detecção de ataques no contexto de uma rede definida por software buscando a minimização da intervenção humana.

1.5 Objetivos específicos

- Integrar o aprendizado de máquina com a rede definida por software através de um sistema web que possibilite a detecção e mitigação automática dos ataques de rede.
- Identificar os algoritmos de aprendizado de máquina e os procedimentos mais adequados para a tarefa de detecção de ataques.
- Criar uma base de fluxos IP agregada para melhorar o aprendizado dos modelos e colaborar com o tempo de processamento.
- Simular uma topologia de rede definida por software virtual para a realização dos experimentos com os ataques de rede.

1.6 Metodologia

Este trabalho caracteriza-se quanto a natureza de pesquisa como aplicada tecnologicamente e as etapas para o desenvolvimento dele podem ser observadas na

Figura 1.

FIGURA 1 - Procedimentos metodológicos aplicados.
FONTE: Autor (2019).

Na primeira etapa foi realizada uma pesquisa exploratória através do levantamento bibliográfico sobre o tema e os trabalhos relacionados nas principais bases científicas especializadas como IEEE (*Institute of Electrical and Electronics Engineers*), ACM (*Association for Computing Machinery*) e Elsevier. O enfoque principal da pesquisa foi no aprendizado de máquina e rede definida por software. Assim, foram utilizados os termos "*machine learning and software-defined networking*", "*machine learning and intrusion detection systems*", "*machine learning and cyber security*" e "*anomaly detection and software-defined networking*" nas buscas. Com o referencial bibliográfico selecionado estabeleceu-se critérios para filtrar as publicações como documentos escritos em língua portuguesa ou inglesa, remoção de itens duplicados e leitura do resumo para avaliar se de fato o artigo estava relacionado com a proposta deste trabalho.

A etapa 2 iniciou-se através da leitura do material bibliográfico selecionado para que fosse possível criar o entendimento sobre os assuntos relacionados a este trabalho. Esta etapa é crucial, pois a base teórica é o fundamento para que as próximas etapas pudessem ser realizadas com sucesso. Com o que foi compreendido desta etapa, foram desenvolvidos artigos científicos que ajudaram na escrita dos capítulos deste trabalho, com destaque, o capítulo de fundamentação teórica.

O primeiro passo na terceira etapa foi identificar as potenciais tecnologias para a coleta dos dados de rede, implementação dos algoritmos de aprendizado de máquina e simulação de uma topologia de rede definida por software. Através da leitura de diferentes artigos científicos foi observado que muitos autores utilizavam um conjunto específico de ferramentas para a coleta e monitoramento dos dados de rede, sendo assim, procurou-se seguir a mesma direção. Nos estudos realizados foram

identificadas algumas linguagens de programação que se destacaram por possuírem bibliotecas específicas para a área de aprendizado de máquina. Foram realizados testes para cada uma delas para avaliar qual seria a mais adequada para alcançar o objetivo deste trabalho.

Conforme a pesquisa foi avançando compreendeu-se que seria necessário integrar o aprendizado de máquina e a rede definida por software através do desenvolvimento de um sistema web. Foi então definida uma arquitetura de software e o sistema foi dividido em dois componentes principais. O primeiro componente foi projetado com o objetivo de formar a base de fluxos IP com tráfego de rede atualizado e ataques de rede reais. Posteriormente, foi implementado um método para agregar os fluxos IP, pois foi percebido que este procedimento colaboraria com o aprendizado e tempo de modelagem dos algoritmos. O outro componente considerado o principal, possibilitou a integração dos paradigmas para que a proposta deste trabalho fosse alcançada.

Após o desenvolvimento do sistema iniciou-se a etapa dos experimentos com os algoritmos de aprendizado de máquina que foram treinados e testados na base de fluxos IP agregada. Os modelos gerados foram avaliados através das métricas comumente utilizadas na literatura para que fosse possível escolher o que obtivesse o melhor desempenho. Outro aspecto diz respeito a escolha do ambiente de rede utilizado para realização dos experimentos. Devido a ausência de uma infraestrutura SDN física no campus, foi utilizado um simulador virtual para desenvolver a topologia necessária para os testes. Fluxos concorrentes com ataques de rede sendo executados em paralelo nesta topologia serviram para avaliar a capacidade de detecção e mitigação da solução. Através destes procedimentos foram coletados dados quantitativos para o desenvolvimento dos resultados deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Definições e classificações de anomalias

Ataques de rede são ações que exploram as vulnerabilidades encontradas em um dispositivo com o objetivo de “[...] invadir um sistema, acessar informações confidenciais, disparar ataques contra outros computadores ou tornar um serviço inacessível” (CERT.BR, 2012). Diferentes técnicas são utilizadas pelos atacantes, tais como, falsificação de e-mail, interceptação de tráfego, desconfiguração de página, ataques de força bruta, ataques de varredura de rede e ataques de negação de serviço. Determinados tipos de ataques como força bruta, varredura de rede e negação de serviço podem alterar o comportamento de uma rede gerando eventos anormais que são conhecidos como anomalias (AMARAL, 2015). Contudo, uma anomalia não é gerada somente por atos maliciosos, mas também, por comportamentos operacionais e não usuais.

As “[...] anomalias operacionais surgem por meio da má configuração de [...]” serviços, dispositivos de rede e pelas falhas de equipamentos causando “[...] mudanças súbitas nos atributos que mensuram o volume do tráfego transmitido, por exemplo, bits por segundo” (CARVALHO, 2018 *apud* BARFORD e PLONKA 2001). O comportamento não-usual é “[...] causado por usuários legítimos, que não têm a intenção de afetar a operação normal ou comprometer a segurança da rede” (AMARAL, 2015). Embora estes dois tipos de comportamento devem ser monitorados e tratados eles não serão abordados no escopo deste trabalho. As anomalias maliciosas possuem características prejudiciais destinadas a comprometer a confidencialidade, integridade e disponibilidade da rede ou informação trafegada nela (AHMED; NASER

MAHMOOD; HU, 2016). Comumente, este tipo de comportamento tem o "[...] objetivo de sondar ou exaurir os recursos de um dado dispositivo e/ou uma rede" (AMARAL, 2015).

A sondagem (*probe*) realiza "[...] buscas minuciosas em redes, com o objetivo de identificar computadores ativos e coletar informações de serviços disponibilizados e programas instalados" (CERT.BR, 2012). Os principais métodos utilizados para esta tarefa são o *port scan* que identifica as portas abertas de um dispositivo e o *network scan* que descobre "[...] os dispositivos que estão ativos na rede, coletando suas informações, tais como sistema operacional e os serviços oferecidos por eles" (CARVALHO, 2018). Com um funcionamento semelhante, o código malicioso denominado *worm* se auto replica na rede buscando por vulnerabilidades nas portas lógicas. Cibercriminosos usam estas técnicas para o reconhecimento de um ambiente de rede para posteriormente lançar os ataques explorando as brechas de segurança encontradas.

Ataques de negação de serviço são eventos altamente prejudiciais, pois possuem o propósito de esgotar os recursos da infraestrutura de rede causando a interrupção do ambiente computacional e impedindo que os usuários legítimos tenham acesso ao serviço provido (AHMED; NASER MAHMOOD; HU, 2016). Tal categoria possui duas vertentes diferindo-se na quantidade de dispositivos atacantes envolvidos. Um ataque de negação de serviço (*Denial of Service* - DoS) ocorre quando um grande número de requisições são enviadas de um único dispositivo para o alvo, enquanto que um ataque de negação de serviço distribuído, múltiplos nós são responsáveis pelo envio das requisições. Uma das estratégias utilizadas pelos atacantes consiste em infectar através de códigos maliciosos as máquinas conectadas na rede obtendo o controle delas para executar este tipo de ataque.

Ataques de negação de serviço possuem variações como UDP (*User Datagram Protocol*) *flooding*, SYN *flooding*, ICMP (*Internet Control Message Protocol*) *flooding*, entre outros (MOUSAVI, 2014). O termo *flooding* é utilizado por causa da principal característica desta anomalia que consiste em enviar um grande número de requisições causando uma inundação nos recursos da rede. Em linhas gerais cada uma das variações explorará uma característica do protocolo de rede utilizado

fazendo com que o receptor aloque os recursos computacionais até que haja o esgotamento deles. Como exemplo, o ataque DoS SYN *flooding* se aproveita do método de estabelecimento de conexões denominado 3-*way handshake* (aperto de mão de três vias) do protocolo TCP (*Transport Control Protocol*). Inicialmente o emissor envia um pacote com a *flag* SYN sinalizada para o receptor que por sua vez, responde com outro pacote sinalizando as *flags* SYN/ACK e aguarda a resposta do emissor. O ponto explorado por este ataque consiste em não enviar a resposta final com a *flag* ACK fazendo com que o receptor aloque os recursos e fique aguardando a resposta que não chegará. “Quando inúmeras conexões incompletas são realizadas simultaneamente, o servidor começa a ser sobrecarregado, dada a quantidade finita de recursos” (AMARAL, 2015).

2.2 Fonte de dados

Para que uma anomalia de rede possa ser detectada é preciso uma fonte de dados adequada que possibilite identificar as discrepâncias que sinalize a ocorrência de comportamentos anormais. Além disso, a origem dos dados pode determinar quais anomalias serão detectadas e também é um fator essencial para a implantação do sistema em uma rede de grande escala (AMARAL *et al.*, 2017). Diferentes tipos de fontes podem ser utilizados para a coleta de dados de rede, como as informações de pacotes (cabeçalho e *payload*), SNMP (*Simple Network Management Protocol*) e fluxos de rede obtidos via sFlow, NetFlow e IPFIX (*IP Flow Information Export*) (HOFSTEDE *et al.*, 2014). Ambas as fontes de dados citadas fazem parte do monitoramento passivo que consiste na contabilização do tráfego de rede extraído de um elemento monitorado (*e.g.* roteador).

A abordagem por pacotes consiste na coleta de “[...] pacotes de rede enviados e recebidos na interface física de um computador capturados por uma API (*Application Programming Interface*) específica chamada pcap” que é utilizada por programas como Wireshark e TCPDUMP (BUCZAK; GUVEN, 2016). Os dados gerados por tal abordagem permitem uma análise mais detalhada sobre o tráfego de rede, contudo, em redes de alta velocidade é requerida uma capacidade de hardware

substancial para armazenar e analisar tais dados (HOFSTEDE *et al.*, 2014). Como alternativa, os fluxos de rede IP coletados de roteadores e switches podem minimizar o volume de dados processados e armazenados (AMARAL, 2015).

"Um fluxo é um conjunto de pacotes passando por um ponto de observação na rede durante um determinado intervalo de tempo" compartilhando propriedades em comum como endereço de IP de origem e destino, porta de origem e destino e protocolo (TRAMMELL; CLAISE, 2013). O processo de criação e manipulação do tráfego de rede antes dos fluxos serem devidamente armazenados segue as etapas de observação, medição e exportação (CARVALHO, 2018). De modo geral, um dispositivo de rede (*flow exporter*) que suporta o protocolo de fluxos coletará, analisará e contabilizará os pacotes trafegados na rede para a criação dos fluxos. Posteriormente, os fluxos serão exportados para uma máquina coletora (*flow collector*) cuja a responsabilidade é armazená-los para a utilização de ferramentas especializadas de análise.

2.3 Detecção de anomalias

"Segurança cibernética é um conjunto de tecnologias e processos projetados para proteger computadores, redes, programas e dados de ataques e acesso não autorizado [...]" (XIN *et al.*, 2018). Tipicamente, a segurança de um ambiente de rede é formada pela integração de tecnologias como *firewall*, antivírus, *antimalwares*, mecanismos de criptografia e *hashing*, sistemas de detecção de intrusão, controles físicos, entre outros. Um sistema de detecção de intrusão é um mecanismo que auxilia o administrador de rede "[...] descobrir, determinar e identificar o acesso não autorizado, duplicação, alteração e destruição de um sistema de informação" (BUCZAK; GUVEN, 2016). Este tipo de mecanismo depende da coleta e processamento dos dados que podem ser obtidos através de eventos gerados em um host ou tráfego de rede gerado e exportado pelos dispositivos de rede.

Na literatura duas abordagens têm sido utilizadas para categorizar um IDS: baseado em assinatura ou em anomalia (AMARAL, 2015 *apud* SCHNEIDER, 2012) (CARVALHO, 2018 *apud* AHMED *et al.*, 2016). A primeira abordagem possui uma

base de assinaturas formada por ataques conhecidos com o objetivo de detectar uma intrusão quando os dados de rede monitorados combinam com algum registro presente nesta base (HAMID; SUGUMARAN; JOURNAUX, 2016). Este tipo de técnica é eficaz em “[...] detectar ataques conhecidos sem gerar uma grande quantidade de falsos alarmes”, porém, novos tipos de ataques (*e.g. zero-day attack*) não são detectados por não haver registros na base de assinaturas necessitando assim, de frequentes atualizações manuais (BUCZAK; GUVEN, 2016).

Um IDS baseado em anomalia “[...] estuda o comportamento normal da rede e do sistema e identifica anomalias como desvios” deste padrão (XIN *et al.*, 2018). No entanto, “definir um estado normal do tráfego de rede não é uma tarefa simples sendo este o maior desafio enfrentado por esse sistema” (AMARAL, 2015 *apud* PROENÇA, 2005). Uma das principais vantagens deste paradigma consiste na capacidade em detectar anomalias desconhecidas, contudo, devido a este fator, um número maior de falsos alarmes pode ser gerado (HAMID; SUGUMARAN; JOURNAUX, 2016). “Outra vantagem, é que o perfil das atividades normais pode ser personalizado para cada sistema, aplicação ou rede, dificultando assim que os atacantes saibam quais atividades eles podem realizar sem serem detectados” (BUCZAK; GUVEN, 2016).

Nos últimos anos, os sistemas de detecção de intrusão baseados em anomalia têm utilizado o aprendizado de máquina para a extração automática dos padrões dos dados de rede, ao contrário dos sistemas baseados em assinatura, onde a extração é realizada manualmente (NAJAFABADI *et al.*, 2015). “Esta técnica considera determinados parâmetros de rede como características e baseado neles, classifica o tráfego de rede como normal ou anômalo” (METI; NARAYAN; BALIGAR, 2017). Desta maneira, a definição do comportamento da rede torna-se um processo mais simplificado, pois ele será criado por algoritmos especializados que aprendem diretamente com os dados coletados da rede.

2.4 Aprendizado de máquina

“O aprendizado de máquina é um subcampo da inteligência artificial que dá ao computador a capacidade de aprender sem ser explicitamente programado” (DAS;

NENE, 2017). Esta área de estudo está “[...] estreitamente relacionada com estatística computacional que se concentra na previsão usando computadores e na otimização matemática que fornece métodos, teorias e aplicações de domínio [...]” (XIN *et al.*, 2018). Os algoritmos providos por este paradigma extraem automaticamente os padrões contidos em grandes quantidades de dados, superando as limitações do entendimento humano na realização desta tarefa. Dependendo do problema e da natureza da base de dados podem haver duas principais abordagens conhecidas como aprendizado supervisionado e não supervisionado.

No aprendizado supervisionado em problemas relacionados à classificação, o algoritmo possui a “[...] tarefa de aproximar uma função de mapeamento das variáveis de entrada para uma variável de saída” que consiste em uma classe para uma dada observação (BROWNLEE, 2017a). A classe associada é considerada a resposta correta para o conjunto de características daquela instância. O objetivo desta abordagem é classificar corretamente uma nova instância não observada durante o treinamento de acordo com os padrões aprendidos pelo modelo. De modo diferente, os modelos não supervisionados realizam o aprendizado através da dedução de estruturas ocultas baseadas na similaridade ou dissimilaridades dos dados devido a inexistência de uma classe para cada amostra (XIN *et al.*, 2018). Os principais procedimentos aplicados para resolver problemas que envolvem o aprendizado de máquina podem ser observados na Figura 2.

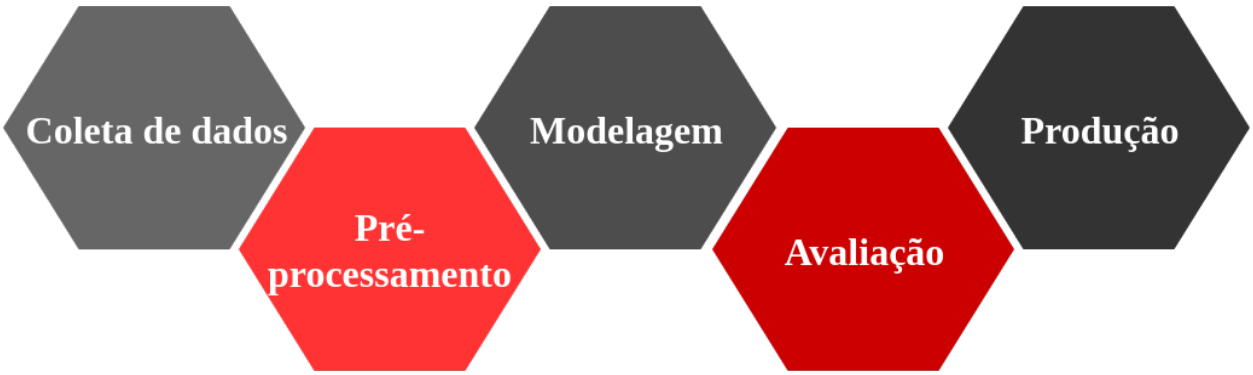


FIGURA 2 - Procedimentos para a resolução de problemas no aprendizado de máquina.
FONTE: Autor (2019).

O primeiro procedimento tem por objetivo encontrar técnicas e fontes para coletar os dados relevantes sobre o problema que pretende-se solucionar, criando assim, a base de dados bruta para uso nas etapas seguintes (ALTEXSOFT, 2019). “O

pré-processamento consiste no processo de aplicar o conhecimento do domínio durante a extração das características relevantes para reduzir a complexidade dos dados e gerar padrões que fazem com que os algoritmos de aprendizado funcionem melhor" (XIN *et al.*, 2018). Na etapa de modelagem a base de dados tratada no pré-processamento é dividida em dois conjuntos de dados para que os algoritmos sejam treinados, testados e os hiperparâmetros sejam encontrados. A etapa de avaliação recebe como entrada os resultados da modelagem e avalia a performance dos algoritmos de acordo com as métricas apropriadas. A última etapa treinará o melhor algoritmo considerando toda a base de dados para posteriormente, executar o modelo criado em um ambiente de produção para que os objetivos propostos sejam alcançados (BUCZAK; GUVEN, 2016).

