Um Testbed para o Ensino de Abordagens de IDS Baseadas em IA em Redes Emuladas

Paulo E. Valentim [1], **Reinaldo B. Braga**[2], **Antonio W. Oliveira**[3]

[1]Instituto Federal de Educação, Ciência e Tecnologia do (IFCE)
Caixa Postal – 60.040-215 – Fortaleza – CE – Brazil

{ericson.valentim,reinaldo.braga,wendell.rodrigues}@ifce.edu.br

Abstract. *In recent years, there has been a significant increase in internet traffic, driven by the global pandemic and the growing importance of online activities. As a result of this growth, the number of cyber crimes has also increased. In this context, Intrusion Detection Systems (IDSs) need to improve detection accuracy and reduce false alarm rates. This work presents an emulated network-based testbed for teaching IDS approaches based on Artificial Intelligence (AI). As a result, a high-level, scalable interface environment was developed that allows the community to focus on their approaches with less concern for issues related to the test environment.*

Resumo. *Nos últimos anos, houve um aumento significativo no tráfego da internet, impulsionado pela pandemia global e pela crescente importância das atividades online. Como resultado desse crescimento, o número de crimes cibernéticos também aumentou. Nesse contexto, os Sistemas de Detecção de Intrusão (IDSs) precisam melhorar a precisão da detecção e reduzir as taxas de falsos alarmes. Este trabalho apresenta um testbed baseado em rede emulada para o ensino de abordagens IDS baseadas em Inteligência Artificial (IA). Como resultado, foi desenvolvido um ambiente de interface escalável e de alto nível, que permite à comunidade se concentrar em suas abordagens com menos preocupações com problemas relacionados ao ambiente de teste.*

1. Introdução

A era moderna de conectividade com a Internet e o avanço das tecnologias de comunicação abriram um campo cada vez maior para o desenvolvimento de ataques cibernéticos. Isso resultou na necessidade de detectar esses ataques sofisticados de forma rápida e precisa. A detecção de comportamento anômalo é uma tarefa de análise de dados que busca identificar padrões emergentes e suspeitos [Marir et al. 2019].

Há uma clara e significativa influência negativa em vários aspectos quando se trata de crimes cibernéticos. De acordo com um estudo realizado pela McAfee, uma empresa de segurança digital, foi revelado que os crimes cibernéticos e as invasões em sistemas corporativos causaram danos superiores a US$ 1 trilhão em 2020 [McAfee 2020].

Uma solução amplamente adotada para monitorar e inspecionar as atividades em um computador ou sistema de rede é o Sistema de Detecção de Intrusão (IDS). Seu objetivo principal é identificar potenciais ameaças [N. Moustafa and Slay 2019]. No entanto,

muitos IDS ainda enfrentam o problema de gerar um alto número de falsos alarmes, resultando em alertas excessivos para situações de baixa ameaça. Para lidar com isso, pesquisadores têm se dedicado ao desenvolvimento de IDS com taxas de detecção mais altas e menos falsos alarmes.

O IDS baseado em aprendizado fornece um sistema para descobrir classes de ataques com base no comportamento de ataque normal e aprendido. Outro ponto forte é a capacidade das técnicas de IA de encontrar semelhanças em uma grande quantidade de dados. Essas técnicas prometem detecção automatizada baseada em dados que infere informações sobre tráfego de rede malicioso a partir da grande quantidade de rastreamentos de tráfego disponíveis [Mishra et al. 2019].

Além disso, a IA pode ser usada para descobrir anomalias nos dados sem a necessidade de conhecimento prévio desses dados. Ao combinar as características das técnicas de IA e unidades de computação capazes, é possível criar uma arma poderosa para responder a ameaças de invasão de rede. Além disso, há muitos dados hoje em dia, mas a experiência humana é limitada e cara. Assim, usando IA, o software pode descobrir automaticamente padrões que os humanos podem não ser capazes de reconhecer devido à escala dos dados. Na ausência de IA, especialistas humanos precisariam definir regras criadas manualmente que não seriam escaláveis [Mahfouz et al. 2020].

