Proposta de projeto: Detecção de Intrusão em Redes com uso de machine learning

Erick Alen Ricioli
Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Cornélio Procópio, Paraná, Brasil
erick.1998@alunos.utfpr.edu.br

Lucas Paschoalick
Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Cornélio Procópio, Paraná, Brasil
paschoalick@alunos.utfpr.edu.br

Luiz Felipe Alves Ferreira
Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Cornélio Procópio, Paraná, Brasil
lferreira.2000@alunos.utfpr.edu.br

Vítor Ângelo Misciato Teixeira*
Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Cornélio Procópio, Paraná, Brasil
vitor.2018@alunos.utfpr.edu.br

I. Introdução e motivação

Com a expansão de funcionalidades *web* atingindo ao mínimo 5 grandes indústrias, como o cinema, e-commerce, jogos, música, alimentos, a *internet* se tornou uma forma de comércio viável e muito explorada, devido a sua praticidade e rapidez, tanto para as empresas quanto para os consumidores, observando um crescimento de 1500% entre 2012 a 2022 [1].

Neste período, empresas e negócios passaram a operar dentro do ambiente da *internet* e com esses novos agentes surge a necessidade de que as informações que trafegam na rede cheguem ao destino de forma segura, ou seja, sem adulterações e de forma confidencial.

Um problema comum, neste cenário, é a invasão de redes de computadores que pode causar um prejuízo a empresa, afetando tanto a produtividade quanto a imagem dela. Com base nisso, são definidos 5 pilares que a segurança deve garantir, sendo eles: confidenciabilidade da mensagem, integridade da mensagem, autentificação da mensagem, não repúdio da mensagem e identificação do usuário [2].

Buscando evitar esse tipo de situação, uma série de técnicas e ferramentas de prevenção são utilizadas, como é o caso dos *Intrusion Detection Systems* (IDS), ou Sistemas de Detecção de Intrusão, conforme [3]

> , "Um IDS é um sistema de software ou hardware que é usado para detectar sinais de atividade maliciosa em uma rede ou um computador individual."

As funções de um IDS são divididas entre sensores IDS, que coletam dados em tempo real sobre o funcionamento dos componentes da rede e computadores, e um gerente IDS, que recebe relatórios de sensores.

Existem dois tipos principais, sendo um deles baseado em rede, que monitora uma rede ou um segmento de uma rede, e o outro baseado em *host*, os quais monitoram um único sistema.

O *Network Intrusion Detection System* (NIDS) detecta comportamentos maliciosos baseados em padrões de tráfego e conteúdo, ou seja, faz comparações utilizando os dados dos pacotes da rede. De acordo com [7], a vantagem de usar um NIDS é que além de ser uma abordagem mais econômica, sua resposta é mais rápida, pois seu monitoramento do tráfego é praticamente instantâneo, detectando ataques à medida que ocorrem. Todavia, ele não detecta se os ataques foram bem sucedidos ou não e podem não reconhecer um ataque lançado durante um período de alto tráfego na rede.

Estes sistemas IDS podem se beneficiar do uso do *machine learning* para realizar a comparação dos padrões de tráfego, pois uma vez que o modelo aprende o padrão usado, ele aumenta a eficácia com base na quantidade de dados que forem fornecidos para análise, o que ajuda na evolução desses sistemas de segurança com base na identificação de padrões - tanto explícitos quanto implícitos. Seus algoritmos são diferenciados pela variedade de estilos de aprendizados que podem ser empregados de acordo com a forma de trabalho de cada um [4]. Os aprendizados podem ser classificados em: supervisionados, não-supervisionados e semi-supervisionados.

O aprendizado supervisionado realiza sua captação, como o ser humano, através de uma supervisão realizada por outro ser humano, mostrando quais os dados que se encaixam com as entradas dadas como exemplo. Os dados ficam padronizados de forma explícita e clara, dentro de rótulos já esperados, pois foram pré-determinados [5].

Já o aprendizado não-supervisionado atua de maneira contrária, requerendo do sistema uma decisão própria, a partir de um conjunto de dados predefinidos pelo responsável. Desse modo, o sistema, se bem implementado, pode encontrar padrões implícitos e que não foram identificados anteriormente, podendo levar novas percepções sobre os dados [5].

Além disso, também existe a aprendizagem semi-supervisionada, que mescla os dois tipos citados acima, e

a aprendizagem por reforço, que trabalha com o sistema de recompensa e punição [5].

Para se desenvolver um NIDS usando inteligência artificial, é necessário o pré-processamento dos dados, treinamento do algoritmo e os testes. Vale ressaltar que o tempo de treinamento de um algoritmo de *deep learning* é maior, devido a complexidade de sua estrutura. Quando o modelo é treinado, realiza-se um teste com seu *dataset* - conjunto de dados -, onde ele é avaliado conforme as predições obtidas, onde o tráfego de rede pode atestar um comportamento padrão ou nocivo [6].

Como trata-se de um problema de classificação, existem diferentes algoritmos que podem ser usados, como, por exemplo: árvores de decisão, KNN, máquina de suporte de vetores, agrupamento k-médias, Redes Neurais Artificiais, métodos com ensemble, dentre outras variações [4].