No contexto da detecção de ataques, "os métodos de aprendizado de máquina foram implantados e considerados bem sucedidos em redes com e sem fio e por conta disso, também foram usados para a detecção de ataques em rede definida por software provando dar resultados efetivos" (METI; NARAYAN; BALIGAR, 2017). Com o enfoque em analisar o comportamento dos fluxos IP classificando cada um deles como normal ou anômalo optou-se pelo aprendizado de máquina supervisionado voltado para a técnica de classificação. Com base nas pesquisas realizadas por Buczak e Guven (2016), Das e Nene (2017) e Xin *et al.* (2018) sobre os algoritmos de aprendizado de máquina para a segurança cibernética, foram identificados os principais classificadores, *Decision Tree* (DT), *Gaussian Naive Bayes* (GNB), *K-Nearest Neighbors* (KNN), *Support Vector Machine* (SVM) e *Multi-Layer Perceptron* (MLP). A Tabela 1 apresenta uma breve descrição dos classificadores selecionados.

TABELA 1 - Algoritmos de aprendizado de máquina supervisionado.

Algoritmo	Descrição
Decision Tree	Estrutura de árvore composta por folhas que representam as classes e ramos formados por regras de decisão baseadas nas características de entrada.
Gaussian Naive Bayes	Teorema de Bayes que supõe de forma ingênua a independência condicional entre cada par de características dado o valor da classe. A probabilidade das características é assumida como Gaussiana.
K-Nearest Neighbors	Aprendizado por instância (*instance learning*) que classifica uma nova entrada com base nas *k* amostras mais próximas em distância dela.
Support Vector Machine	Descoberta de um hiperplano de separação no espaço de características entre duas classes, de forma que a distância entre o hiperplano e os pontos de dados mais próximos de cada classe seja maximizada.
Multi-Layer Perceptron	Rede de neurônios artificiais inspirada no cérebro humano, no qual cada neurônio realiza uma pequena parte do processamento, transferindo o resultado entre as camadas até chegar na última onde a classe é definida.

FONTE: Buczak e Guven (2016), Lobato *et al.* (2016), Najafabadi *et al.* (2015) e Scikit-learn (2019).

2.5 Rede definida por software

As ações de gerência de rede para atender a demanda das novas tecnologias como computação em nuvem (*cloud computing*), virtualização de funções rede (*network function virtualization*), Internet das Coisas têm sido dificultadas pela complexidade e heterogeneidades das redes de computadores tradicionais. Além disso, detectar e mitigar a ocorrência de ataques de rede se tornou uma tarefa árdua e improfícua de ser realizada manualmente devido ao aumento no número de dispositivos de rede com suas próprias configurações. A rede definida por software surgiu como alternativa para endereçar estes desafios proporcionando um padrão mais simplificado e livre das limitações encontradas nas arquiteturas de redes existentes, possibilitando um melhor controle sobre os recursos da rede (MOUSAVI, 2014).

Os dispositivos de rede convencionais possuem acoplados internamente o plano de dados e o plano de controle, ambos, exclusivamente definidos pelos fabricantes e geralmente inflexíveis impedindo o desenvolvimento de novas funcionalidades. O plano de dados possui a responsabilidade de encaminhar os pacotes trafegados na rede por meio dos protocolos e regras estabelecidos pelo plano de controle. Diferentemente, na SDN estes “planos são desacoplados permitindo que o controle da rede seja diretamente programável e a infraestrutura abstraída para aplicativos e serviços de rede” (OPEN NETWORKING FOUNDATION, 2019). Agora, os dispositivos de rede tornam-se elementos simplificados sendo responsáveis somente pela transmissão de dados enquanto que a lógica de controle foi transferida para um elemento externo denominado controlador SDN. A arquitetura SDN é dividida em três camadas principais, conforme apresentado na Figura 3.

FIGURA 3 - Arquitetura SDN.
FONTE: Autor (2019).

A camada de infraestrutura representa os dispositivos de encaminhamento (*e.g. switches* e roteadores) “[...] compostos por uma ou mais tabelas de fluxos ligadas, que possuem entradas configurados pelo controlador” (CARVALHO, 2018). Cada entrada é composta por um conjunto de regras formada por uma “[...] tupla de doze elementos (*e.g.* MAC de origem/destino, IP de origem/destino, portas de origem/destino, protocolo, prioridade, etc.) contendo as informações dos protocolos da camada de enlace, rede e transporte, da arquitetura TCP/IP” (COSTA, 2013). Além disso, uma entrada está associada com uma ação, contadores para a geração de estatísticas e outras especificações de acordo com a versão do protocolo OpenFlow utilizada.

O OpenFlow é um protocolo aberto proposto para padronizar a comunicação entre os *switches* e o controlador SDN (YAN *et al.*, 2016). Todas as mensagens transmitidas entre a camada de infraestrutura e controle através do “[...] OpenFlow são realizadas por meio de um canal seguro usando o protocolo TLS (*Transport Layer Security*) ou em casos onde a criptografia não seja aplicada, através do protocolo TCP” (CARVALHO, 2018). Os pacotes trafegados na rede são comparados com as regras das tabelas de fluxos em busca de uma combinação (*match*). Caso ocorra, os pacotes relacionados prosseguirão para o destino conforme as ações definidas pelo administrador de rede. Por outro lado, caso não seja encontrada uma combinação (*table-miss*), será enviada uma mensagem através do protocolo OpenFlow para a camada superior solicitando uma decisão do controlador.

O elemento centralizado presente na camada de controle denominado controlador atua como um sistema operacional provendo a abstração da infraestrutura da rede, serviços essenciais e uma API para o desenvolvimento de aplicações (CARVALHO, 2018). No caso supracitado, onde a camada inferior solicita uma decisão, o *switch* OpenFlow armazenará em *buffer* os pacotes e enviará uma mensagem de *packet_in* contento apenas o cabeçalho do pacote para o controlador (WANG; XU; GU, 2015). O controlador analisará os dados do cabeçalho e enviará como resposta uma mensagem de *packet_out* contendo uma regra, ação e outras associações que serão inseridas na tabela de fluxo do dispositivo que realizou a solicitação.

A camada de aplicação permite o desenvolvimento de novas soluções como protocolos de roteamento, balanceamento de carga, sistemas de detecção e mitigação, entre outros. Esta camada provê a abstração necessária dos conjuntos de instruções de baixo nível usados pelo protocolo OpenFlow para programar os dispositivos de encaminhamento. A comunicação realizada entre a camada de aplicação e controle é realizada através de API's padronizadas pelo controlador utilizado. Sendo assim, o administrador de rede não precisa mais lidar com diferentes interfaces (*e.g. Command Line Interface* - CLIs) e configurações específicas dos dispositivos, uma vez que eles possuam habilitados a tecnologia OpenFlow (AIZUDDIN *et al.*, 2017).

2.5.1 Ponto de falha

O controlador possibilita o controle de inúmeros *switches* OpenFlow de forma centralizando sendo considerando um dos principais elementos e vantagens na perspectiva da gerência de rede. Porém, em uma infraestrutura SDN tradicional, se o controlador falhar toda a rede poderá ficar indisponível e devido a este fator, ele se torna um ponto de falha e alvo de ataques (DACIER *et al.*, 2017)(YAN *et al.*, 2016). Em um ataque DoS do tipo TCP *flooding* por exemplo, utilizando portas de origem aleatórias, a cada pacote enviado para o alvo (*e.g.* servidor web) não haverá uma regra nas tabelas de fluxos dos *switches* OpenFlow. Neste caso, será necessário a inserção de uma nova regra pelo controlador para cada fluxo iniciando a troca de mensagens de *packet_i*n e *packet_out*. O primeiro componente a ser afetado será o elemento de comutação, pelo fato de possuir uma memória limitada para armazenar em buffer os pacotes até que as regras sejam instaladas. Posteriormente, o controlador terá que processar as inúmeras solicitações enviadas e como resultado, um grande volume de tráfego será gerado consumindo o recurso de computação do controlador em pouco tempo (WANG; XU; GU, 2015).

Assim, a ocorrência de um ataque em uma rede definida por software pode ser altamente prejudicial, se o tempo de detecção e aplicação de medidas de reparo forem demorados. Portanto, a detecção antecipada de um ataque se torna crucial neste paradigma (PANDIKUMAR; ATKILT; HASSEN, 2017). Para isto são requeridos a fonte de dados adequada, uma janela de tempo de exportação curta e técnica modernas

que reduzam a intervenção humana.

2.6 Trabalhos relacionados

2.6.1 Detecção e mitigação de anomalias em rede definida por software

Em Pandikumar *et al.* (2017) é proposto um mecanismo para detectar antecipadamente os ataques DDoS em SDN com multicontroladores. A solução é implementada dentro do controlador com o objetivo de evitar a sobrecarga no pico do ataque. Utiliza-se o cálculo de entropia associado ao IP de destino em uma janela de 50 pacotes que chegam ao controlador e o resultado é comparado com um limiar definido pelos autores. É considerado um ataque quando o valor dos cálculos de entropia permanecerem abaixo do limiar por cinco janelas consecutivas. Para mitigar o ataque o tempo de permanência dos fluxos maliciosos nos elementos de comutação é alterado para um período menor com o intuído de evitar a sobrecarga dos *switches* devido às limitações de recurso.

Em Singh *et al.* (2015), foi desenvolvido um mecanismo de prevenção de ataques DoS baseado em uma infraestrutura de rede definida por software dividido em quatro etapas. Cada etapa é responsável por efetuar uma ação com o intuído de interromper a fonte causadora do tráfego intenso, caso continue, a próxima é acionada. A primeira etapa analisa o número de pacotes por segundo e se o valor exceder a 1000, o tráfego é direcionado para um mecanismo de *buffering* que armazena os pacotes em uma fila. A segunda etapa realiza o bloqueio durante um período de 100 segundos. Na terceira etapa é gerado durante três tentativas um pacote de solicitação ECHO para que o IP fonte diminua a taxa de transferência do tráfego de envio. Caso todas as etapas não sejam suficientes para interromper o tráfego intenso, um ataque foi detectado. Mediante a este fator, a quarta etapa bloqueia o IP do atacante através de um comando enviado pelo controlador.

No trabalho de Wang *et al.* (2015) buscou-se prevenir a perda dos tráfegos normais e da infraestrutura SDN durante um ataque DoS. A solução desenvolvida chamada de *FloodGuard* possui dois módulos implementados como aplicações de

controle. O primeiro módulo tem a função de dinamicamente inserir regras de fluxo de forma proativa em tempo de execução por meio de uma análise das mensagens de *packet_in* gerenciadas pelo controlador. As regras proativas visam a separação dos fluxos normais dos maliciosos evitando a sobrecarga do plano de dados e controle. O segundo módulo armazena temporariamente em um dispositivo os fluxos provenientes do ataque e os envia ao controlador por meio do algoritmo de agendamento Round-Robin e a uma taxa limitada. Para detectar a ocorrência de um ataque os autores comparam o resultado do cálculo da porcentagem de uso da rede com base na taxa de tempo real das mensagens de *packet_in* e dos recursos da infraestrutura com um limiar estabelecido.

As soluções apresentadas nos trabalhos supracitados requerem que seja realizada uma análise sobre o comportamento da rede para a definição de limiares que serão usados para a detecção de ataques. Devido o surgimento de novos serviços e aplicações diariamente, as atuais de rede de computadores se tornaram dinâmicas, fazendo que tais abordagem necessitem de constantes atualizações. Desta maneira, buscou-se explorar neste trabalho o uso do aprendizado de máquina para detecção de ataques que possibilita a identificação automática de padrões para diferir um tráfego anômalo de um normal. Além disso, diferente de Pandikumar *et al.* (2017) e Wang *et al.* (2015) que utilizam a inspeção de pacotes como fonte de dados, optou-se pe utilização dos fluxos IP para reduzir o tempo de processamento durante a detecção de ataques.

2.6.2 Aprendizado de máquina para detecção de intrusão

Belouch *et al.* (2018) avaliaram a performance do desempenho da detecção de intrusão baseada nos algoritmos de aprendizado de máquina *Decision Tree*, *Naive Bayes*, *Random Forest* e *Support Vector Machine* utilizando o Apache *Spark*. Foi utilizada a base de dados UNSW-NB15 contendo tráfego de rede normal e anômalo com cerca de 49 características e 9 famílias de ataques. Esta base de dados possui um conjunto de treinamento com 175340 amostras e um de teste contendo 82000 amostras. Para avaliar o desempenho dos algoritmos foram utilizadas as métricas de acurácia, sensibilidade, especificidade, tempo de treinamento e tempo de predição. Os

autores salientam que o *Random Forest* obteve o melhor desempenho em relação aos demais classificadores alcançando uma acurácia de 97.49% e o menor tempo de predição com cerca de 0.08 segundos. O pior desempenho foi do algoritmo *Naive Bayes* que obteve uma acurácia de 74.19%, embora tenha atingido o menor tempo de treinamento com um total de 2.25 segundos.

Hamid *et al.* (2016) realizaram uma análise comparativa de diferentes técnicas de aprendizado de máquina supervisionado da ferramenta Weka para a detecção de intrusão. Devido a considerações computacionais os autores utilizaram 10% da base de dados KDD CUP 1999 totalizando 494020 amostras de dados brutos TCP simulados compostos por 22 tipos de ataques divididos em cinco grupos principais. Foram utilizadas as métricas de avaliação oriundas da matriz de confusão em conjunto com a técnica de 10-*folds cross-validation* levando em consideração a média dos resultados gerados. Relatou-se que o melhor classificador foi o *PART* do grupo *Rule Based* com 99.97% de *F-Measure* enquanto que o pior foi o *Input Mapped* atingindo apenas 41.20%. Foi levado em consideração a técnica de redução de características com o objetivo de diminuir a complexidade do conjunto de dados para melhorar o custo computacional. O resultado foi a seleção de 11 características das 41 através do avaliador *CfsSubSetEval* com o método de busca *BestFirst*.

Najafabadi *et al.* (2015) propuseram a detecção de ataques de força bruta através do aprendizado de máquina e dados agregados Netflow. Os autores identificaram que as características dos fluxos entre uma falha de login e um ataque de força bruta não eram discriminativas o suficiente. Para contornar essa situação foi desenvolvido um método para agregar os fluxos com o mesmo IP de origem, IP de destino e porta de destino observados durante um intervalo de 5 minutos. A base de dados gerada contém 425 instâncias anômalas, 558 normais e um total de 19 características. Os algoritmos escolhidos foram o 5-*Nearest Neighbor*, *C4.5D*, *C4.5N* e *Naive Bayes* ambos provenientes da ferramenta Weka. Foi executado 4 vezes o 5-*folds cross-validation* em conjunto com *Area Under the receiver operating characteristic Curve* (AUC). Foi calculado a média final com base em 20 AUC e os resultados demonstram um total acima de 98% para todos os classificadores. Os autores concluem que os bons resultados foram devido a agregação realizada que resultou em

características mais discriminativas.

Os trabalhos revisados apresentam em comum uma análise comparando diversos algoritmos de aprendizado de máquina supervisionados para a detecção de intrusão. De modo semelhante este trabalho seguiu a mesma linha de pesquisa distinguindo-se dos demais trabalhos ao comparar tais algoritmos em uma base de dados com fluxos IP em sua forma original e agregado. O objetivo é avaliar a influência que a agregação causa no tempo de processamento e nos resultados das predições. As características utilizadas para a agregação e a janela temporal para a coleta dos fluxos diferem-se de Najafabadi *et al.* (2015) que embora apresentem um método para agregar os fluxos IP, não comparam os resultados em relação aos fluxos IP não agregados. Desta maneira, não é possível aferir se realmente houve um impacto nos resultados dos classificadores.

2.6.3 Aprendizado de máquina para detecção de anomalias em rede definida por software

Bhunia e Gurusamy (2017) propuseram um framework seguro de IoT baseado em rede definida por software denominado *SoftThings* para detectar e mitigar anomalias. A solução é composta por um controlador SDN mestre que se conecta nos controladores clusters formados por três módulos responsáveis pelo aprendizado, classificação e gerenciamento de fluxos dos *switches* OpenFlow. Foi utilizado pelos autores o *Support Vector Machine* linear e não linear para a detecção de anomalias. Os experimentos foram realizados através do emulador mininet e do controlador SDN POX sendo considerados os cenários onde o dispositivo IoT era o alvo do ataque, o dispositivo IoT comprometido realizava um ataque e vários dispositivos IoT realizavam um ataque DDoS. Os resultados comprovam que o SVM não linear atingiu resultados mais precisos alcançando para cada um dos respectivos cenários uma precisão de 98%, 97% e 98%. A mitigação foi realizada através do bloqueio dos fluxos anômalos resultando na restauração da taxa de transferência com uma média de 2.7 segundos.

Meti *et al.* (2017) implementaram no controlador Ryu, um sistema de detecção de intrusão baseado nos algoritmos de aprendizado de máquina *Naive Bayes*,

Support Vector Machine e *Neural Network*. Foi desenvolvida uma base de dados em tempo real a partir da coleta do tráfego TCP do laboratório dos autores considerando os atributos *host_time* e *no_of_request* que consistem no número de host conectados por segundos. Ao detectar um host com comportamento anormal sua comunicação é negada e o MAC (*Media Access Control*) inserido em uma ACL (*Access Control List*) mantida no controlador para detectar os próximos ataques. O SVM demonstrou nos resultados ser a melhor escolha apresentando para ambas as métricas avaliadas com uma taxa de 80%. O *Naive Bayes* obteve o pior desempenho que segundo os autores foi devido a pouca quantidade de dados utilizada. A *Neural Network* obteve resultados satisfatórios nas métricas de acurácia e precisão, porém apresentou uma baixa revocação.

Mohammed *et al.* (2018) propuseram um mecanismo colaborativo de mitigação de DDoS baseado no aprendizado de máquina em rede definida por software. A solução desenvolvida foi armazenada em um servidor dedicado responsável por executar o algoritmo *Naive Bayes* com base nas características extraídas das mensagens de packet-in enviados pelos controladores. Na detecção de um pacote anormal o servidor realizará um processo de comunicação entre os controladores para que o dispositivo atacante seja bloqueado. Foi utilizado para o treinamento do modelo a base de dados NSL-KDD usando 25 das 41 características existentes resultando em um *F1-Score* de 98%. Os autores realizaram no ambiente de rede criado um ataque DDoS real do tipo SYN *flooding* originado de três controladores para um controlador alvo resultando em um *F1-Score* de 66%. Os autores identificaram características redundantes na base de dados, e depois de eliminadas, o modelo foi treinado novamente e o resultado alcançado foi de 77%.

As soluções apresentadas buscam detectar ataques em uma rede definida por software. No entanto, alguns autores utilizaram base de dados com alguns problemas, tais como dados antigos, duplicados e não representativos, enquanto que outros, utilizaram apenas um algoritmo de aprendizado de máquina não justificando a escolha dele em relação aos demais. Considerando estas questões este trabalho busca apresentar uma base de dados com o atual cenário das redes de computadores e realizar um estudo para a escolha do melhor algoritmo de aprendizado de máquina

com os seus hiperparâmetros através de uma análise comparativa. Como contribuição adicional foi desenvolvido um sistema web para que o administrador de rede possa configurar os parâmetros de pré-processamento e escolher o melhor algoritmo de aprendizado de máquina para o ambiente de rede monitorado.