Pesquisas recentes usam diferentes técnicas e processos para construir IDSs. Em sua pesquisa [Mishra et al. 2019] apresentam uma revisão sistemática abrangente sobre o uso de técnicas de aprendizagem para IDS. [Zieglmeier et al. 2018] implementou um testbed para um cenário real de ataques automotivos. [Sai Kiran et al. 2020] criaram um sistema de detecção de intrusão para ambiente IoT usando técnicas de aprendizado de máquina. Os pesquisadores [Mahfouz et al. 2020], [G. Karatas and Koray 2018] e [Kanimozhi and Jacob 2021] usaram técnicas de *Machine Learning* (ML) para IDS e [Zhong et al. 2020], [Kim et al. 2020] e [Ferrag et al. 2020] técnicas de *Deep Learning* (DL). Foi proposto em [Silva et al. 2017] um ambiente virtual de aprendizagem utilizando redes emuladas para o ensino de redes de computadores. Todos eles tiveram que construir seu próprio ambiente (real ou emulado) para testar e validar suas soluções.

De fato, os pesquisadores enfrentam dificuldades durante o processo de teste e validação de um IDS, exigindo grande complexidade e conjuntos de dados específicos para treinar os modelos que geralmente não são acessíveis à comunidade acadêmica.

Com o objetivo de facilitar as pesquisas relacionadas aos IDS, apresentamos um testbed para o ensino de abordagens de IDS baseadas em IA em redes emuladas. Nosso propósito é fornecer um ambiente educacional utilizando uma rede de computadores emulada para avaliar a viabilidade de técnicas de aprendizado de máquina e aprendizado profundo na implementação de um IDS. Com nosso fluxo de trabalho, tanto a comunidade acadêmica quanto a técnica podem utilizar modelos de aprendizado personalizados, executar algoritmos pré-definidos, modificá-los e criar conjuntos de dados a partir de capturas de pacotes em tempo real em máquinas virtuais. O ambiente é escalável, replicável e capaz de simular redes de computadores do mundo real. Além disso, nossa proposta inclui uma interface gráfica que simplifica a seleção de recursos, pré-processamento do conjunto de dados e execução do modelo, tornando-o aplicável inclusive em ambientes de sala de aula.

As principais contribuições deste trabalho são as seguintes:

- Desenvolver um testbed para o ensino em um ambiente emulado com máquinas virtuais para realizar testes experimentais e validar abordagens;
- Avaliar o modelo proposto e outros modelos usando métricas de precisão.

2. Descrição do Modelo

O testbed proposto baseia-se na construção de um ambiente de rede de computadores emulado para verificar a viabilidade de técnicas de aprendizado de máquina e aprendizado profundo para a implementação de um IDS. Embora seja possível alterar a topologia da rede de teste executada no emulador adicionando novos componentes, a Figura 1 apresenta um esquema inicial de conexão.

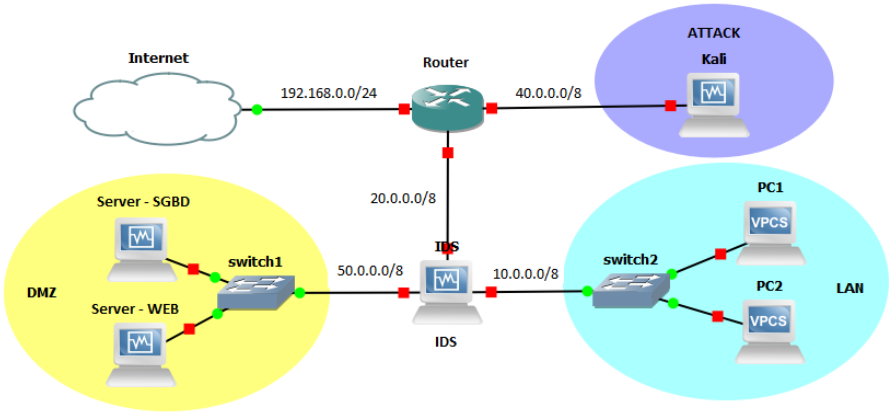

Figura 1. Topologia de rede inicial.