3 SISTEMA PROPOSTO

O desenvolvimento do sistema proposto neste trabalho tem como objetivo principal minimizar a intervenção humana na detecção e reparo de ataques utilizando para estes fins, o aprendizado de máquina e a rede definida por software. Seguindo uma prática comum da área do aprendizado de máquina, foram realizados testes iniciais com o objetivo de explorar o problema para obter uma melhor compreensão, uma vez que não se sabe exatamente quais serão os melhores procedimentos. A medida que o número de linhas do código foi aumentado, foram criados novos pacotes e módulos baseado no princípio da ortogonalidade, ou seja, componentes simples e independentes que interagem entre si. Desta maneira, o sistema foi sendo implementado de forma gradual com novas funcionalidades sendo adicionados à medida que a pesquisa avançava.

O resultado de todo o esforço foi a criação de um sistema web, que integra o aprendizado de máquina e a rede definida por software para detectar e mitigar ataques de rede buscando a minimização da intervenção humana. A arquitetura de software utilizada foi o MVC (*Model-View-Controller*), que possibilitou estruturar o código fonte de maneira organizada colaborando para a manutenção dele durante o desenvolvimento. A linguagem Python 3.7[1] foi utilizada para a implementação do sistema devido ela ser uma linguagem de programação de propósito geral como uma sintaxe simples e rápida para prototipar disponibilizando uma ampla gama de bibliotecas para área de ciência dos dados, aprendizado de máquina, desenvolvimento web, entre outros. A principal biblioteca utilizado para a implementação dos algoritmos de aprendizado de máquina, métodos de mineração dos dados e ferramentas

[1] https://docs.python.org/3.7/

para análise dos dados foi o *scikit-learn*[2]. Para a construção da aplicação web foi utilizado o *microframework* Flask[3] e por meio das tecnologias HTML[4] (*HyperText Markup Language*), CSS[5] (*Cascading Style Sheets*), JavaScript[6] foi gerado a interface gráfica para a interação do usuário.

FIGURA 4 - Página inicial do sistema desenvolvido.
FONTE: Autor (2019).

A página inicial do sistema apresentada na Figura 4, demonstra os dois principais componentes do sistema denominados *Intrusion Prevention System* (IPS) e *Network Dataset Creation* (NDC). A terminologia IPS foi utilizada porque a aplicação estende as capacidades de um tradicional IDS que consiste em um sistema que somente realiza a detecção necessitando a intervenção humana para analisar as intrusões para executar alguma ação. Um IPS é um “sistema de controle capaz de aceitar e rejeitar os pacotes baseados em um conjunto de regras” com o objetivo de bloquear um ataque detectado (PETTERS, 2018). Assim, através do aprendizado de máquina o sistema desempenha o papel de detecção e com a rede definida por software todo o controle necessário para prevenir automaticamente que uma atividade maliciosa prossiga é alcançado.

[2] https://scikit-learn.org/stable/index.html
[3] https://flask.palletsprojects.com/
[4] https://www.w3schools.com/tags/
[5] https://www.w3schools.com/cssref/
[6] https://www.w3schools.com/js/

3.1 Network Dataset Creation

A base de dados é um dos fatores que determinam quais os tipos de ataques serão detectados, implicando também na escalabilidade, que deve ser levada em consideração para a implantação da solução em grandes redes (AMARAL *et al.*, 2017). Portanto, o componente NDC foi projeto para a partir de uma base de dados de rede no formato pcap construir uma nova base composta por fluxos IP minimizando assim, o volume de dados coletados, processados e armazenados (MORO *et al.*, 2018). A ideia de criar um componente reutilizável foi visando as constantes mudanças no perfil do tráfego de rede devido ao surgimento de novos serviços e aplicações executadas nas redes atuais. Na ocorrência de situações onde a base de fluxos IP não reflita mais a realidade do ambiente de rede monitorado, será possível criar e imediatamente disponibilizar uma nova base de fluxos IP para o componente IPS. A Figura 5 demonstra as principais funcionalidades executados pelo NDC.

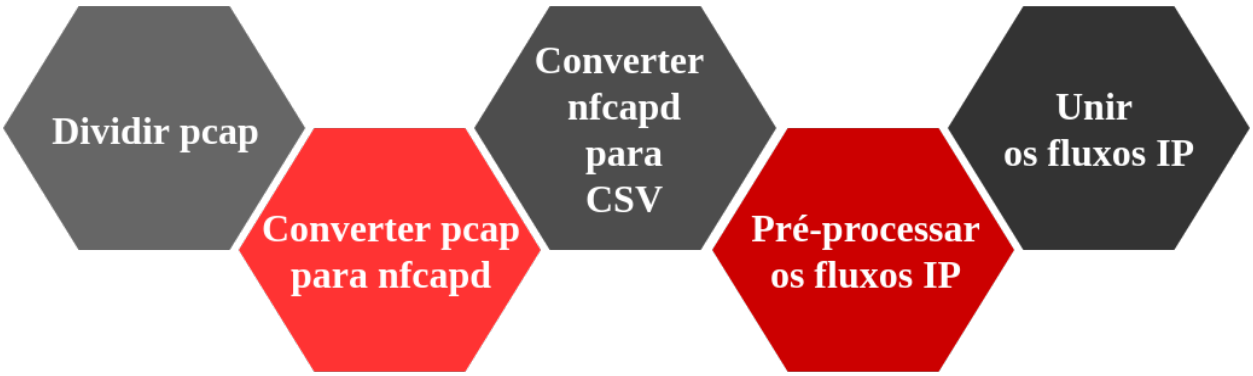


FIGURA 5 - Funcionalidades do NDC.
FONTE: Autor (2019).

A primeira funcionalidade utiliza a ferramenta tcpdump[7] para dividir um arquivo pcap caso ele seja demasiadamente grande para o processamento, enquanto que a segunda por meio de outra ferramenta denominada nfpcapd[8] converte tais arquivos para o formato nfcapd que armazena os dados de acordo com o protocolo NetFlow. O arquivo nfcapd não é legível pela aplicação sendo necessário a execução do terceiro procedimento que irá convertê-lo para um arquivo CSV (*Comma-Separated Values*) gerando os fluxos IP que serão utilizados nos demais

[7] https://www.tcpdump.org/
[8] https://github.com/phaag/nfdump

procedimentos. Para este fim, o nfdump foi utilizado que com as outras ferramentas representam um conjunto de programas comumente utilizado para o monitoramento da rede. Vale ressaltar que cada funcionalidade não depende da execução da outra, porém, deve haver os arquivos necessários para realizar tal procedimento.

A quarta funcionalidade é responsável por pré-processar os fluxos IP armazenados no arquivo CSV, sendo um aspecto importante para alcançar o conjunto de dados final que será utilizado durante a criação dos modelos (BUCZAK; GUVEN, 2016). Foi observado durante o desenvolvimento desta funcionalidade que muitas características dos fluxos IP não contribuem para o aprendizado dos algoritmos por não serem representativas ou conterem valores nulos. Foram selecionadas então, 3 características principais de um fluxo IP: duração em segundos (td), quantidade de pacotes (pkt) e quantidade de bytes (byt) conforme destacado em negrito na Tabela 2. Os IPs de origem e destino foram omitidos por motivos de privacidade devido serem endereços públicos.

TABELA 2 - Fluxos IP com as características selecionadas em negrito.

sa	da	pr	flg	sp	dp	**td**	**pkt**	**byt**
.....165	173	udp	[0]	58853	547	**5**	**5**	**3648**
.....165	119	tcp	[0, 1, 1, 0, 0, 1]	54547	443	**8**	**65**	**5420**
.....165	119	tcp	[0, 1, 1, 0, 1, 1]	54552	443	**13**	**12**	**900**
.....165	119	tcp	[0, 1, 1, 0, 1, 1]	54551	443	**13**	**11**	**868**

FONTE: Autor (2019).

Além disso, foi analisado que o desenvolvimento de um método de agregação dos fluxos IP com base no IP de origem, IP de destino e protocolo em um intervalo de tempo definido pelo usuário impactaria no tempo de processamento e resultados dos classificadores. Como exemplo, um ataque DoS envia milhares de requisições que se assemelham muito a uma requisição normal realizada por um usuário, o que dificulta a extração de algum tipo de padrão por parte dos algoritmos. Ao agregar os fluxos originados destas requisições se obtém dados relevantes que contribuirão para a distinção de um tráfego legítimo. Foi estabelecido um parâmetro de limite para a agregação visando manter a base de dados balanceada devido à existência de casos em que milhares de fluxos seriam reduzidos para uma única amostra. Quando a quantidade de fluxos (flw) atinge o limite estabelecido, a agregação é interrompida e um novo fluxo é iniciado. A Tabela 3 apresenta o resultado da aplicação deste método

nos fluxos IP apresentados na Tabela 2.

TABELA 3 - Fluxos IP agregados com as características selecionadas em negrito.

sa	da	pr	flg	sp	dp	**td**	**pkt**	**byt**	**bps**	**bpp**	**pps**	**nsp**	**ndp**	**flw**
.....165	173	udp	[0]	{58853}	{547}	**5**	**5**	**3648**	**5837**	**5837**	**1**	**1**	**1**	**1**
.....165	119	tcp	[0, 3, 3, 0, 2, 3]	{54547, 54551, 54552}	{443}	**13**	**88**	**7188**	**4423**	**653**	**7**	**3**	**1**	**3**

FONTE: Autor (2019).

Nota-se que as características de porta de origem (sp) e porta de destino (dp) não foram utilizadas, pois o valor não era representativo para o aprendizado. Porém, durante a agregação foram contadas as ocorrências únicas delas resultando na criação de duas novas características denominadas número de porta de origem (nsp) e número de porta de destino (ndp). Na Tabela 2 os três últimos fluxos duraram respectivamente 8, 13 e 13 segundos, contudo, ambos iniciaram no mesmo período (13:41:08) e finalizam em tempos diferentes (13:41:16, 13:41:21 e 13:41:21). Para agregar estes fluxos e gerar a duração da comunicação foi considerado o tempo de início e o tempo de término com maior período resultando em 13 segundos. Visando acrescentar novos dados representativos, foram computadas a partir das características existentes a quantidade de bytes por segundos (bps), bytes por pacotes (bpp) e pacotes por segundo (pps). Além disso, foi contabilizada a quantidade de fluxos (flw) na comunicação que em conjunto com o nsp e ndp distinguem dos fluxos normais, determinados tipos de ataques que utilizam as portas de origem ou destino aleatórias.

Para que seja possível aplicar o aprendizado de máquina supervisionado nos fluxos IP pré-processados eles deverão ser classificados como normal (0) ou como ataque (1) para que o algoritmo possa relacionar os padrões extraídos a uma das classes existentes. A funcionalidade de pré-processamento pode ser realizada inúmeras vezes até que a quantidade de dados necessária seja alcançada. Ao finalizar este procedimento, todos os fluxos IP pré-processados serão unificados para a formação da base de dados final através da execução do último procedimento. Todas as etapas do componente NDC utilizam uma rota genérica que se adapta de acordo com o procedimento escolhido pelo usuário. Na Figura 6 está sendo demonstrado o procedimento de dividir um arquivo pcap.

FIGURA 6 - Página para a seleção dos arquivos.
FONTE: Autor (2019).

Cada funcionalidade está vinculada a um diretório base onde os arquivos são lidos e armazenados. Neste caso, ao escolher *Split pcap* (dividir pcap) serão listados todos os subdiretórios e arquivos do diretório raiz pcap. Na Figura 6, o subdiretório *syn_flood* do diretório pcap foi selecionado resultando na atualização do conteúdo da página. Dos três arquivos disponíveis somente o primeiro foi selecionado para ser dividido de acordo com o parâmetro definido pelo usuário na página de configuração que pode ser observada na Figura 7.

FIGURA 7 - Página para a definição dos parâmetros.
FONTE: Autor (2019).

Para esta funcionalidade foi disponibilizado apenas um parâmetro que

consiste no tamanho que o arquivo pcap selecionado será dividido (*Split size*). Ao enviar o formulário a funcionalidade iniciará a execução dos procedimentos relacionados a ela e a página será atualizada mostrando para o usuário a mensagem de "*Loading*…". Todos os formulários disponibilizados realizam a validação dos dados antes do envio para o processamento. Caso algo esteja errado ou algum campo seja deixado em branco, o usuário será comunicado com a respectiva mensagem de erro. A Figura 8 apresenta um exemplo onde o botão *Submit* foi clicado com um parâmetro inválido.

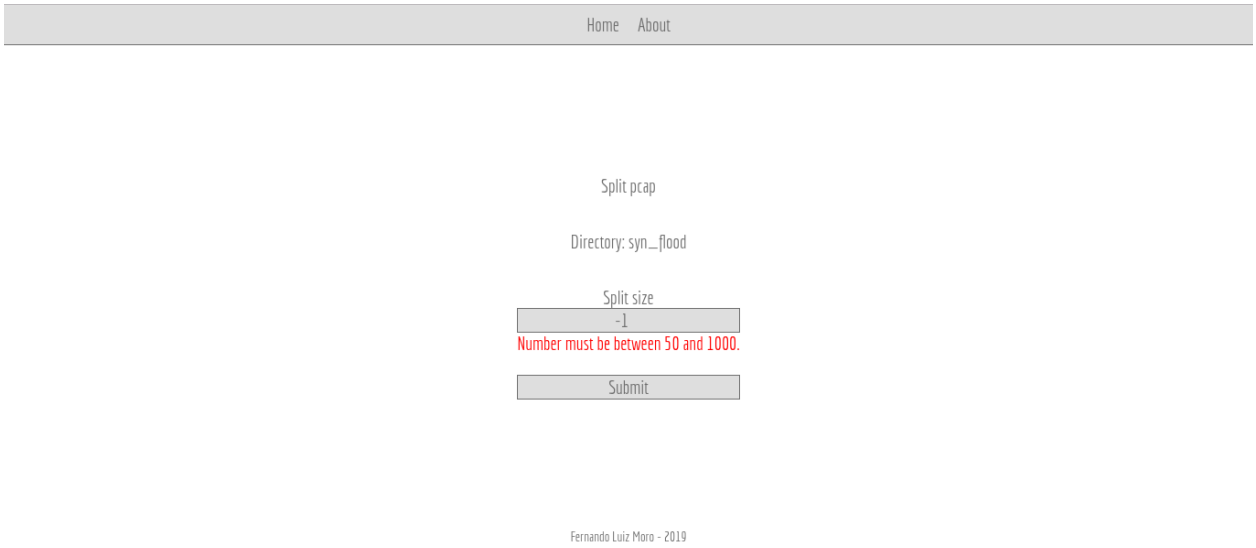


FIGURA 8 - Página indicando um erro no formulário.
FONTE: Autor (2019).

Ao final de cada funcionalidade são armazenados os novos arquivos e o usuário é redirecionado para a página na qual o procedimento foi iniciado, neste cenário, na página *Split pcap* no diretório pcap/syn_flood.

3.2 Intrusion Prevention System

O componente IPS integra o aprendizado de máquina e a rede definida por software proporcionando um sistema de segurança automatizado para monitorar o ambiente de rede em tempo real. Desta maneira, o administrador de rede poderá concentrar suas ações em aspectos mais analíticos uma vez que a complexidade das atividades de detecção e mitigação são desempenhadas pelo sistema. Conforme a

Figura 9, o IPS é dividido em quatro procedimentos principais que são executados a partir da saída gerado pelo seu antecessor.

FIGURA 9 - Procedimentos do IPS.
FONTE: Autor (2019).

De acordo com a Figura 4 o usuário deverá escolher uma das duas opções disponibilizados pelo IPS que são a criação de um novo modelo (*New*) ou carregar um modelo existente (*Load*). Caso a segunda opção seja selecionada os procedimentos de configuração e modelagem serão pulados iniciando o funcionamento do sistema em tempo real. Para exemplificar todos os passos, será considerado a primeira interação do usuário com o sistema onde a primeira opção deverá ser escolhida. Ao clicar em *New* o usuário será direcionado para a página de configuração da base de fluxos IP conforme a Figura 10.

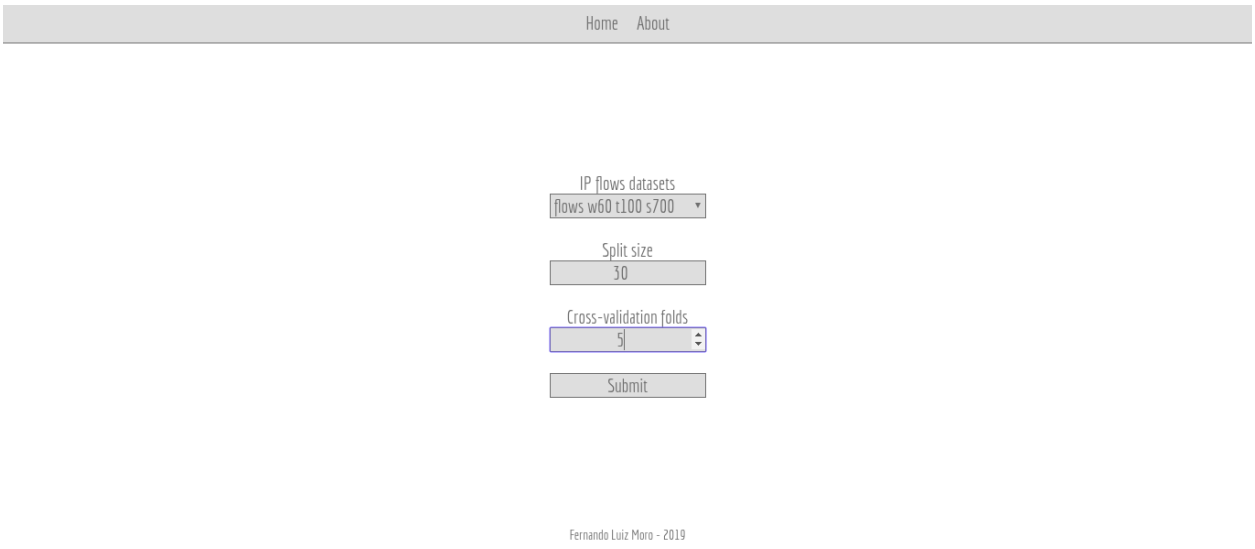


FIGURA 10 - Página de configuração da base de fluxos IP.
FONTE: Autor (2019).

A primeira etapa do procedimento de configuração permite definir qual base de fluxos IP (*IP flows datasets*) será utilizada para treinar e testar os algoritmos de aprendizado de máquina. Foi escolhido por motivos de demonstração uma base composta por 700 fluxos IP. Em *Split size* é especificado o tamanho da divisão para

separar a base de fluxos IP em um conjunto de treinamento e outro de teste. O campo *Cross-validation folds* define o número de subconjuntos que serão formados pelo método de *Stratified k-fold cross-validation*. Ao concluir o preenchimento do formulário o usuário prosseguirá para a segunda etapa da configuração que é ilustrada na Figura 11.

FIGURA 11 - Página de configuração dos métodos, algoritmos e características.
FONTE: Autor (2019).

Caso seja necessário, pode ser definido o método de padronização, normalização ou transformação dos dados (*e.g.*, *Min Max Scaler*, *Standard Scaler*, *Normalizer*). Posteriormente são escolhidos os algoritmos de aprendizado de máquina que serão avaliados e as características dos fluxos IP que alimentarão os algoritmos durante o aprendizado. A escolha dos algoritmos disponibilizados pelo sistema foi justificada no capítulo 2 na seção 2.4, porém, vale ressaltar que é possível incluir novos algoritmos de acordo com a necessidade. Nota-se na Figura 11 que múltiplos algoritmos foram marcados para possibilitar a modelagem com as mesmas configurações, porém, com diferentes processos sendo executados. Ao submeter este formulário uma mensagem de "*Training*…" aparecerá para o usuário. A duração do treinamento varia de acordo com as configurações realizadas podendo levar de segundos até mesmo a horas ou dias.

A modelagem consiste em uma etapa onde os algoritmos de aprendizado de máquina são treinados e os hiperparâmetros são otimizados (BUCZAK; GUVEN,

2016). Foram adotados alguns procedimentos comumente encontrados na literatura para a aplicação na base de fluxos IP escolhida pela usuário. Considerando que o usuário inseriu o valor de 30 no campo *Split size*, os dados serão divididos aleatoriamente em um conjunto de treinamento contendo 70% das amostras e outro conjunto destinado para o teste com os 30% restantes. Esta abordagem é importante, pois avaliar a habilidade de um classificador no conjunto de treinamento resultaria em uma pontuação tendenciosa, assim, para dar uma estimativa imparcial sobre a habilidade do modelo o conjunto de teste é utilizado (BROWNLEE, 2017b). Durante esta divisão é aplicada a técnica de estratificação, que preserva em cada conjunto a mesma proporção de amostras de cada classe em relação a base de dados original (SCIKIT-LEARN, 2019).

Os algoritmos de aprendizado de máquina possuem um conjunto de hiperparâmetros, que são parâmetros não aprendidos diretamente durante o treinamento e que podem impactar na predição e performance computacional dos modelos (SCIKIT-LEARN, 2019). Deste modo, foi utilizado o *Grid Search* em conjunto com *Stratified k-fold cross-validation* com o objetivo de buscar a melhor combinação de hiperparâmetros para cada classificador. A técnica de *cross-validation* é aplicada no conjunto de treinamento realizando a divisão por meio do parâmetro *k* estabelecido, sendo usado como exemplo o valor de k=5 estabelecido na Figura 9. Semelhante a divisão inicial, este procedimento separa um conjunto de validação isolado para avaliar a habilidade do modelo preservando a proporção de cada classe durante as divisões. Através do *Grid Search* são geradas inúmeras combinações de um mesmo modelo de acordo com os hiperparâmetros definidos. Cada combinação será treinada em k-1 divisões e avaliada no conjunto de validação. Este procedimento é aplicado por 5 vezes seguidas alternando a ordem das divisões utilizadas. Ao finalizar o *cross-validation* uma combinação foi avaliada e a média dos resultados obtidos é armazenada. Após o término, o *cross-validation* inicia novamente para uma outra combinação até que todos as combinações sejam avaliadas. O melhor resultado obtido determinará qual combinação de hiperparâmetros será utilizada. A Figura 12 ilustra todas as divisões aplicadas na base de fluxos IP pelas diferentes técnicas adotadas.

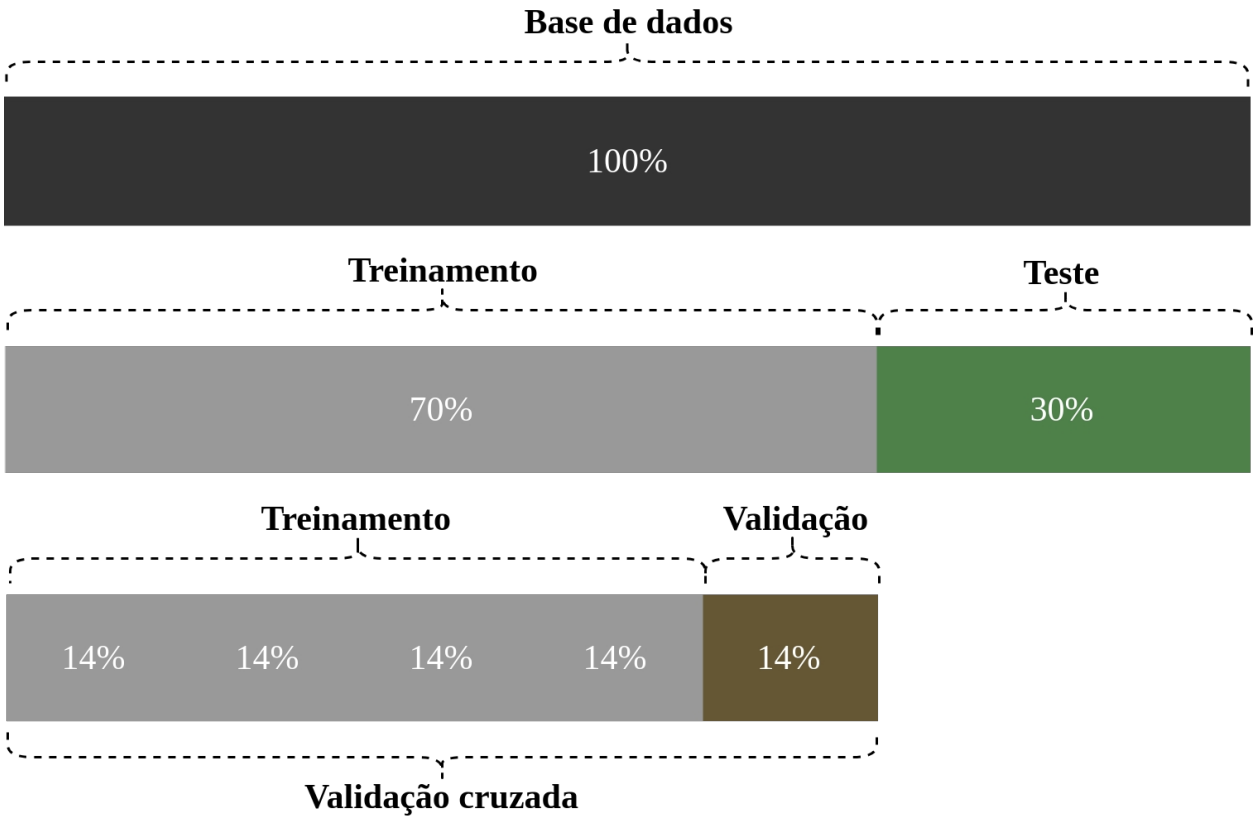
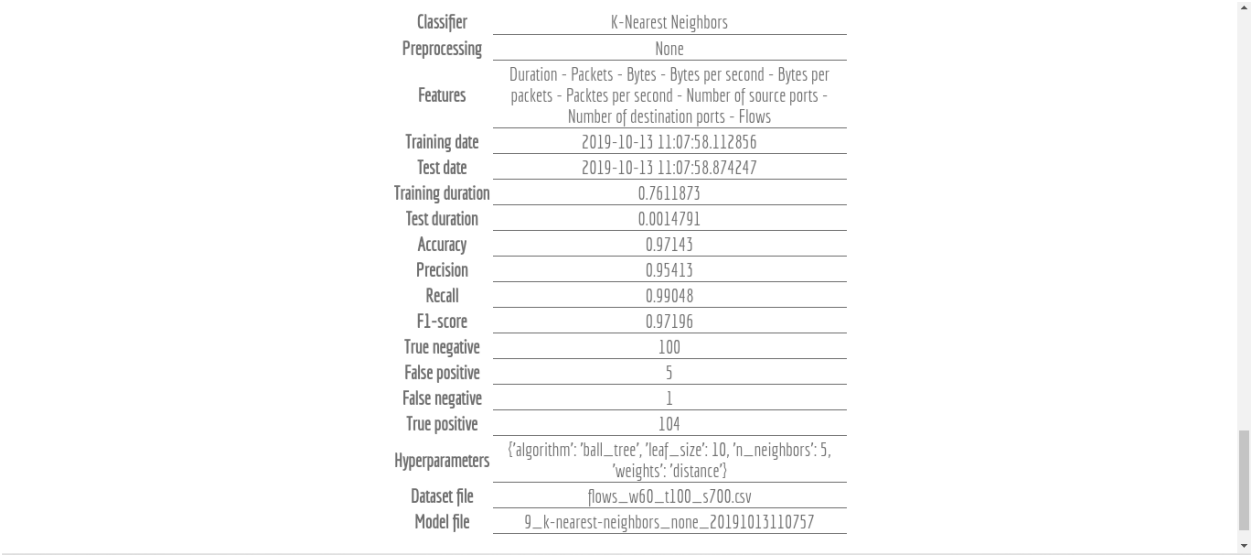


FIGURA 12 - Divisões aplicada na base de fluxos IP.
FONTE: Autor (2019).

Com os classificadores treinados e os hiperparâmetros selecionados é realizado a predição no conjunto de teste que representa os fluxos IP não observados durante o treinamento. Os resultados das predições que consistem nas classes estimadas pelos algoritmos são utilizados como base para as métricas de acurácia, precisão, revocação e F1-score originadas da matriz de confusão (BELOUCH; EL HADAJ; IDHAMMAD, 2018)(BUCZAK; GUVEN, 2016)(HAMID; SUGUMARAN; JOURNAUX, 2016). A Figura 13 mostra a página que apresenta ao usuário os resultados do treinamento e teste dos classificadores.



Classifier	K-Nearest Neighbors
Preprocessing	None
Features	Duration - Packets - Bytes - Bytes per second - Bytes per packets - Packtes per second - Number of source ports - Number of destination ports - Flows
Training date	2019-10-13 11:07:58.112856
Test date	2019-10-13 11:07:58.874247
Training duration	0.7611873
Test duration	0.0014791
Accuracy	0.97143
Precision	0.95413
Recall	0.99048
F1-score	0.97196
True negative	100
False positive	5
False negative	1
True positive	104
Hyperparameters	{'algorithm': 'ball_tree', 'leaf_size': 10, 'n_neighbors': 5, 'weights': 'distance'}
Dataset file	flows_w60_t100_s700.csv
Model file	9_k-nearest-neighbors_none_20191013110757

FIGURA 13 - Página de resultados dos classificadores.
FONTE: Autor (2019).

Na Figura 11 foram marcados dois algoritmos, porém, na Figura 13 apenas o resultado do classificador *K-Nearest Neighbors* é mostrado devido o outro resultado estar localizado mais abaixo da página. Com estas informações o administrador de rede poderá avaliar e escolher qual foi o classificador que obteve o melhor desempenho. Ao clicar no nome do classificador iniciará o retreinamento que repetirá todo o procedimento de modelagem considerando agora toda a base de fluxos IP (100%) como conjunto de treinamento. O resultado é a formação do modelo de aprendizado de máquina que será disponibilizado na opção *Load* da página inicial e utilizado em tempo real para a detecção de ataques.

Determinadas operações dos quatros procedimentos aplicados pelo IPS utilizam como apoio o esquema de banco de dados apresentado na Figura 14. O banco de dados foi desenvolvido através da biblioteca chamada sqlite3[9] que consiste em uma interface para o banco de dados SQLite escrito em C. SQLite provê um "[...] banco de dados baseado em disco que não requer um processo de servidor separado permitindo consultar as tabelas usando uma variante não padrão da linguagem de consulta SQL" (PYTHON SOFTWARE FOUNDATION, 2019). Para facilitar a implementação do esquema e posteriormente as consultas ao banco de dados foi implementado paradigma ORM (*Object-Relational Mapper*) que possibilitou a construção de classes

[9] https://docs.python.org/3.7/library/sqlite3.html

que são mapeadas para as respectivas tabelas do banco de dados.

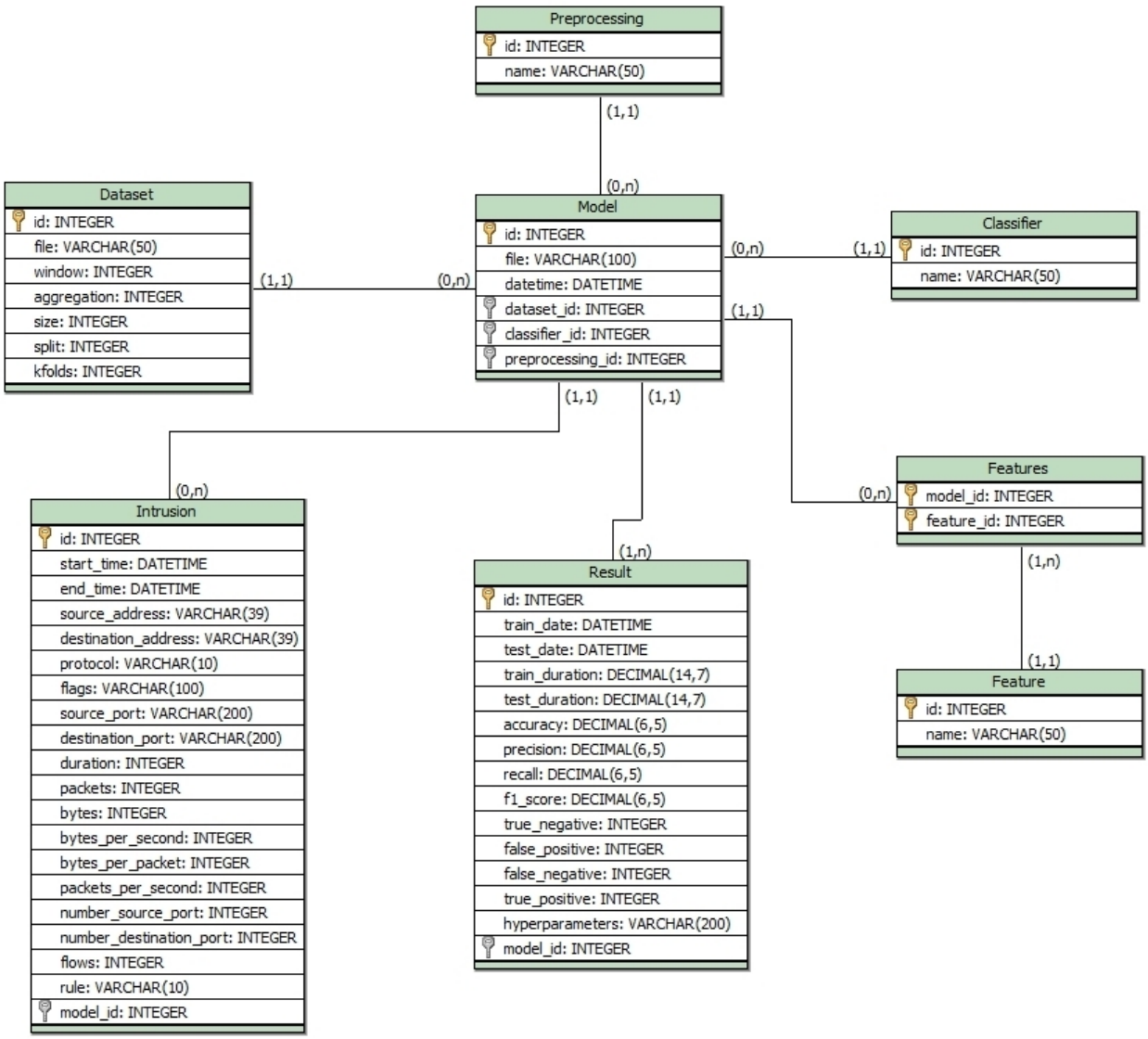


FIGURA 14 - Esquema lógico do banco de dados.
FONTE: Autor (2019).

3.2.1 Núcleo

O núcleo do IPS é representado pelos procedimentos de detecção e mitigação que requerem a existência de um modelo de aprendizado de máquina gerado na etapa de modelagem ou carregado através da opção Load. Mais adiante será observado que muitas funcionalidades utilizadas pelo componente NDC também são utilizadas pelo núcleo do IPS. Devido a aplicação de ortogonalidade e a separação de responsabilidades pode-se aproveitar as funções já desenvolvidas em diferentes partes

do código fonte sem nenhuma alteração. A Figura 15 destaca as principais etapas que são executadas sem a intervenção humana pelo núcleo do IPS.

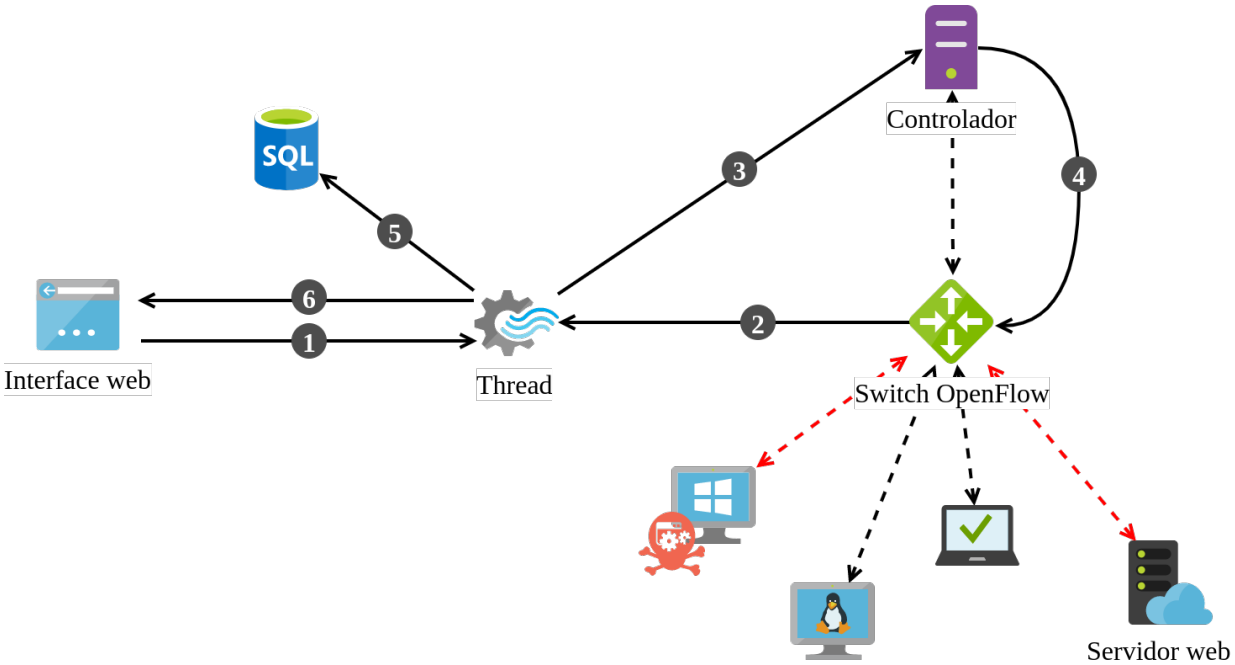


FIGURA 15 - Núcleo do IPS.
FONTE: Autor (2019).