2.1. Topologia de Rede

Os nós mostrados na Figura 1 serão responsáveis por simular ataques cibernéticos tanto para a rede LAN quanto para qualquer um dos servidores DMZ (WEB ou SGBD). Uma VM baseada em Kali Linux possui as ferramentas de sistema pré-instaladas e adicionamos algumas outras como parte da solução, como pacote Anaconda Python3, scapy para gerar pacotes de dados personalizados de acordo com o tipo de ataque desejado e script Slowloris para gerar ataques do tipo negação de serviço e [CICFlowMeter 2018].

O componente principal neste cenário é o IDS. Trata-se de uma VM Linux com 16 GB de memória e três processadores virtuais, sendo responsável por verificar o tráfego de rede e detectar possíveis intrusões através da ferramenta CICFlowMeter, e uma ferramenta fornecida para gerenciar modelos ML ou DL e banco de dados relacional para armazenar informações de detecção para avaliação posterior.

O IDS tem três interfaces de rede. Uma para conexão com o roteador externo, outra para a rede DMZ onde estão localizados os servidores locais e a última interface para conexão com a LAN onde estão colocados os dispositivos do usuário.

A DMZ é composta por duas VMs, ambas com sistema operacional Linux. O servidor WEB é uma VM executando o apache2. O SGBD é uma VM executando os seguintes serviços: apache2, sshd, ftpd e mysql. Dispositivos LAN são componentes adicionais que permitem simular um ambiente do mundo real.

2.2. Aplicando KDD no Experimento

Como parte do fluxo de trabalho proposto, aplicamos conceitos de descoberta de conhecimento a bancos de dados (KDD). A Figura 2 mostra os passos do KDD e como usamos em nossa abordagem é descrito a seguir.

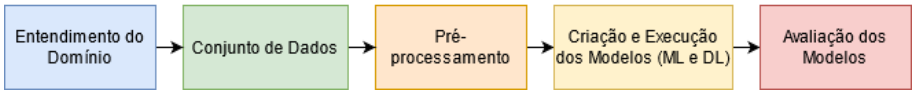

Figura 2. Etapas do processo KDD usadas no fluxo de trabalho.

2.2.1. Compreensão do Domínio

É realizada uma extensa pesquisa na literatura sobre IDS, técnicas de aprendizagem e conjuntos de dados mais utilizados, com o objetivo principal de entender os principais problemas enfrentados pelos pesquisadores. Neste trabalho, esta primeira etapa do processo pode ser comparada aos estudos realizados para a construção da motivação e descrição da problemática da proposta.

2.2.2. Conjunto de Dados

Esta etapa é detalhada na Figura 3. Consiste na geração de um conjunto de dados utilizando a ferramenta CICFlowMeter. Ao instanciar o ambiente virtual em emulador (GNS3) e executar o CICFlowMeter, é necessário escolher um dos dois modos de operação da ferramenta: offline ou tempo real. O modo off-line gera um conjunto de dados (.csv) a partir de uma captura de pacote fornecida pelo usuário por meios próprios. No modo tempo real, a ferramenta gera um conjunto de dados a partir da captura de pacotes de uma interface de rede selecionada pelo usuário. Após selecionar a interface de rede, é necessário gerar o tráfego de dados de acordo com o tipo de ataque que se pretende realizar no testbed. Ao final de ambos os modos a ferramenta produz um conjunto de dados (.csv) que é utilizado na etapa seguinte da proposta.

2.2.3. Pré-processamento

Aqui, realizamos um estudo detalhado dos 84 recursos gerados pela ferramenta CICFlowMeter. A Figura 4 detalha os passos deste processo. Depois de carregar o conjunto de dados, os dados passam por um processo de leitura linha por linha para limpar rótulos, registros duplicados, normalização, registros ausentes e erros de formatação. A fim de organizar os dados de forma balanceada, os dados foram submetidos à padronização utilizando a biblioteca *sklearn*. Depois de terminar o processo de limpeza e formatação, um novo conjunto de dados é gerado que, posteriormente, será testado pela abordagem selecionada.

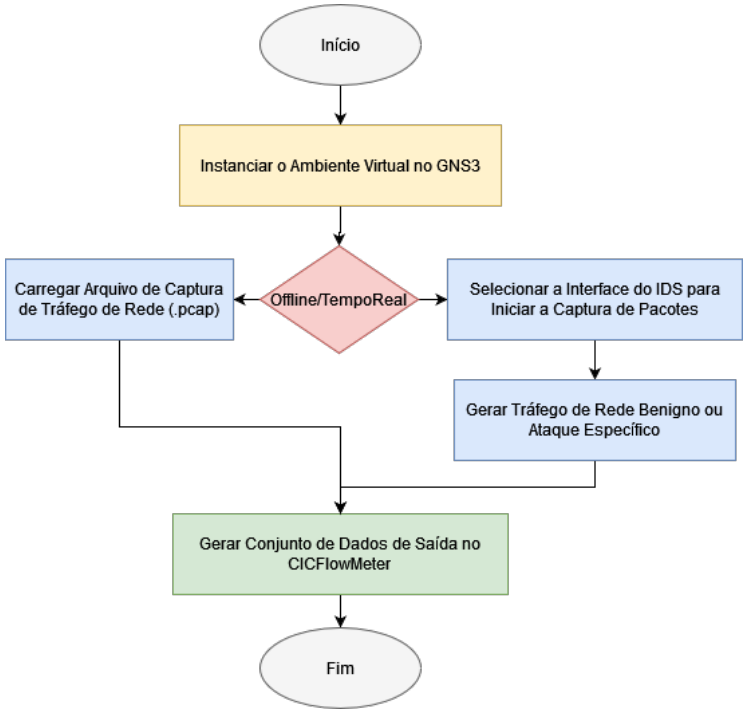

Figura 3. Fluxo de execução para gerar o conjunto de dados.

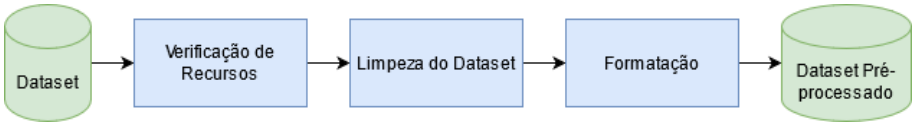

Figura 4. Etapas de pré-processamento.

2.2.4. Criação e Execução de Modelos

O ambiente fornece nativamente quatro técnicas de aprendizagem conhecidas para IDS, Rede neural convolucional (CNN), Naive Bayes (NB), Regressão Logistica (RL) e Multi-Layer Perceptron (MLP), treinadas usando o [CSE-CIC 2018], destaca-se que a priori, todas técnicas usaram parâmetros padrões da biblioteca *sklearn*. Também fornece oito tipos de ataques para realizar testes: Brute Force SSH/FTP, Brute Force WEB, Bot, DOS Golden Eye - DOS Slowloris, DOS Slow HTTP - Hulk, DDOS LOIC HTTP - DDOS HOIC, Infiltration e SQL Injection. Ele ainda entrega três capturas de dados no formato *.pcap* com tráfego Benigno, com ataque DOS Slowloris gerado online e outra com tráfego misto usual para teste.