Com o modelo selecionado, na etapa 1 é gerada uma *thread* contento um novo fluxo de controle separado. A *thread* escutará uma porta de rede específica através da ferramenta nfcapd com o objetivo de coletar e pré-processar os fluxos IP exportados pelo *switch* OpenFlow a cada *n* segundos (etapa 2). A funcionalidade de pré-processamento é realizada de forma diferente do que no componente NDC que realiza a agregação e criação das novas características em todo o conjunto de dados primeiro para depois continuar para o próximo processo. No núcleo do IPS apenas um fluxo IP dos dados coletados é separado para ser comparado com os demais com o objetivo de agregá-lo com as amostras iguais e ao término criar as novas características (*e.g.* bytes por segundo). O resultado, é a formação de um fluxo IP agregado que imediatamente será analisado pelo modelo de aprendizado de máquina, que com base no aprendizado obtido durante a modelagem verificará se este fluxo pode ser classificado como ataque ou não. Este procedimento é repetido nos fluxos IP restantes até que não haja mais nenhuma amostra disponível. Desta maneira, a detecção se torna mais eficaz, pois é possível identificar a ocorrência de um ataque nos

primeiros fluxos IP analisados.

Durante o procedimento supracitado se o modelo detectar um ataque a etapa 3 será acionada, gerando uma regra de bloqueio. Através da API REST (*Representational State Transfer*) denominado *Static Flow Entry Pusher* (SFEP) o sistema comunicará o controlador SDN (etapa 4) para inserir a regra diretamente nas tabelas de fluxos do *switch* OpenFlow. No exemplo ilustrado na Figura 15 um computador com sistema operacional Windows realiza um ataque contra o servidor web. Com a conclusão da etapa 4 a comunicação entre estes dispositivos será totalmente interrompida impedindo a continuação do ataque e consequentemente preservando a disponibilidade do servidor web e do controlador SDN. Após a mitigação do ataque, a etapa 5 possui a responsabilidade de armazenar a intrusão no banco de dados para em seguida na etapa 6, abrir um socket de rede com a interface web do usuário para comunicar o número de intrusões detectadas.

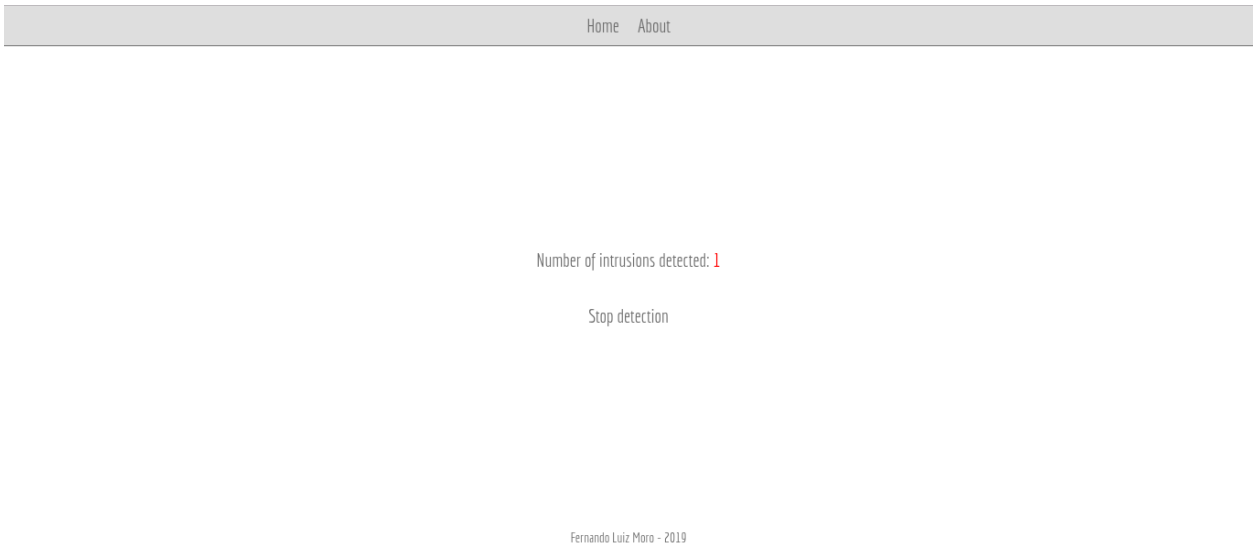


FIGURA 16 - Página de detecção em tempo real.
FONTE: Autor (2019).

A informação apresentada na Figura 16 é um breve sumário que vai sendo atualizado automaticamente à medida que os ataques vão sendo detectados. Para melhorar a compreensão sobre o ataque ocorrido o administrador de rede poderá clicar no número apresentado na tela para abrir uma nova página com os detalhes. Caso o usuário deseje encerrar o núcleo do sistema ele poderá clicar no *Stop detection* que ocasionará a interrupção da *thread* e o redirecionamento para a página inicial.

FIGURA 17 - Página das intrusões detectadas.
FONTE: Autor (2019).

Os detalhes apresentados na Figura 17 mostram as informações pertinentes sobre a intrusão como o dispositivo que originou o ataque (*Source address*), o dispositivo alvo (*Destination address*), a duração do ataque (*Duration*), número de bytes transmitidos (*Bytes*), número de pacotes enviados (*Packets*), entre outras características. Caso seja observado que o modelo classificou incorretamente um ataque, o usuário poderá clicar no nome da regra de bloqueio no campo *Remove rule* para removê-la das tabelas de fluxos do *switch* OpenFlow do qual o dispositivo atacante está conectado. O efeito do tratamento de um falso alarme consiste no restabelecimento da comunicação entre os dispositivos que haviam sido bloqueados.

4 RESULTADOS

Os resultados e os experimentos apresentados a seguir foram executados em uma máquina com sistema operacional Ubuntu 18.04.3 LTS, com processador Intel Core i7-4510U de 4 núcleos com frequência de 2.00 GHz, 8 GB de memória RAM, 1 TB de disco rígido e placa de vídeo ADM Radeon M265.

4.1 Base de fluxos IP agregada

Um dos principais pontos para a obtenção de bons resultados no aprendizado de máquina é uma base de dados com amostras representativas do problema que se pretende resolver. No entanto, as bases de dados utilizadas amplamente na literatura, tais como a DARPA 1998, DARPA 1999 e o KDD 1999 possuem tráfego sintetizado e com grande número de redundâncias causando viés e consequentemente afetando o resultado dos algoritmos (BUCZAK; GUVEN, 2016). Alternativamente, o NSL-KDD foi criado com o objetivo de resolver os problemas apresentados pelos seus antecessores, porém, o tráfego de rede normal e os ataques gerados não representam mais o cenário atual das redes de computadores (BELOUCH; EL HADAJ; IDHAMMAD, 2018 *apud* TAVALLAEE *et al.* 2009).

Para contornar esta questão, considerou-se neste trabalho uma base de dados de rede com 95 GB fornecida pelo laboratório do Grupo de Teleinformática e Automação da Universidade Federal do Rio de Janeiro, contendo tráfego normal e ataques de rede reais divididos em duas principais classes: *Denial of Service* e *Probe* (LOBATO; LOPEZ; DUARTE, 2016). Os ataques que compõem a primeira classe são, ICMP *flood*, *land*, *nestea*, *smurf*, SYN *flood*, *teardrop* e UDP *flood*. Nos ataques

destinados a sondagem de rede e sistemas foram utilizados o TCP SYN *scan*, TCP *connect scan*, SCTP INIT *scan*, *Null scan*, FIN *scan*, Xmas *scan*, TCP ACK *scan*, TCP *Window scan* e TCP Maimon *scan*.

O componente NDC possibilitou a criação da base de fluxos IP a partir da base de dados de rede descrita acima. Foi definida uma janela temporal de 60 segundos para converter os arquivos pcap em nfcapd para então, gerar o arquivo CSV com os fluxos IP não tratados. Na etapa de pré-processamento os fluxos foram formatados e agregados considerando o intervalo de tempo definido, o IP de origem, o IP de destino, protocolo e um limite de agregação de 100 amostras. Estes procedimentos foram aplicados para cada arquivo pcap da base de dados gerando uma série de arquivos CSV com os fluxos IP pré-processados. O último passo, foi unir todos estes arquivos para formar a base de dados final para ser utilizado pelo sistema durante o aprendizado dos algoritmos. A Tabela 4 apresenta a quantidade de fluxos IP antes e depois da aplicação do método de agregação.

TABELA 4 - Fluxos IP em relação aos fluxos IP agregados.

Classe da amostra	Tipo de tráfego	Fluxos IP	Fluxos IP agregados	Total
Negativa (0)	Normal	68872	27856	27856
Positiva (1)	*Land*	73	66	27899
	Teardrop	121	62	
	Probe	555000	5561	
	Smurf	555000	5551	
	ICM *flood*	555000	5551	
	UDP *flood*	555000	5553	
	SYN *flood*	555000	5555	

FONTE: Autor (2019).

Conforme a Tabela 4 a quantidade de fluxos IP com tráfego de rede normal era de 68872 e após a agregação foram contabilizadas 27856, uma redução de 59.55%. Optou-se pela construção de uma base de dados balanceada para evitar que os algoritmos de aprendizado de máquina fossem tendenciosos em relação a uma classe específica. Portanto, de modo semelhante os fluxos provenientes de diversos ataques foram agregados resultando em 27899 amostras. Observa-se que a grande maioria dos tipos de ataques são variações de DoS e apenas um tipo contém ataques de sondagem.

Ressalta-se que a base de dados de rede já estava neste padrão. Da listagem dos ataques supracitados houveram poucos casos que não puderam ser considerados (*e.g. nestea*), pois durante a conversão apresentaram erros nas ferramentas de terceiros.

Os ataques DoS consistem em milhares de requisições enviadas com o objetivo de esgotar os recursos do alvo, sendo assim, os fluxos gerados possuem propriedades com pouca variação. Por conta deste fator, os critérios estabelecidos para a agregação são facilmente alcançados gerando poucos fluxos IP agregados. Para evitar o desbalanceamento em relação à classe negativa foram coletadas quando disponíveis, amostras consideravelmente maiores da classe positiva. Na Tabela 4, os fluxos agregados do ataque DoS SYN *flood* resultaram em 5555 amostras, porém, 555000 amostras foram utilizados como base para este processo. Para melhor compreender o impacto do método de agregação, a Figura 18 apresenta um gráfico comparando a quantidade de fluxos IP em relação aos fluxos IP agregados.

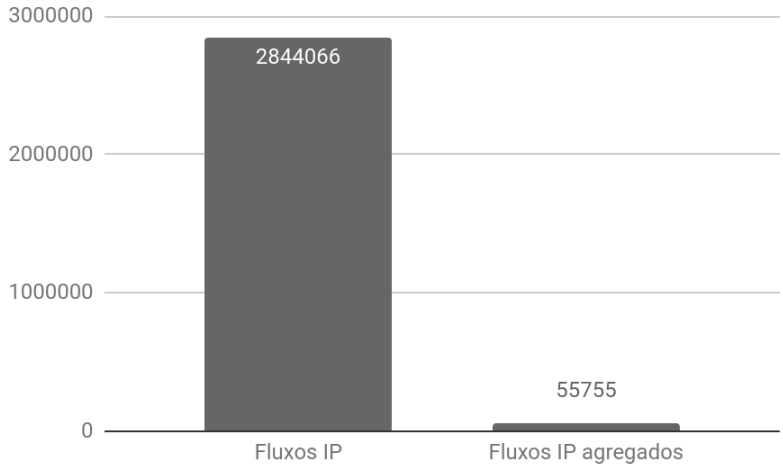


FIGURA 18 - Quantidade total de fluxos IP em relação aos fluxos IP agregado.
FONTE: Autor (2019).

O método de agregação reduziu em 98.04% a quantidade de fluxos IP que os algoritmos de aprendizado de máquina deveriam processar. A base de fluxos IP final disponibilizada para utilização no sistema resultou em um total de 55755 amostras com propriedades mais discriminativas para a identificação de padrões. Para possibilitar a execução rápida de testes das funcionalidades envolvendo o aprendizado de máquina foi criada uma outra base de dados com apenas 700 amostras. Esta base de

dados menor foi utilizada durante a explicação do sistema proposto.

4.2 Análise dos algoritmos de aprendizado de máquina

O sistema através do componente IPS antes de avaliar os algoritmos de aprendizado de máquina executa o procedimento de configuração, onde na primeira etapa foi definido a escolha da base com 55755 fluxos IP, a divisão de 70% das amostras para o treinamento e os 30% restantes para o teste, o método de Grid Search e *Stratified* 5-*fold cross-validation*. O valor 5 e 10 para o parâmetro *k* são comumente utilizados na literatura (HAMID; SUGUMARAN; JOURNAUX, 2016)(LOBATO; LOPEZ; DUARTE, 2016)(NAJAFABADI *et al.*, 2015). Neste trabalho porém, optou-se pela utilização de *k*=5, pois foram encontrados resultados semelhantes através de testes realizados na base de dados com *k*=10, contudo, para o algoritmo *Decision Tree* por exemplo, houve um aumento de 85.40% no tempo de treinamento. Na segunda etapa nenhum método de padronização, normalização ou transformação dos dados foi escolhido, para possibilitar a avaliação da performance dos algoritmos na base de fluxos IP na sua escala original. Dos classificadores disponibilizados pelo sistema foram selecionados todos com exceção do *Multi-Layer Perceptron* devido às limitações de hardware. Para o treinamento foram consideradas todas as características dos fluxos IP e as métricas de avaliação utilizadas podem ser observadas na Tabela 5.

TABELA 5 - Métricas de avaliação.

Métrica	Descrição	Fórmula
Acurácia	Proporção das classificações corretas em relação a todas as amostras da base de dados.	$\frac{VP + VN}{VP + FP + VN + FN}$
Precisão	Proporção das classificações positivas corretas em relação a todas as classificações positivas.	$\frac{VP}{VP + FP}$
Revocação	Proporção das classificações positivas corretas em relação a todas as amostras positivas.	$\frac{VP}{VP + FN}$
F1-score	Média harmônica entre a precisão e revocação.	$\frac{2 * (\text{Precisão} * \text{Revocação})}{\text{Precisão} + \text{Revocação}}$

FONTE: Buczak e Guven (2016) e Hamid, Sugumaran e Journaux (2016).

Os algoritmos de aprendizado de máquina utilizados neste trabalho realizam

a classificação binária, ou seja, cada fluxo IP é categorizado como classe negativa (0 - tráfego normal) ou classe positiva (1 - tráfego de ataque). As variáveis que compõem as fórmulas derivam da matriz de confusão com base nas duas classes definidas. O significado para cada uma delas pode ser compreendido a seguir (CARVALHO, 2018):

- VP (Verdadeira Positivo): amostra positiva que foi classificada como positiva.
- FP (Falso Positivo): amostra negativa que foi classificada como positiva.
- VN (Verdadeira Negativo): amostra negativa que foi classificada como negativa.
- FN (Falso Negativo): amostra positiva que foi classificada como negativa.

Com base nestas premissas, a ausência de valor nas variáveis FP e FN representa uma classificação ideal, onde o algoritmo conseguiu distinguir corretamente todas as amostras. Os resultados do procedimento de modelagem são apresentados na Tabela 6 e Tabela 7.

TABELA 6 - Resultados provenientes das métricas de avaliação.

Algoritmos	Acurácia	Precisão	Revocação	F1-score	Tempo de treinamento (segundos)	Tempo de teste (segundos)
Decision Tree	99.93%	99.96%	99.90%	99.93%	184.27	0.05
Gaussian Naive Bayes	51.25%	50.65%	100.00%	67.24%	2.08	0.04
K-Nearest Neighbors	99.87%	99.92%	99.82%	99.87%	138.51	1.13
Support Vector Machine	99.67%	100.00%	99.33%	99.66%	811.20	2.95

FONTE: Autor (2019).

TABELA 7 - Resultados provenientes da matriz de confusão.

Algoritmos	Verdadeiro Positivo	Falso Positivo	Verdadeira Negativo	Falso Negativo
Decision Tree	8362	3	8354	8
Gaussian Naive Bayes	8370	8155	202	0
K-Nearest Neighbors	8355	7	8350	15
Support Vector Machine	8314	0	8357	56

FONTE: Autor (2019).

Conforme os resultados descritos na Tabela 6, todos os classificadores alcançaram um *F1-score* acima de 99%, com exceção do *Gaussian Naive Bayes*. Ao

investigar a matriz de confusão na Tabela 7, nota-se que este algoritmo não apresentou nenhum FN, portanto, todos os fluxos de ataque foram devidamente classificados como tal resultando em uma revocação de 100.00%. Porém, houve 8155 amostras contendo tráfego normal que foram erroneamente rotuladas com a classe positiva resultando em uma baixa precisão de 50.65%. O *Support Vector Machine* demonstrou um comportamento oposto ao GNB não rotulando nenhum FP, mas, classificando 56 amostras como FN. Estes dados indicam que este classificador foi capaz de identificar com exatidão o padrão do tráfego normal, porém, falhou em detectar alguns fluxos originados de ataques. Por conta destes fatores o SVM atingiu uma precisão de 100% e uma revocação de 99.33%.

O *K-Nearest Neighbors* em comparação aos dois algoritmos analisados, obteve de modo geral um melhor resultado nas métricas de avaliação, com destaque para o *F1-score* de 99.87% que superou os demais. Mesmo assim, ainda observa-se que o KNN deixou de classificar 15 amostras positivas rotulando elas como amostras negativas. Em um ambiente real de monitoramento a não detecção de fluxos IP atacantes pode ser extremamente prejudicial para a infraestrutura de rede, sendo assim, busca-se um modelo que minimize a taxa de FN e maximize a taxa de VP. Na Tabela 6 o GNB aparenta ser o modelo ideal, porém, devido ao alto número de FP muitos falsos alarmes seriam gerados o que culminaria no bloqueio dos dispositivos que estão enviando tráfego normal. Desta maneira, o *Decision Tree* seria a melhor indicação pois foi o segundo que menos contabilizou FN e somente gerou 3 falsos alarmes, atingindo um F1-Score de 99.93%. Além disso, este classificador possui um ótimo tempo de processamento levando 184.27 segundos para o treinamento na base de fluxos IP e 0.05 segundos para realizar a predição no conjunto de teste. Estes fatores são essenciais para a atuação do sistema e a detecção em tempo real.