Cada técnica de aprendizagem foi treinada e testada 10 (dez) vezes com cada conjunto de dados, que representam um dia de tráfego de rede benigno e malicioso. Nós aplicamos a função (train test split) da biblioteca *skleran* para dividir o conjunto de dados de forma balanceada nas proporções 80% para fase de treinamento dos modelos e 20% para fase de testes. Após o treinamento dos modelos, a melhor rodada foi selecionada e as métricas de desempenho foram aplicadas. Para facilitar a execução dos modelos de aprendizagem pelo usuário, existe ainda uma ferramenta gráfica para interfacear o fluxo do testbed (Figura 5).

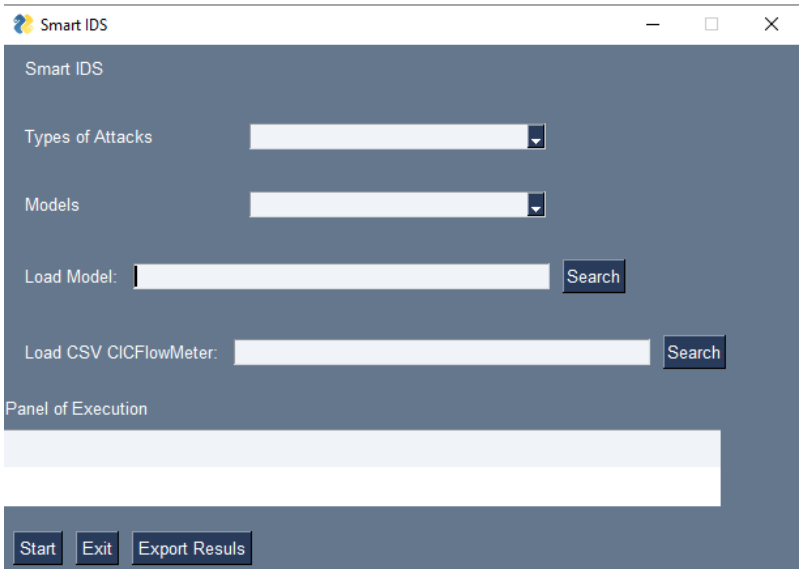

Figura 5. Interface gráfica desenvolvida em Python.

Para usuários avançados, o ambiente permite a execução de um modelo de aprendizado treinado pelo usuário no formato *.pkl*. O usuário deve criar um fluxo de tráfego malicioso através da ferramenta CICFlowMeter e gerar um arquivo CSV rotulado no atributo 'Label' com '1' e '0' para tráfego benigno, este arquivo será dividido e utilizado no processo de treinamento e validação do modelo.

Resultados, como caminhos de entrada e saída dos arquivos *.csv*, número de registros, tipo de ataque, técnica utilizada, número de registros classificados como rede de tráfego benigna e maliciosa e, por fim, a probabilidade geral são apresentados após o processamento. Eles podem ser exportados nos formatos *.json* e *.txt*. Logs classificados como tráfego de rede malicioso podem ser salvos no banco de dados relacional fornecido.

2.2.5. Ambiente Experimental

Os experimentos foram realizados inicialmente em computador com sistema operacional Linux Ubuntu 22.04 LTS, processador i7-7700 3.60 GHz (4 núcleos) e memória RAM de 32GB. No entanto, por limitações de processamento, memória e de uma GPU para execução do modelo de aprendizagem profunda, optamos por transferir parte dos experimentos para a plataforma do *Google Colab* que possui máquinas pré-configuradas para execução de algoritmos em Python3 e vários modelos avançados de GPU.

3. Resultados e Discussão

Esta seção apresenta os resultados dos experimentos utilizando a metodologia proposta. Na Tabela 1 é apresentada a quantidade de registros benignos e classes de ataques por conjunto de dados após realizarmos o pré-processamento dos dados. Totalizando mais de 8 milhões de registros analisados durante o processo de detecção de intrusão em nossos experimentos.