Os resultados das métricas de avaliação para a maioria dos algoritmos de aprendizado de máquina foram otimistas, o que pode indicar um possível overfitting. Este termo se refere a um modelo que aprendeu todos os detalhes e ruídos dos dados de treinamento de modo que isso, impacte negativamente a habilidade de generalização em novos dados (BROWNLEE, 2016). Ao analisar os tipos de ataque da

classe positiva na Tabela 4, compreende-se que há uma quantidade de fluxos IP elevada em relação às amostras do tráfego normal, porém, com características semelhantes. Ao aplicar o método de agregação as características que são criadas com base na quantidade de fluxos enviados se tornam mais distinguíveis das demais. Assim, o processo de aprendizado dos modelos é facilitado justificando os valores alcançados. Para esclarecer a importância da agregação, foram realizados testes com uma base de fluxos IP não agregada onde observou-se resultados piores entre 89.70% a 99.18% na métrica de *F1-score*.

Apesar do GNB ter alcançado resultados abaixo do esperado, ele foi considerado o modelo com o melhor tempo de treinamento sendo 66.59 vezes mais rápido que o segundo colocado. O SVM demorou 811.20 segundos para realizar o treinamento e 2.95 segundos para a predição, sendo considerado o algoritmo mais lento em termos de processamento. O KNN e o DT obtiveram resultados semelhante no tempo de treinamento. Porém, no tempo de testes o DT demorou 0.05 segundos em relação ao 1.13 segundos do KNN. Vale ressaltar que o tempo de treinamento e teste consideraram as etapas de aprendizado e busca dos hiperparâmetros, no entanto, cada classificador possui a sua própria quantidades de hiperparâmetros. Como exemplo, nos experimentos foram definidos 4 hiperparâmetros para o *K-Nearest Neighbors* e apenas 1 para o *Gaussian Naive Bayes*, cada qual, com múltiplas opções (*e.g. n_neighbors*: [5, 10, 15]). Além da complexidade computacional de cada modelo, as inúmeras combinações geradas pelo *Grid Search* influenciam diretamente no tempo de processamento. Portanto, os dados gerados servem com uma estimativa em relação ao tempo. Os melhores hiperparâmetros de cada algoritmo avaliado são descritos na Tabela 8.

TABELA 8 - Hiperparâmetros dos modelos.

Algoritmos	Hiperparâmetros
Decision Tree	{'criterion': 'entropy', 'max_depth': 9, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'}
Gaussian Naive Bayes	{'var_smoothing': 1e-05}
K-Nearest Neighbors	{'algorithm': 'ball_tree', 'leaf_size': 10, 'n_neighbors': 5, 'weights': 'distance'}
Support Vector Machine	{'C': 10.0, 'cache_size': 5000.0, 'gamma': 0.001, 'kernel': 'rbf'}

FONTE: Autor (2019).

O objetivo de apresentar os dados da Tabela 8 consiste em permitir a replicação dos experimentos realizados neste trabalho. Para a melhor compreensão do funcionamento dos hiperparâmetros consulte a documentação em (SCIKIT-LEARN, 2019).

4.3 Experimentos

Para validar o funcionamento em tempo real do componente IPS foi simulado uma rede definida por software virtual através da ferramenta Mininet[10]. A Figura 19 mostra a topologia de rede criada para os experimentos que é composta por 7 *hosts*, 4 *switches* OpenFlow e um controlador SDN chamado Floodlight[11] vinculado no endereço 127.0.0.1:6653.

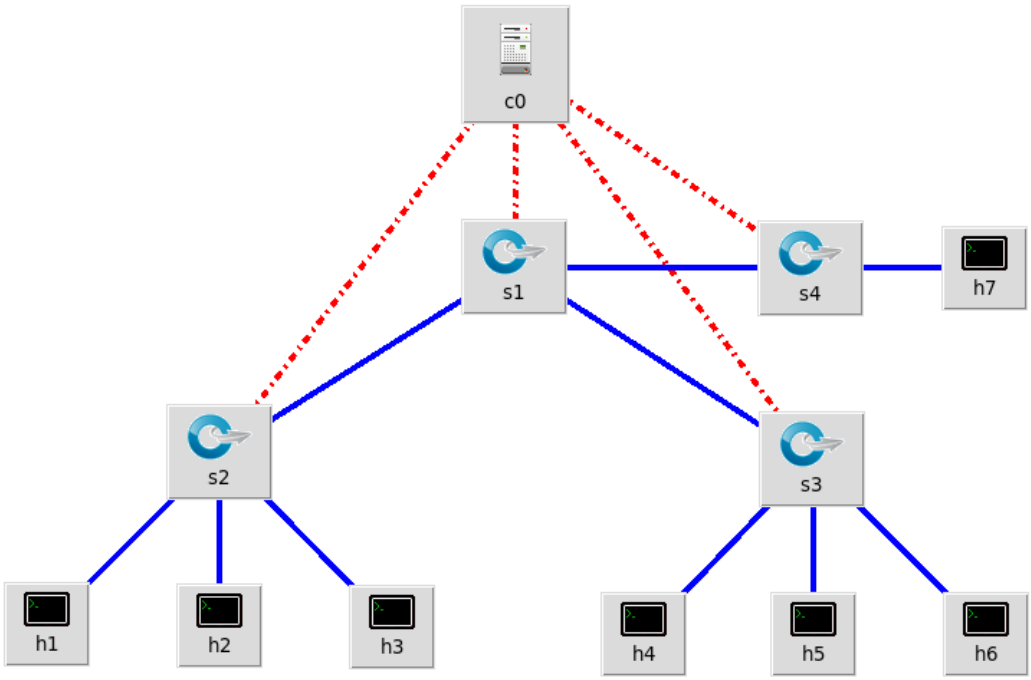


FIGURA 19 - Topologia de rede usada nos experimentos.
FONTE: Autor (2019).

As sub-redes existentes na topologia são interligadas em um *switch* OpenFlow central (s1) habilitado para exportar os fluxos IP através do protocolo

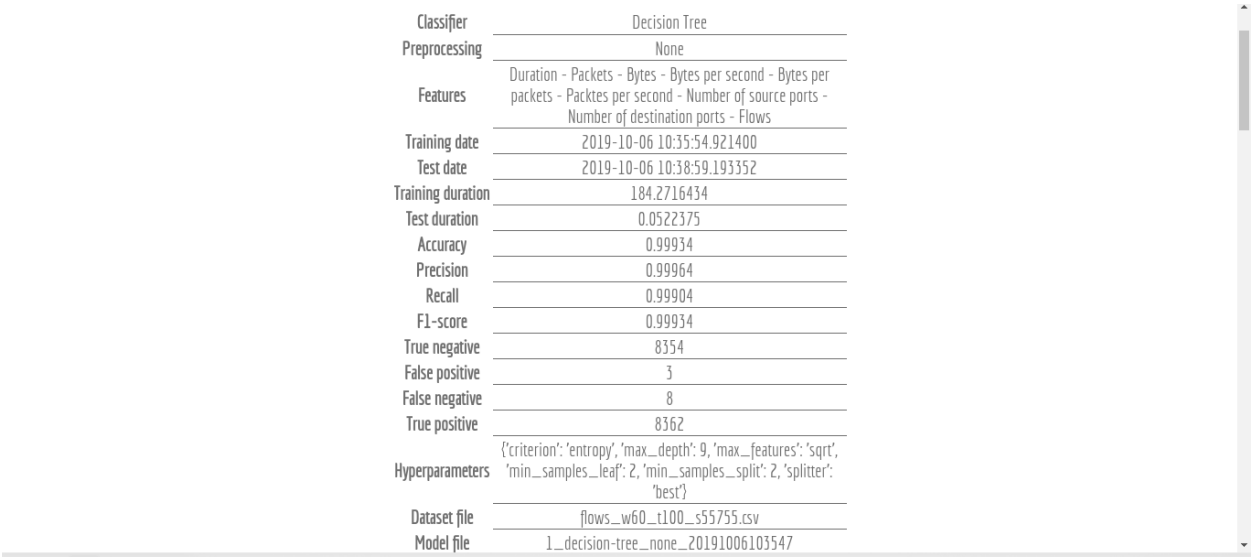
[10] http://mininet.org/
[11] http://www.projectfloodlight.org/floodlight/

NetFlow a cada 60 segundos para o endereço 127.0.0.1:7777. A definição da janela temporal utilizada para a obtenção da fonte de dados é crucial, pois por padrão um fluxo IP pode permanecer ativo por um longo período até o seu término. Os fluxos podem se manter ativos até 30 minutos na *cache* de um roteador, e ocorrendo este período, eles serão expirados e exportados (COUTO, 2012). Porém, aguardar um longo período durante a ocorrência de um ataque pode resultar em sérios prejuízos para a rede. Segundo Brauckhoff *et al.* (2012) diferentes janelas de tempo são encontradas na literatura como 900, 600, 300 e 60 segundos. Devido a constante comunicação entre os *switches* OpenFlow e o controlador, aguardar um período de tempo superior a 60 segundos pode acarretar a indisponibilidade destes elementos.

A topologia de rede foi desenvolvida com a intenção de ter uma sub-rede com um servidor HTTP (*HyperText Transfer Protocol*) recebendo as requisições dos demais *host* e sendo alvo dos ataques durante os experimentos. Portanto, através do módulo http escrito em Python foi implementado no *host* 7 portador do IP 192.168.0.17 um servidor HTTP rodando na porta 80. O *host* 2 (192.168.0.12) foi configurado para enviar as requisições UDP, o *host* 4 (192.168.0.14) as requisições HTTP e *host* 6 (192.168.0.16) as requisições TCP. Para ambos os *hosts*, a quantidade de pacotes, bytes, a porta de origem utilizada e o tempo para a realização das requisições são gerados aleatoriamente de acordo com o padrão da base de fluxos IP que foi utilizada para o treinamento do modelo. O intuito destes procedimentos foi de simular o ambiente de rede sobre o qual os dados originais foram coletados e avaliar se o modelo era capaz de distinguir os fluxos concorrentes de vários ataques.

Devido às limitações de hardware, optou-se por replicar os ataques de rede em uma escala menor e com o número de fluxos concorrentes reduzidos para evitar sobrecarga no processamento da máquina utilizada para os experimentos. Contudo, os parâmetros utilizados para a execução dos ataques seguem as abordagens comumente adotadas em cenários reais. O propósito desta metodologia consiste em avaliar a capacidade de detecção e mitigação do componente IPS para que com isso, seja analisado se a disponibilidade do controlador SDN e dos dispositivos da infraestrutura de rede foi preservada. Com base nos testes e análises dos algoritmos de aprendizado

de máquina verificou-se que o *Decision Tree* é o mais adequado para ser utilizado pelo sistema. O resultado produzido no procedimento de modelagem para este algoritmo pode ser observado na Figura 20.



Classifier	Decision Tree
Preprocessing	None
Features	Duration - Packets - Bytes - Bytes per second - Bytes per packets - Packtes per second - Number of source ports - Number of destination ports - Flows
Training date	2019-10-06 10:35:54.921400
Test date	2019-10-06 10:38:59.193352
Training duration	184.2716434
Test duration	0.0522375
Accuracy	0.99934
Precision	0.99964
Recall	0.99904
F1-score	0.99934
True negative	8354
False positive	3
False negative	8
True positive	8362
Hyperparameters	{'criterion': 'entropy', 'max_depth': 9, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'}
Dataset file	flows_w60_t100_s55755.csv
Model file	1_decision-tree_none_20191006103547

FIGURA 20 - Resultados do modelo selecionado.
FONTE: Autor (2019).

Ao selecionar o modelo, o sistema através do núcleo do IPS inicia os procedimentos de monitoramento em tempo real aguardando o recebimento dos fluxos IP gerados pela comunicação dos dispositivos de rede. Para este fim, o controlador SDN, os *switches* OpenFlow e a geração do tráfego de rede normal foram iniciados e durante 6 minutos não foram lançados nenhum tipo ataque para avaliar se o IPS geraria algum falso alarme. Neste período foi observado que nenhum fluxo normal foi classificado incorretamente pelo modelo indicando que o aprendizado em relação ao perfil do tráfego normal da rede foi aprendido corretamente.

4.3.1 Ataque DoS

Neste experimento o *host* 1 (192.168.0.1) foi responsável por gerar através da ferramenta hping3, um ataque DoS do tipo SYN *flood* com as portas de origem aleatórias. Foi configurado o envio de 100 pacotes por segundos com 100 bytes cada para a porta 80 do *host* 7. O ataque foi lançado quando restavam 30 segundos para finalizar a janela de exportação e devido ao grande volume gerado observou-se logo

nos primeiros segundos, a perda dos pacotes e aumento da latência de rede dos demais hosts conforme mostrado na Figura 21.

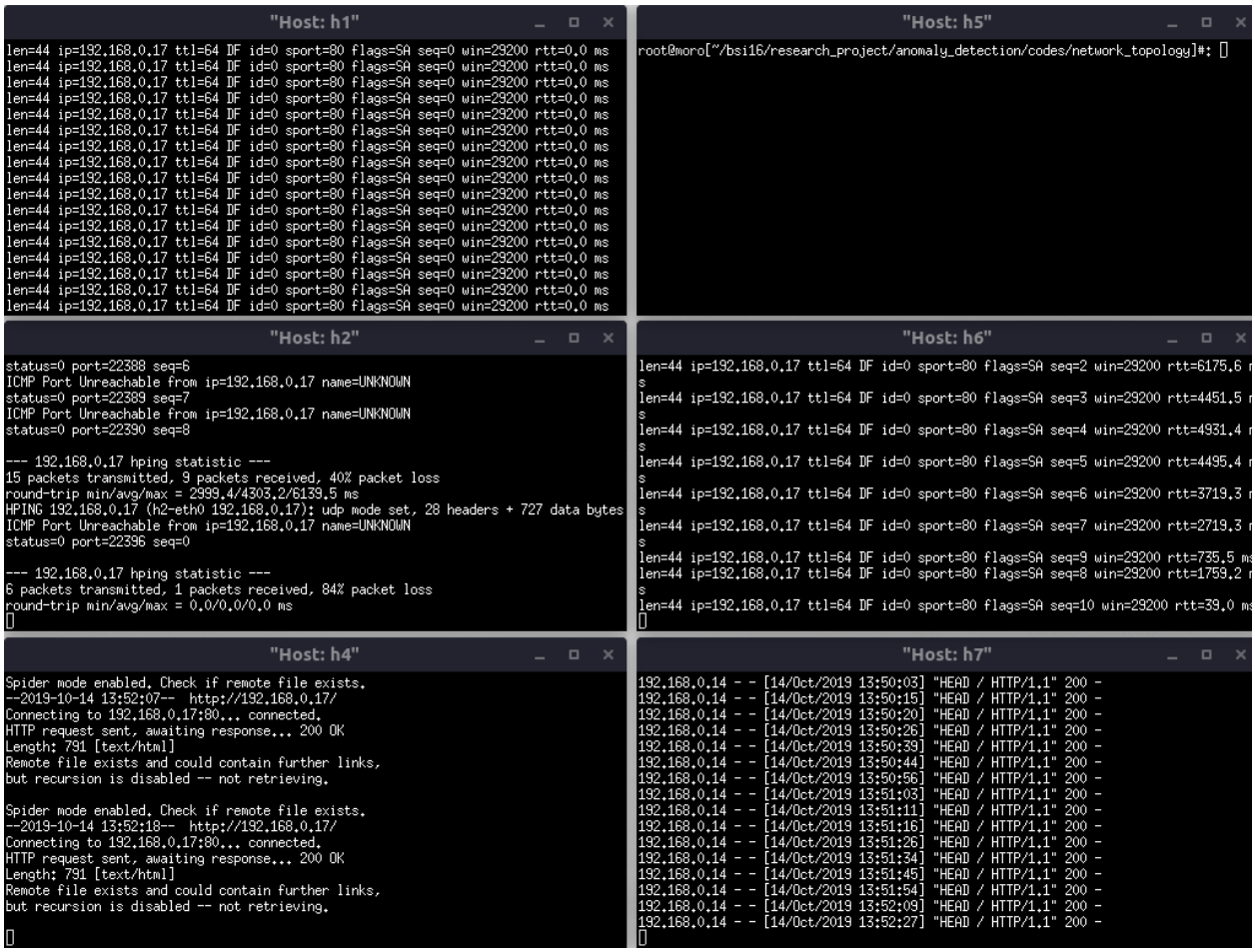


FIGURA 21 - Comunicação entre os *hosts* durante o ataque DoS.
FONTE: Autor (2019).

Nota-se que o *host* 2 perdeu 84% dos pacotes UDP enviados enquanto que o host 6 aumentou de 7.0 milisegundos a 6175.6 milisegundos a latência de rede durante as requisições TCP. O controlador SDN antes deste procedimento havia alocado 317.19 MB de memória, consumindo apenas entre 90.00 a 100.00 MB. Contudo, durante a ocorrência do ataque DoS todo este espaço foi imediatamente consumido forçando o controlar a alocar mais espaço para evitar a indisponibilidade da rede. A Figura 22 mostra o ápice de espaço alocado na interface do controlador Floodlight.

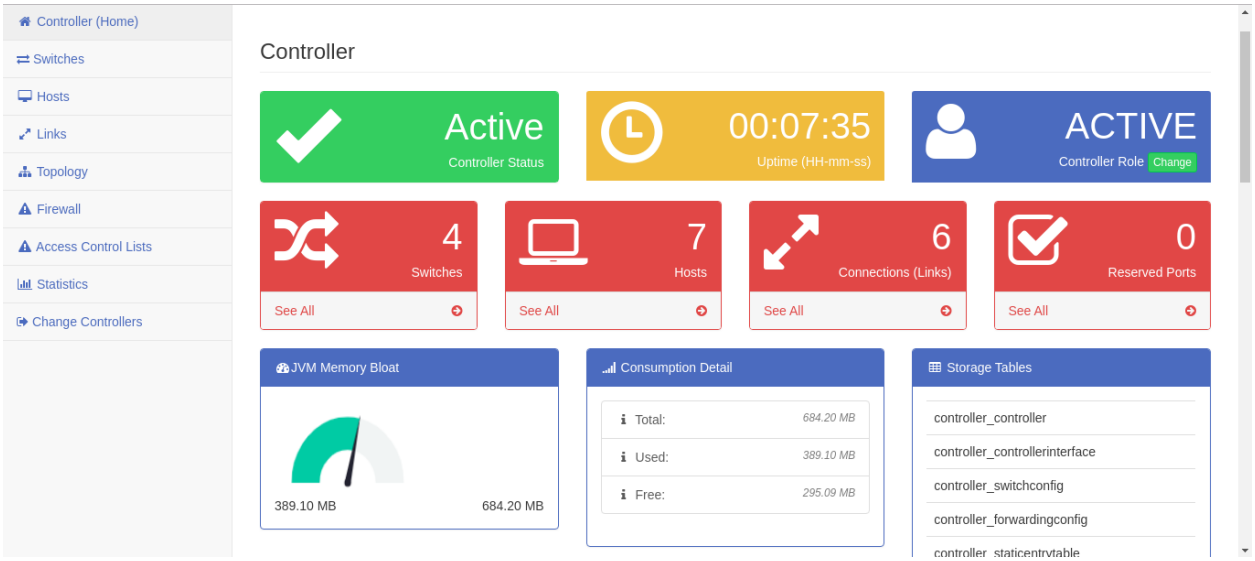


FIGURA 22 - Consumo de memória do controlador SDN durante o ataque DoS.
FONTE: Autor (2019).