Aplicamos as técnicas de aprendizado listadas anteriormente em cada conjunto de dados 10 vezes e registramos os tempos de execução e os melhores resultados de desem-

Tabela 1. Quantidade de registros por conjunto de dados.

Conjunto de Dados	Tipo de Tráfego	Quantidade de Registros
14-02-2018	Benign	663.808
	SSH-Bruteforce	187.589
	FTP-BruteForce	193.354
15-02-2018	Benign	988.050
	DoS-GoldenEye	41.508
	DoS-Slowloris	1.099
16-06-2018	Benign	446.772
	DoS-SlowHTTPTest	139.890
	DoS-Hulk	461.912
21-02-2018	Benign	360.833
	DoS-HOIC	686.012
	Dos-LOIC-UDP	1.730
22-02-2018	Benign	1.042.603
	BruteForce-Web	249
	BruteForce-XSS	79
	SQL-Injection	34
23-02-2018	Benign	1.042.301
	BruteForce-Web	362
	BruteForce-XSS	151
	SQL-Injection	53
28-02-2018	Benign	538.666
	Infiltration	68.236
01-03-2018	Benign	235.778
	Infiltration	92.403
02-03-2018	Benign	758.334
	Bot	286.191

penho, conforme métricas estabelecidas. Experimentamos vários parâmetros referentes a estrutura das camadas de aprendizagem profunda para os algoritmos CNN e MLP, sempre em busca dos melhores resultados.

Por sua vez, a Tabela 2 mostra o desempenho do algoritmo CNN. Utilizamos 10 épocas para o seu treinamento, medimos também seu desempenho utilizando a configuração de GPU da plataforma *Google Colab*. Este modelo apresentou bons resultados atingindo 0.99 pontos em todas as métricas para 7 conjuntos de dados, e seu tempo de execução com CPU foi bastante elevado em comparação com o tempo de GPU. Com o conjunto de dados 15-02-2018 a velocidade de execução com GPU chegou a ser 11 vezes mais rápido. Ao analisar este modelo, verificamos altas taxas de falso positivo (FP) para tráfego benigno e falso negativo (FN) para detecção de Infiltração com o conjunto de dados 01-03-2018.

O desempenho do algoritmo MLP é apresentado na Tabela 3. Ele foi configurado com 5 camadas ocultas, máximo 200 iterações e modo de ativação *logistc*. Este modelo também apresentou 0.99 pontos em todas as métricas para 7 conjuntos de dados. O maior

Tabela 2. Desempenho do algoritmo CNN em todos os conjuntos de dados.

Dataset	Acurácia	Precisão	Recall	F1-score	CPU (seg.)	GPU (seg.)
14-02-2018	0.99992	0.99992	0.99992	0.99992	905.90	83.21
15-02-2018	0.99968	0.99969	0.99968	0.99968	1169.19	100.61
16-02-2018	0.99804	0.99805	0.99804	0.99804	797.44	87.38
21-02-2018	0.99999	0.99999	0.99999	0.99999	900.94	105.01
22-02-2018	0.99973	0.99961	0.99973	0.99964	923.03	110.31
23-02-2018	0.99964	0.99963	0.99964	0.99956	904.28	110.04
28-02-2018	0.88724	0.84120	0.88724	0.83442	531.86	65.60
01-03-2018	0.88176	0.88118	0.88176	0.88145	292.46	36.57
02-03-2018	0.99904	0.99904	0.99904	0.99904	925.75	108.29

tempo de execução com CPU foi de 77.42 segundos para o conjunto de dados 02-03-2018.

Tabela 3. Desempenho do algoritmo MLP em todos os conjuntos de dados.

Dataset	Acurácia	Precisão	Recall	F1-score	Tempo CPU (seg.)
14-02-2018	0.99989	0.99989	0.99989	0.99989	36.54
15-02-2018	0.99969	0.99969	0.99969	0.99969	72.19
16-02-2018	0.99807	0.99808	0.99807	0.99807	42.66
21-02-2018	0.99997	0.99997	0.99997	0.99997	48.52
22-02-2018	0.99972	0.99956	0.99972	0.99963	59.43
23-02-2018	0.99961	0.99959	0.99961	0.99950	57.04
28-02-2018	0.88426	0.83094	0.88426	0.83991	52.56
01-03-2018	0.81951	0.81313	0.81951	0.80823	75.33
02-03-2018	0.99564	0.99571	0.99564	0.99566	77.42

As Tabelas 4 e 5 mostram os desempenhos dos algoritmos NB e RL. O NB apresentou o pior desempenho com acurácia de apenas 0.13 pontos para o conjunto de dados 28-02-2018. O RL também não demostrou bons resultados em alguns conjuntos de dados, sua acurácia foi em torno de 0.85 pontos.

A partir das tabelas de desempenho, podemos observar que os algoritmos CNN e MLP obtiveram altas taxas em todas as métricas em grande parte dos conjuntos de dados chegando a 0.99 pontos. O CNN foi o que obteve as métricas mais altas levando em consideração todos conjuntos de dados seguido pelo MLP. O NB apresentou o pior desempenho seguido pelo RL. Ambos não conseguiram desempenhar um bom papel para modelos de detecção de intrusão. Em relação ao tempo de execução, o CNN é viável somente para utilização com GPU pois seu tempo de execução com CPU passou de 1100 segundos, chegando a ser 11 vezes maior em comparação com a execução com GPU, ambos utilizando o conjunto de dados 15-02-2018. O RL por vez, apresentou o melhor tempo de execução, no entanto, foi o pior em precisão. Em segundo lugar o MLP apresentou ótimos tempos de execução e excelente precisão para grande parte dos conjuntos de dados.

Um fato que nos chamou atenção foi a baixa eficácia de todos os modelos em relação aos conjuntos de dados 28-02-2018 e 01-03-2018. Ambos possuem registros be-

Tabela 4. Desempenho do algoritmo NB em todos os conjuntos de dados.

Dataset	Acurácia	Precisão	Recall	F1-score	Tempo CPU (seg.)
14-02-2018	0.44980	0.89281	0.44980	0.52644	19.05
15-02-2018	0.67159	0.95661	0.67159	0.76640	21.89
16-02-2018	0.99301	0.99302	0.99301	0.99301	14.53
21-02-2018	0.99997	0.99997	0.99997	0.99997	18.16
22-02-2018	0.23559	0.99965	0.23559	0.38119	20.93
23-02-2018	0.49391	0.99945	0.49391	0.66084	22.27
28-02-2018	0.13053	0.81936	0.13053	0.06147	10.58
01-03-2018	0.36786	0.64468	0.36786	0.32022	7.53
02-03-2018	0.79082	0.88118	0.79082	0.80230	23.25

Tabela 5. Desempenho do algoritmo RL em todos os conjuntos de dados.

Dataset	Acurácia	Precisão	Recall	F1-score	Tempo CPU (seg.)
14-02-2018	0.81926	0.70807	0.81926	0.75159	111.01
15-02-2018	0.95457	0.92823	0.95457	0.93751	103.33
16-02-2018	0.96704	0.96760	0.96704	0.96341	92.21
21-02-2018	0.85071	0.85462	0.85071	0.84480	110.37
22-02-2018	0.99963	0.99930	0.99963	0.99947	252.23
23-02-2018	0.99944	0.99894	0.99944	0.99919	140.83
28-02-2018	0.88725	0.78721	0.88725	0.83424	29.81
01-03-2018	0.71821	0.67069	0.71821	0.60499	17.43
02-03-2018	0.76225	0.75998	0.76225	0.76106	43.85

nignos e de infiltração. Ao analisar as saídas, verificamos uma grande quantidade de registros classificados incorretamente. Altas taxas de FP e FN demostram a falta de capacidade dos modelos em perceber as particularidades de cada tipo de tráfego de rede ou então, uma limitação do próprio conjunto de dados na captura desse tipo de ataque. O melhor resultado nesses conjuntos de dados foi alcançado pelo CNN com 0.88 pontos de acurácia.