Como pode ser visto na Figura 22, o consumo de memória estava em 389.10 MB, no entanto, este valor chegou a alcançar todo os 684.20 MB alocados. Ao encerrar o tempo da coleta dos dados o *switch* s1 exportou os fluxos IP para o IPS, que primeiramente realizou a formatação para o padrão utilizado pelo modelo levando para este procedimento, um total de 5.0133719 segundos. Posteriormente, iniciou-se a agregação dinâmica em que a medida que um fluxo era agregado ele imediatamente era analisado para verificar se um ataque estava ocorrendo. O primeiro fluxo IP agregado originado do ataque foi classificado pelo modelo em apenas 0.0004632 segundos desencadeando automaticamente o processo de mitigação. O IPS em 0.0403457 segundos descobriu em qual *switch* o dispositivo atacante estava vinculado e em 0.1829739 enviou a regra de bloqueio para o controlador que através do protocolo OpenFlow inseriu ela diretamente nas tabelas de fluxos do *switch* s2. Após, o núcleo do IPS comunicou o administrador de rede sobre o ocorrido e disponibilizou na interface web o ataque detectado conforme demonstrado na Figura 23.

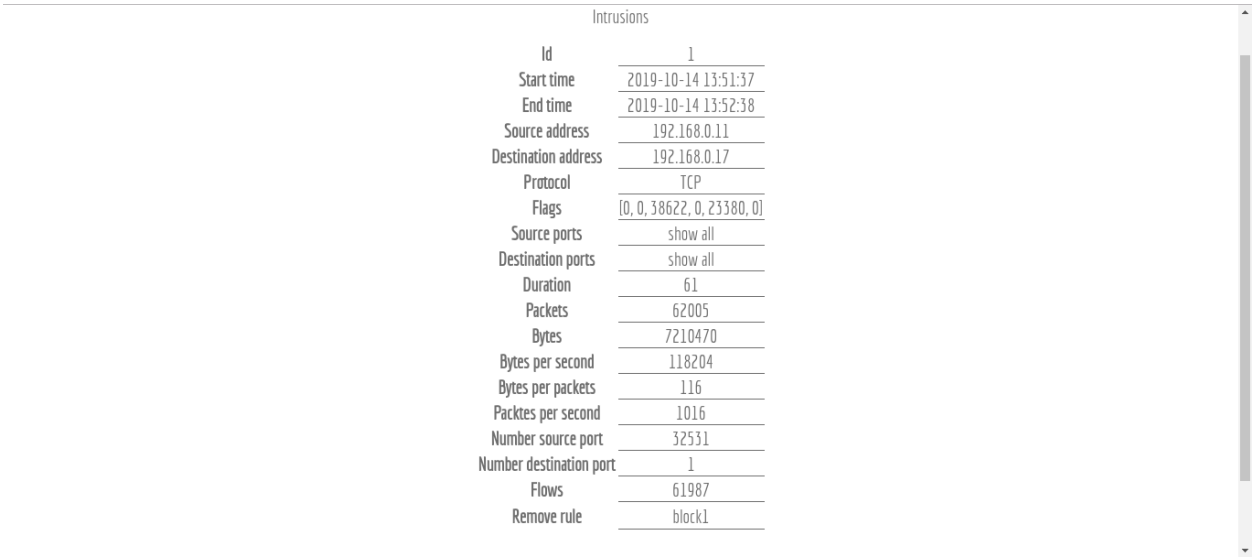


FIGURA 23 - Informações sobre o ataque DoS detectado.
FONTE: Autor (2019).

A Figura 23 mostra que no ataque foram enviados 62005 pacotes com um total de 7210470 bytes, 61987 fluxos originados de 32531 portas diferentes. Foi informado para o administrador de rede que o ataque durou 61 segundos, porém, conforme frisado anteriormente o ataque foi lançado faltando 30 segundos e ao fechar o tempo para exportação ele automaticamente foi bloqueado. Devido a grande quantidade de fluxos gerados uma boa parte das amostras foram exportadas somente na outra janela temporal, o que justifica a duração informada. Como efeito das ações realizadas pelo IPS não houve mais perdas de pacotes, a latência de rede e o consumo de memória do controlador foram restabelecidas preservando a disponibilidade da rede. Vale ressaltar que em testes realizados sem a atuação deste sistema o ataque não foi interrompido, o que resultou na indisponibilidade de toda a infraestrutura de rede em poucos minutos.

4.3.2 Ataque de port scan

Para avaliar a sensibilidade do sistema em detectar a ocorrência de um ataque com curta duração e pouco volume, foi configurado o lançamento de um TCP SYN *port scan* através do host 5 vinculado ao *switch* 3. O objetivo deste ataque é varrer as portas de rede reservadas (1 à 1023) do servidor HTTP através do envio de um pacote com 20 bytes para cada uma das portas. A Figura 24 mostra o estado dos

hosts após o término do ataque que teve a duração de apenas 1 segundo.

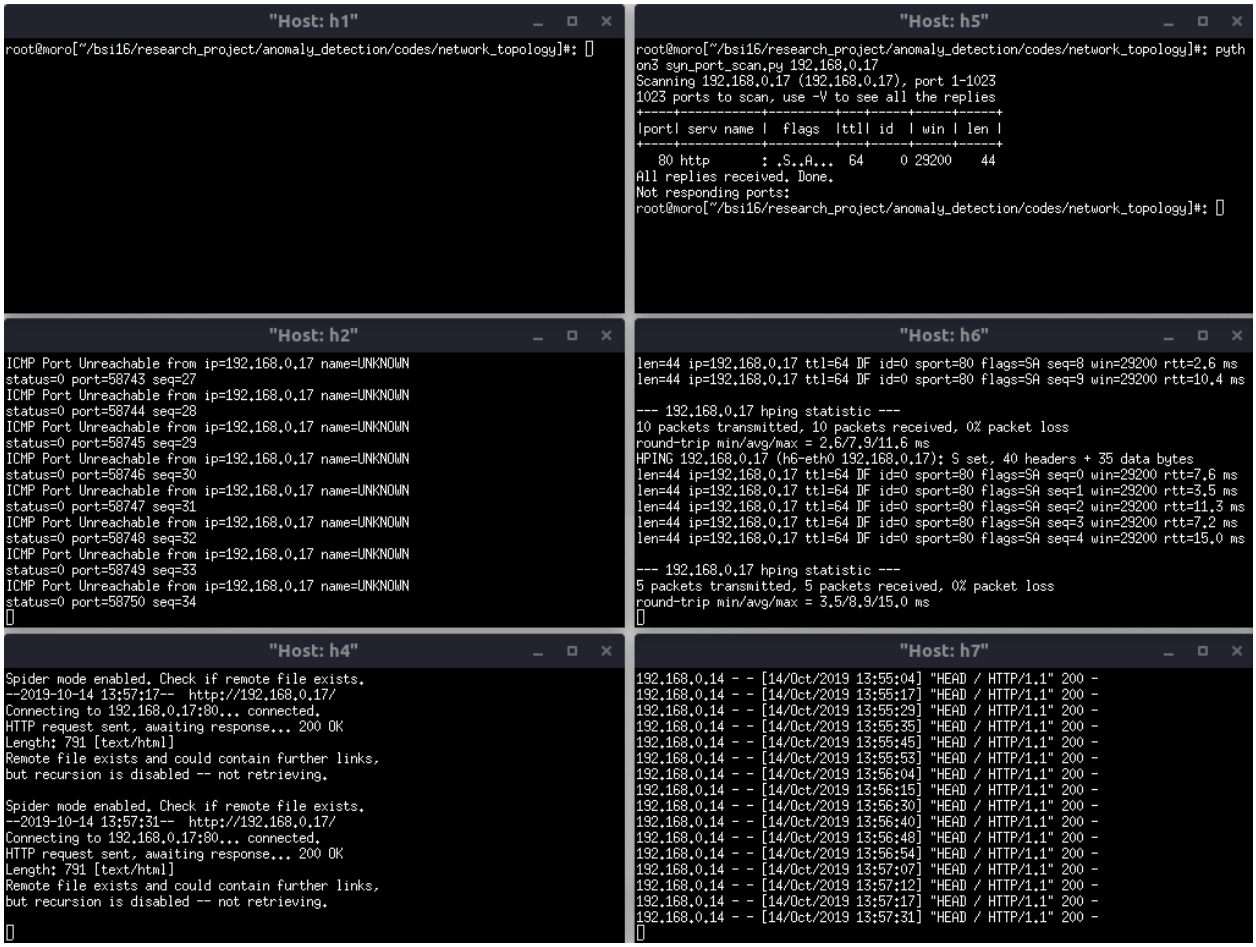


FIGURA 24 - Comunicação entre os *hosts* durante o ataque de *port scan*.
FONTE: Autor (2019).

No terminal do *host* 5 na Figura 24, percebe-se que o único serviço detectado por esta sondagem de rede foi justamente o servidor HTTP que estava sendo executado pelo *host* 7. O consumo de memória do controlador SDN em relação a este ataque foi inexpressivo, não afetando nenhuma comunicação entre os *hosts* e o servidor HTTP. Contudo, a detecção deste tipo de ataque é crucial, uma vez que comumente os atacantes fazem o uso desta técnica para identificar os serviços que estão ativos para em seguida lançar ataques como o DoS. Ao finalizar a janela temporal em que este ataque foi lançado os dados foram analisados pelo modelo de aprendizado de máquina que com exatidão detectou e automaticamente bloqueou o ataque. O resultado disponibilizado para o administrador de rede pode ser observado na Figura 25.

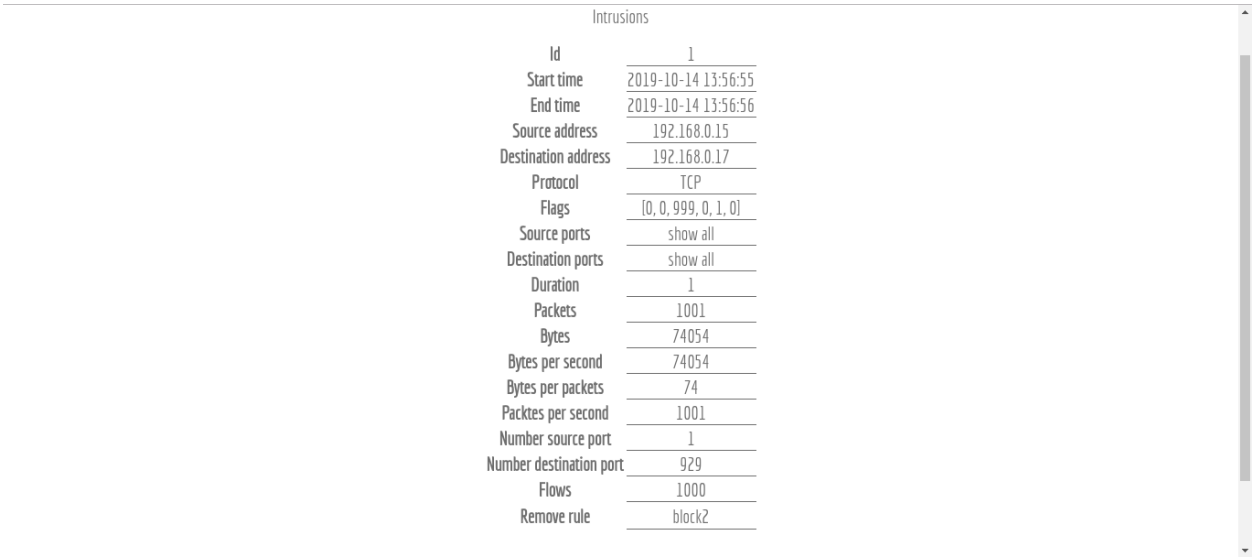


FIGURA 25 - Informações sobre o ataque de *port scan* detectado.
FONTE: Autor (2019).

Conforme supracitado foi estipulado o escaneamento de 1023 portas, contudo, na Figura 25 este número foi contabilizado em apenas 929. O motivo deste fator foi a velocidade no qual o ataque foi lançado e as limitações de hardware onde os experimentos foram realizados resultando na perda de alguns pacotes. Porém, mesmo com menos dados a sensibilidade do IPS permitiu a detecção. Ao comparar o volume de bytes enviados pelo *port scan* em relação ao ataque DoS é observado uma quantidade 97.37 vezes menor, indicando o motivo pelo qual este ataque não afetou a rede. As informações dos campos *Source ports* e *Destination ports* estão ocultas devido ao grande número de portas geradas. A Figura 26 mostra estas informações através do clique na opção *show all*.

FIGURA 26 - Portas utilizadas no ataque de *port scan*.
FONTE: Autor (2019).

A porta de origem foi gerada aleatoriamente sendo utilizado o número 1414, contudo, as portas de destino seguiram o intervalo de números definido pelas portas reservadas. Nota-se que devido a grande quantidade de portas listadas não é possível visualizar todas elas em uma única janela.

4.3.3 Ataque simultâneo

Outro cenário importante e mais complexo é a detecção de ataques ocorrendo simultaneamente. Este experimento foi realizado com o objetivo de a aferir a capacidade e a sensibilidade do sistema em distinguir múltiplos ataques dentro da mesma janela temporal. Portanto, utilizando o mesmo alvo e as mesmas configurações definidas anteriormente, o ataque DoS foi executado através do *host* 1 e após 10 segundos, o ataque de *port scan* foi lançado pelo *host* 5. O tempo entre os ataques foi determinado com o propósito de dificultar ainda mais a detecção de ambos, pois o DoS traz uma sobrecarga na infraestrutura de rede ocasionando perdas de pacotes. Foi observado durante a ocorrência dos ataques a perda de 59% dos pacotes UDP, 10% dos pacotes TCP e um consumo de 784.33 MB de memória RAM pelo controlador SDN. Mediante a este cenário, o componente IPS ao analisar os fluxos IP conseguiu detectar os dois ataques de rede com sucesso. A Figura 27 mostra a quantidade de ataques de detectados.

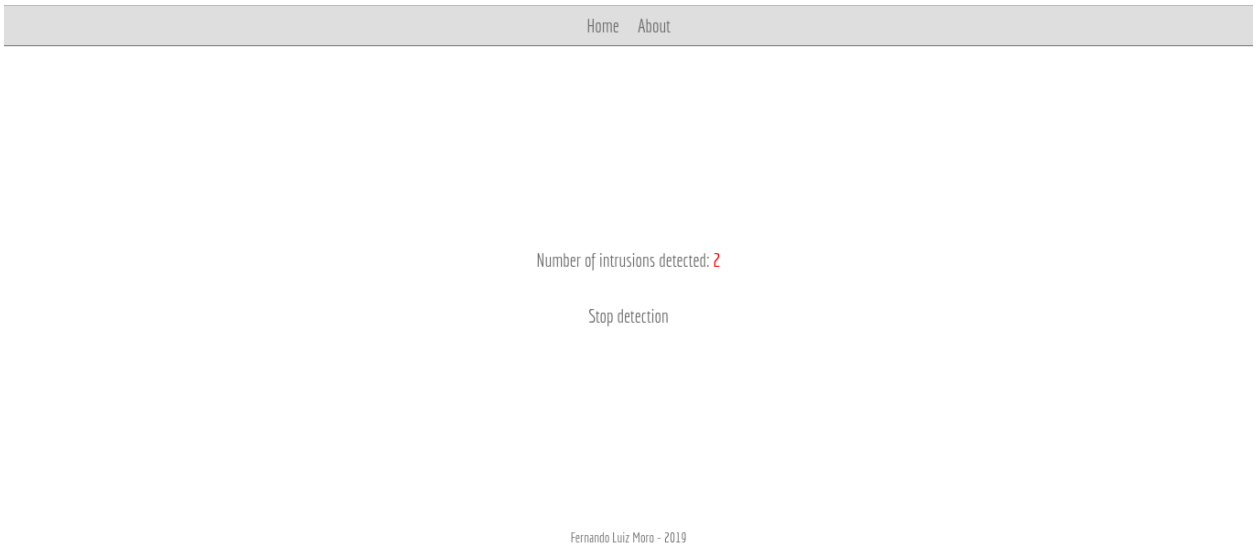


FIGURA 27 - Quantidade de ataques simultâneos detectados pelo IPS.
FONTE: Autor (2019).

Ao lançar o *port scan* após um intervalo de tempo, correu-se o risco de perder todos os pacotes gerados devido ao alto volume de requisições enviadas pelo ataque DoS. Contudo, ao analisar os detalhes desta ocorrência observou-se que apenas 255 pacotes dos 1023 enviados foram perdidos. O procedimento de mitigação inseriu sem a intervenção humana as regras de bloqueio nos *switches* OpenFlow dos quais os dispositivos atacantes estavam conectados, mesmo eles estando em diferentes sub redes. A Figura 28 e 29 mostram as regras de bloqueio nas tabelas de fluxos do *switches* OpenFlow 2 e 3.

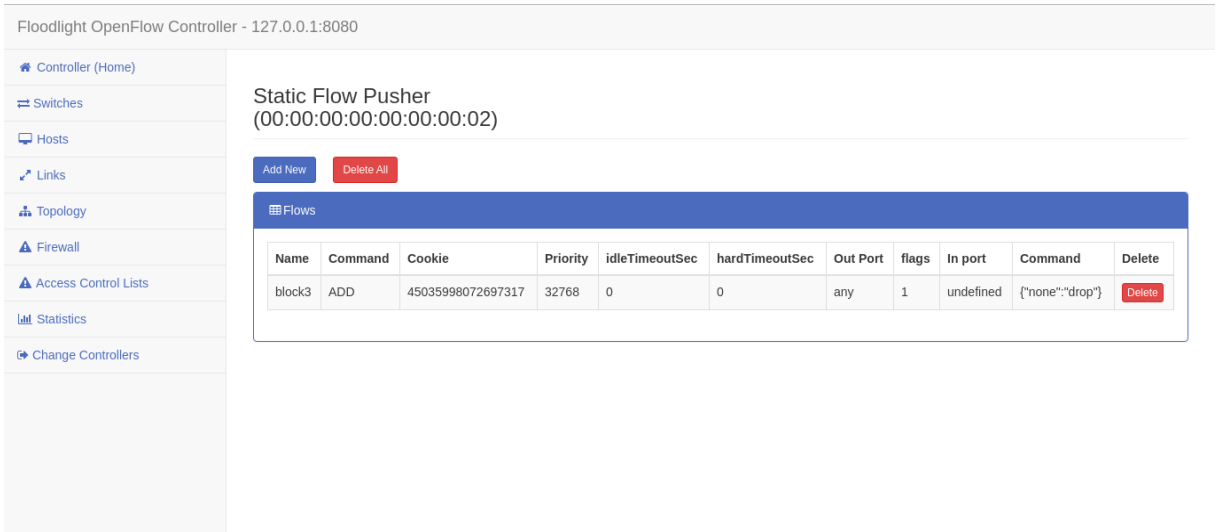


FIGURA 28 - Regra de bloqueio inserida no *switch* OpenFlow 2.
FONTE: Autor (2019).

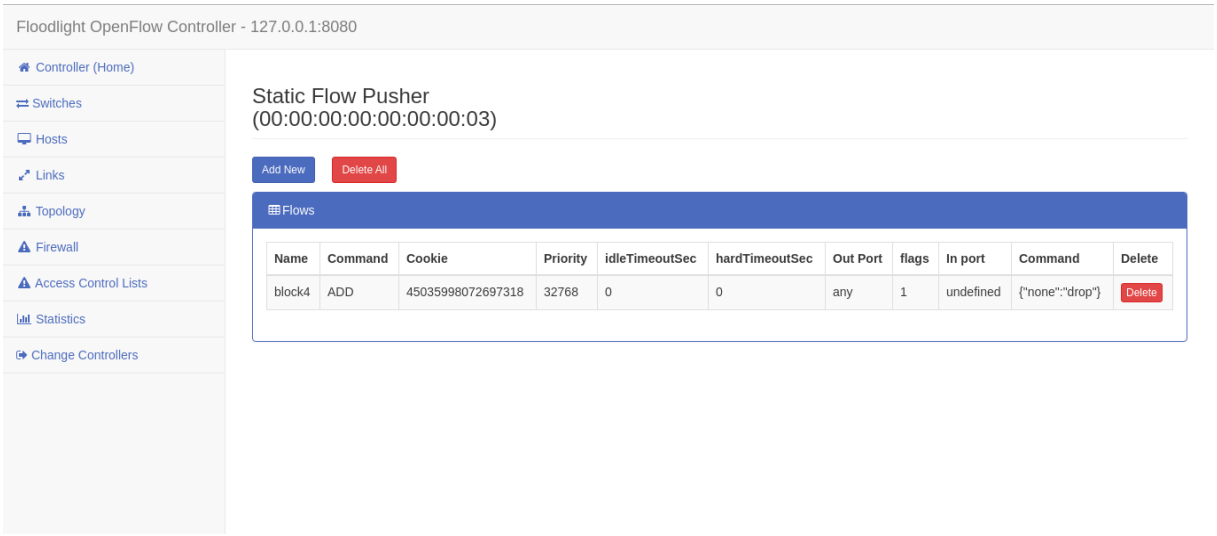


FIGURA 29 - Regra de bloqueio inserida no *switch* OpenFlow 3.
FONTE: Autor (2019).

As regras inseridas ficam ativas por tempo indeterminado ou até que o administrador de rede remova elas manualmente através do IPS. Caso seja necessário, é possível definir o tempo de permanência das regras de acordo com a política de segurança da organização. Conforme observado nas figuras, o comando para cada ação é de rejeitar todos os fluxos provenientes dos dispositivos atacantes.

5 CONSIDERAÇÕES FINAIS

Este trabalho de conclusão de curso propôs uma solução voltada para reduzir a intervenção humana no processo de detecção e mitigação de ataques de rede. No cenário atual das redes de computadores estas atividades são complexas para a gerência de rede em função de diferentes aspectos, tais como o aumento do número de vulnerabilidades e dispositivos conectados, a heterogeneidade das redes e a complexidade dos ataques. Para endereçar estes desafios foi utilizado o aprendizado de máquina no contexto de uma rede definida por software visando a criação de um sistema mais inteligente e hábil para atuar em um ambiente de rede real. Sendo assim, foi desenvolvido um sistema web através da linguagem de programação Python que possibilitou a automatização do diagnóstico de ataques de rede, implicando nas reduções do tempo de reparo, erros da análise e custo operacional.

O sistema é composto por dois componentes principais o *Network Dataset Creation* e *Intrusion Detection System*. O NDC possibilitou a criação de uma base de fluxos IP agregada que reduziu em 98.04% a quantidade de dados que foram processados pelos algoritmos de aprendizado de máquina. A aplicação do método de agregação desenvolvido permitiu melhorar os resultados e o tempo de processamento dos modelos. O componente IPS proporcionou o treinamento, teste e avaliação dos algoritmos *Decision Tree, Gaussian Naive Bayes, K-Nearest Neighbors* e *Support Vector Machine*. Com base na análise das informações geradas pelo IPS o *Decision Tree* foi o que obteve os melhores resultados e se mostrou eficiente para aplicação em tempo real devido a minimização dos falsos negativos e a maximização dos verdadeiros positivos que resultou em um *F1-Score* de 99.93%.

Para validação do sistema foram realizados vários experimentos em uma rede definida por software simulada, sendo que neste trabalho, foram apresentados três deles. No primeiro experimento utilizou-se um ataque DoS e observou-se que o grande volume de requisições afetou o consumo de memória do controlador SDN e a comunicação entre os *hosts*. Mesmo com a sobrecarga na rede o sistema proposto conseguiu em poucos segundos após o término da janela de exportação, detectar e mitigar a ocorrência deste ataque. A sensibilidade do sistema foi avaliada com sucesso no segundo experimento através da detecção de um ataque de *port scan* configurado para gerar um volume inexpressivo de dados. O terceiro e último experimento buscou avaliar a capacidade do sistema em detectar e mitigar ataques de rede em um cenário mais complexo. Para isso foi executado dois ataques simultaneamente de diferentes sub-redes contra o servidor HTTP. Novamente, os ataques foram detectados corretamente e as regras de bloqueio foram automaticamente aplicadas em cada um dos switches OpenFlow onde os dispositivos atacantes estavam conectados.

Em todos os experimentos realizados a detecção e mitigação foram executadas pelo sistema sem a intervenção humana provendo eficiência e agilidade no tratamento dos ataques e consequentemente proporcionando maior resiliência à rede, bem como maximizando a disponibilidade e a confiabilidade dos serviços prestados por ela. Desta forma, através do sistema proposto neste trabalho as tarefas atribuídas ao gerente de rede podem ser reavaliadas, proporcionando a ele um enfoque nos aspectos mais estratégicos e de mais alto nível da gerência de rede.

Outro aspecto a ser destacado como contribuição deste trabalho, diz respeito ao sistema viabilizar a utilização de inúmeros algoritmos de aprendizado de máquina propostos na literatura. A utilização e disseminação mais contundente da inteligência artificial e do aprendizado de máquina ainda enfrentam determinados desafios como os aspectos relacionados a modelagem do problema que será tratado, o modo que os algoritmos serão treinados e avaliados e como viabilizar a automatização destes processos. Através do sistema proposto estas tarefas complexas são automatizadas e simplificadas.

Como trabalho futuro o sistema poderá ser implantado e avaliado em um

ambiente de rede real. Novos algoritmos poderão ser acrescentados no componente IPS e avaliados através dos métodos de padronização, normalização e transformação dos dados com objetivo de aferir o impacto nos resultados dos modelos. Além disso, pode ser explorado o tratamento automatizado dos falsos alarmes para reduzir ainda mais a intervenção do administrador de rede, bem como aprimorar a usabilidade e aparência da interface web do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

ABEL, M.; FIORINI, S. R. Uma revisão da Engenharia de Conhecimento: evolução, paradigmas e aplicações. **International Journal of Knowledge Engineering and Management**, v. 2, p. 35, 2013.

ABRAMS, L. **Dramatic Increase of DDoS Attack Sizes Attributed to IoT Devices**. Disponível em: <https://www.bleepingcomputer.com/news/security/dramatic-increase-of-ddos-attack-sizes-attributed-to-iot-devices/>. Acesso em: 25 maio. 2019.

AHMAD, I. **How Much Data Is Generated Per Minute? The Answer Will Blow Your Mind Away**, 2018. Disponível em: <https://www.digitalinformationworld.com/2018/06/infographics-data-never-sleeps-6.html>. Acesso em: 4 maio. 2019

AHMED, M.; NASER MAHMOOD, A.; HU, J. A survey of network anomaly detection techniques. **Journal of Network and Computer Applications**, v. 60, p. 19–31, 2016.

AIZUDDIN, A. A. et al. **DNS amplification attack detection and mitigation via sFlow with security-centric SDN**. Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication - IMCOM '17. **Anais**... In: THE 11TH INTERNATIONAL CONFERENCE. Beppu, Japan: ACM Press, 2017. Disponível em: <http://dl.acm.org/citation.cfm?doid=3022227.3022230>. Acesso em: 4 mar. 2019,

ALTEXSOFT. **Machine Learning Project Structure: Stages, Roles, and Tools. AltexSoft**, 2019. Disponível em: <https://www.altexsoft.com/blog/datascience/machine-learning-project-structure-stages-roles-and-tools/>. Acesso em: 6 abr. 2019.

AMARAL, A. A. et al. Deep IP flow inspection to detect beyond network anomalies. **Computer Communications**, v. 98, p. 80–96, 2017.

AMARAL, A. D. A. **COMPUTAÇÃO AUTONÔMICA APLICADA AO DIAGNÓSTICO E SOLUÇÃO DE ANOMALIAS DE REDES DE COMPUTADORES**. Doutorado—Campinas, SP: Universidade Estadual de Campinas (UNICAMP), 2015.

BELOUCH, M.; EL HADAJ, S.; IDHAMMAD, M. Performance evaluation of intrusion detection based on machine learning using Apache Spark. **Procedia Computer Science**, v. 127, p. 1–6, 2018.

BHUNIA, S. S.; GURUSAMY, M. **Dynamic attack detection and mitigation in IoT using SDN**. 2017 27th International Telecommunication Networks and Applications Conference (ITNAC). **Anais**... In: 2017 27TH INTERNATIONAL TELECOMMUNICATION NETWORKS AND APPLICATIONS CONFERENCE (ITNAC). Melbourne, VIC: IEEE, 2017. Disponível em: <http://ieeexplore.ieee.org/document/8215418/>. Acesso em: 30 mar. 2019.

BRAUCKHOFF, D. et al. Anomaly Extraction in Backbone Networks Using Association Rules. **IEEE/ACM Transactions on Networking**, v. 20, n. 6, p. 1788–1799, 2012.

BROWNLEE, J. **Overfitting and Underfitting With Machine Learning Algorithms. Machine Learning Mastery**, 2016. Disponível em: <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/>. Acesso em: 21 out. 2019.

BROWNLEE, J. **Difference Between Classification and Regression in Machine Learning. Machine Learning Mastery**, 2017a. Disponível em: <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>. Acesso em: 10 nov. 2019.

BROWNLEE, J. **What is the Difference Between Test and Validation Datasets? Machine Learning Mastery**, 2017b. Disponível em: <https://machinelearningmastery.com/difference-test-validation-datasets/>. Acesso em: 15 out. 2019.

BUCZAK, A. L.; GUVEN, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. **IEEE Communications Surveys & Tutorials**, v. 18, n. 2, p. 1153–1176, 2016.

CARVALHO, L. F. **Um Ecossistema para Detecção e Mitigação de Anomalias em Redes Definidas por Software**. Doutorado—Campinas, SP: Universidade Estadual de Campinas (UNICAMP), 2018.

CERT.BR. **Cartilha de Segurança para Internet**. 2. ed. São Paulo, SP: Comitê Gestor da Internet no Brasil (CGI.br), 2012.

CISCO. **Cisco 2018 - Annual Cybersecurity Report**. San Jose, CA: Cisco, 2018. Disponível em: <https://www.cisco.com/c/en/us/products/security/security-reports.html#~stickynav=4>.

COSTA, L. R. **OpenFlow e o Paradigma de Redes Definidas por Software**. Bacharelado—Brasília, DF: Universidade de Brasília, 2013.

COUTO, A. V. DO. **Uma abordagem de Gerenciamento de Redes baseado no Monitoramento de Fluxos de Tráfego NetFlow com o suporte de técnicas de Business Intelligence**. Mestrado—Brasília, DF: Universidade de Brasília, 2012.

DACIER, M. C. et al. Security Challenges and Opportunities of Software-Defined Networking. **IEEE Security & Privacy**, v. 15, n. 2, p. 96–100, 2017.

DAS, S.; NENE, M. J. **A survey on types of machine learning techniques in intrusion prevention systems**. 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). **Anais**... In: 2017 INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, SIGNAL PROCESSING AND NETWORKING (WISPNET). Chennai: IEEE, 2017. Disponível em: <http://ieeexplore.ieee.org/document/8300169/>. Acesso em: 6 abr. 2019.

GARTNER. **Gartner Identifies Top 10 Strategic IoT Technologies and Trends**. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends>. Acesso em: 4 maio. 2019.

HAMID, Y.; SUGUMARAN, M.; JOURNAUX, L. **Machine Learning Techniques for Intrusion Detection: A Comparative Analysis**. Proceedings of the International Conference on Informatics and Analytics - ICIA-16. **Anais**... In: THE INTERNATIONAL CONFERENCE. Pondicherry, India: ACM Press, 2016. Disponível em: <http://dl.acm.org/citation.cfm?doid=2980258.2980378>. Acesso em: 2 mar. 2019.

HOFSTEDE, R. et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. **IEEE Communications Surveys & Tutorials**, v. 16, n. 4, p. 2037–2064, 2014.

KAKIHATA, E. M. et al. Intrusion Detection System Based On Flows Using Machine Learning Algorithms. **IEEE Latin America Transactions**, v. 15, n. 10, p. 1988–1993, 2017.

KEMP, S. **Digital 2019: Global Internet Use Accelerates**. Disponível em: <https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates>. Acesso em: 4 maio. 2019.

LOBATO, A. G. P.; LOPEZ, M. A.; DUARTE, O. C. M. B. An Accurate Threat Detection System through Real-Time Stream Processing. **Technical Report, Electrical Engineering Program**, p. 6, 2016.

METI, N.; NARAYAN, D. G.; BALIGAR, V. P. **Detection of distributed denial of service attacks using machine learning algorithms in software defined networks**. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). **Anais**... In: 2017 INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING, COMMUNICATIONS AND INFORMATICS (ICACCI). Udupi: IEEE, 2017. Disponível em: <http://ieeexplore.ieee.org/document/8126031/>. Acesso em: 23 mar. 2019.

MOHAMMED, S. S. et al. **A New Machine Learning-based Collaborative DDoS Mitigation Mechanism in Software-Defined Network**. 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). **Anais**... In: 2018 14TH INTERNATIONAL CONFERENCE ON WIRELESS AND MOBILE COMPUTING, NETWORKING AND COMMUNICATIONS (WIMOB). Limassol: IEEE, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8589104/>. Acesso em: 23 mar. 2019.

MORO, F. L. et al. Detecção e mitigação de um ataque DoS em seu estágio inicial em uma rede definida por software. **IX SULCOMP**, v. 9, p. 10, 2018.

MOUSAVI, S. M. **Early Detection of DDoS Attacks in Software Defined Networks Controller**. Mestrado—Ottawa, Ontario: Carleton University, 2014.

NAJAFABADI, M. M. et al. **Detection of SSH Brute Force Attacks Using Aggregated Netflow Data**. 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). **Anais**... In: 2015 IEEE 14TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS (ICMLA). Miami, FL, USA: IEEE, dez. 2015. Disponível em: <http://ieeexplore.ieee.org/document/7424322/>. Acesso em: 2 mar. 2019.

OPEN NETWORKING FOUNDATION. **Software-Defined Networking (SDN) Definition**. Disponível em: <https://www.opennetworking.org/sdn-definition/>. Acesso em: 4 mar. 2019.

PANDIKUMAR, D. T.; ATKILT, F.; HASSEN, C. A. Early Detection of DDoS Attacks in a Multi-Controller Based SDN. **International Journal of Engineering Science and Computing**, v. 7, n. 6, p. 13422–13429, 2017.

PETTERS, J. **IDS vs. IPS: What is the Difference?** Disponível em: <https://www.varonis.com/blog/ids-vs-ips/>. Acesso em: 21 set. 2019.

PWNIE EXPRESS. **2017 Internet of Evil Things Report | Pwnie Express**. Disponível em: <https://www.pwnieexpress.com/2017-internet-of-evil-things-report>. Acesso em: 4 maio. 2019.

PYTHON SOFTWARE FOUNDATION. **sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.7.4 documentation**. Disponível em: <https://docs.python.org/3.7/library/sqlite3.html>. Acesso em: 30 set. 2019.

SARANG, R. **Trending: IoT Malware Attacks of 2018**. Disponível em: <https://securingtomorrow.mcafee.com/consumer/mobile-and-iot-security/top-trending-iot-malware-attacks-of-2018/>. Acesso em: 25 maio. 2019.

SCIKIT-LEARN. **Scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation**. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 6 abr. 2019.

SINGH, S.; KHAN, R. A.; AGRAWAL, A. **Prevention mechanism for infrastructure based Denial-of-Service attack over software Defined Network**. International Conference on Computing, Communication & Automation. **Anais**... In: 2015 INTERNATIONAL CONFERENCE ON COMPUTING, COMMUNICATION & AUTOMATION (ICCCA). Greater Noida, India: IEEE, 2015. Disponível em: <http://ieeexplore.ieee.org/document/7148442/>. Acesso em: 30 mar. 2019.

TRAMMELL, B.; CLAISE, B. **Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information**. Disponível em: <https://tools.ietf.org/html/rfc7011>. Acesso em: 24 fev. 2019.

WANG, H.; XU, L.; GU, G. **FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks**. 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. **Anais**... In: 2015 45TH ANNUAL IEEE/IFIP INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS (DSN). Rio de Janeiro, RJ: IEEE, 2015. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7266854>. Acesso em: 4 mar. 2019.

XIN, Y. et al. Machine Learning and Deep Learning Methods for Cybersecurity. **IEEE Access**, v. 6, p. 35365–35381, 2018.

YAN, Q. et al. Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges. **IEEE Communications Surveys & Tutorials**, v. 18, n. 1, p. 602–622, 2016.