4. Conclusão e Perspectivas

Este trabalho apresentou um testbed juntamente com um fluxo de trabalho para ensinar, testar e validar abordagens de IDS baseados em IA. A proposta abrangente inclui módulos que permitem o uso ou criação de conjuntos de dados, seleção de métodos de aprendizagem predefinidos ou personalizados, e geração de padrões de ataque para verificar a eficácia da solução. Como resultado, esse trabalho permite que pesquisadores e profissionais possam ensinar, propor e comparar abordagens de forma simplificada em relação aos modelos atuais.

Todas as ferramentas e configurações discutidas aqui estarão disponíveis gratuitamente no GitHub do projeto (AI/IDS) para auxiliar o processo de ensino prático de IDS. Como perspectiva futura, é necessário expandir e modificar facilmente os métodos existentes. Além disso, uma classificação interna automatizada em forma de ranking das melhores abordagens seria bem-vinda.

Referências

CICFlowMeter (2018). *Applications*. Disponível em: https://www.unb.ca/cic/research/applications.html.

CSE-CIC (2018). *CSE-CIC-IDS2018*. Disponível em: https://www.unb.ca/cic/datasets/ids-2018.html.

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., and Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419.

G. Karatas, O. D. and Koray, A. S. (2018). *Deep Learning in Intrusion Detection Systems*. pages 3–4. *International Congress on Big Data*.

Kanimozhi, V. and Jacob, T. P. (2021). Artificial intelligence outflanks all other machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. *ICT Express*, 7(3):366–370.

Kim, A., Park, M., and Lee, D. H. (2020). Ai-ids: Application of deep learning to real-time web intrusion detection. *IEEE Access*, 8:70245–70261.

Mahfouz, A., Abuhussein, A., Venugopal, D., and Shiva, S. (2020). Ensemble classifiers for network intrusion detection using a novel network attack dataset. *Future Internet*, 12(11).

Marir, N., Wang, H., and Feng, G. (2019). Unsupervised feature learning with distributed stacked denoising sparse autoencoder for abnormal behavior detection using apache spark. In *2019 IEEE 2nd International Conference on Knowledge Innovation and Invention (ICKII)*, pages 473–476.

McAfee (2020). *Report Cyber Crime*. Disponível em: https://www.mcafee.com/enterprise/en-us/assets/reportsrp-hidden-costs-of-cybercrime.pdf.

Mishra, P., Varadharajan, V., Tupakula, U., and Pilli, E. S. (2019). *A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection*. volume 21. *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS*.

N. Moustafa, J. H. and Slay, J. (2019). *A holistic review of Network Anomaly Detection Systems: A comprehensive survey*. volume 128, pages 33–55. *Journal of Network and Computer Applications*.

Sai Kiran, K., Devisetty, R. K., Kalyan, N. P., Mukundini, K., and Karthi, R. (2020). Building a intrusion detection system for iot environment using machine learning techniques. *Procedia Computer Science*, 171:2372–2379. Third International Conference on Computing and Network Communications (CoCoNet'19).

Silva, I., Marques, C., and Lima, R. (2017). Integrando o emulador gns3 como suporte de ensino na disciplina de redes de computadores no ambiente ava. page 1727.

Zhong, W., Yu, N., and Ai, C. (2020). Applying big data based deep learning system to intrusion detection. *Big Data Mining and Analytics*, 3(3):181–195.

Zieglmeier, V., Kacianka, S., Hutzelmann, T., and Pretschner, A. (2018). A real-time remote IDS testbed for connected vehicles. *CoRR*, abs/1811.10945.