Portanto, a quantidade de brechas de ataque que um sistema pode fornecer - de maneira involuntária - aumenta concomitantemente, podendo levar muito tempo para ser encontrada por um ser humano. Dessa forma, cresce a motivação da implementação do *machine learning* dentro de um sistema de rede que detecta anomalias dentro do sistema com uma boa eficácia conforme os parâmetros estipulados.

O restante do texto é organizado da seguinte maneira: a Seção II traz o objetivo do trabalho; a Seção III descreve a metodologia aplicada para a resolução do problema proposto; a Seção IV apresenta os resultados obtidos nos testes e a Seção V conclui o trabalho, abordando os principais pontos.

II. Objetivo

Dado o exposto, o objetivo do trabalho é desenvolver e testar modelos de *machine learning* que sejam capazes de classificar se um fluxo de pacotes em uma rede representa uma ameaça ou não, e, caso represente, qual o tipo específico de ameaça, ou seja, implementar um NIDS.

III. Metodologia

Para a implementação do NIDS, foi utilizado o *dataset* NSL-KDD para treinamento e testes, disponibilizado pela *University of New Brunswick* (UNB) e utilizado em trabalhos de pesquisa na área. Como ferramenta, foi utilizada a linguagem *Python*, tendo como base o que é chamado *PyData Stack*, composto pelas bibliotecas *pandas, numpy, sklearn e matplotlib*, densenvolvendo no ambiente do *Jupyter Notebook*.

Na Figura 1 é apresentado o *workflow* do desenvolvimento das atividades.

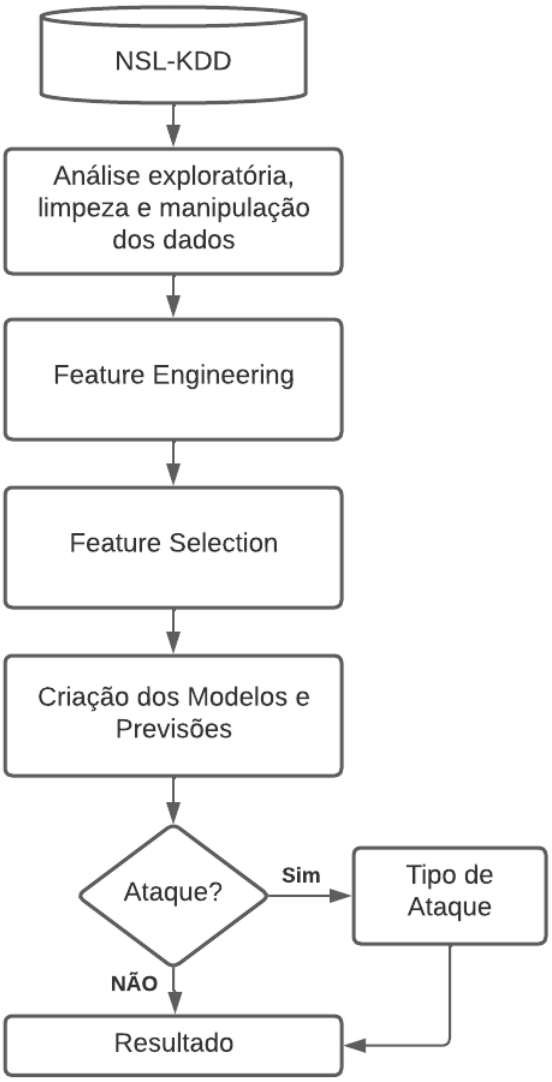

Figura 1. *Workflow* do projeto.

Tendo em vista tais atividades, o trabalho foi divididas em quatro etapas, sendo elas:

- **Análise exploratória, limpeza e manipulação dos dados:** nesta etapa é feito um estudo analítico dos dados, buscando ter um panorama geral dos valores que podem ser encontrados *dataset*, como eles estão correlacionados, se existem problemas de consistência e se há necessidade de fazer mudanças na estrutura;
- ***Feature Engineering* & *Feature Selection*:** ambas as fases relacionadas à escolha de quais atributos utilizar na construção e treinamento de um modelo de *machine learning*. É feito um estudo dos atributos mais representativos e importantes, bem como tratamento dos mesmos, que servirão como base para a fase de seleção;
- **Criação dos Modelos e Previsões:** com os dados selecionados e já tratados, nesta etapa são escolhidos os

algoritmos para o treinamento, bem como são geradas as previsões a partir dos registros do conjunto de testes;
- **Métricas de avaliação:** com as previsões em mãos, são geradas medidas estatísticas para analisar os resultados obtidos e saber como os modelos se comportaram.

A seguir, será apresentado um maior detalhamento do que foi feito em cada uma dessas etapas.

A. Análise exploratória dos dados

Por se tratar de um *dataset* criado e mantido por uma universidade, o NSL-KDD já vem muito bem estruturado, não possuindo campos de valores nulos ou registros duplicados, o que permite a economia de tempo que seria destinado para sua limpeza.

Sobre a estrutura, cada registro possui 43 atributos, sendo 41 desses utilizados para predição e 2 de variáveis *target*. Nesse caso, dos 2 atributos *target*, apenas os valores de *attack type* interessam, podendo ser divididos em cinco classes: *normal* (não ataque), *Denial of Service* (DoS), *Probe*, *User to Root* (U2R), e *Remote to Local* (R2L). Porém, na verdade, cada registro é rotulado como sendo uma subclasse de ataque. Portanto, primeiramente foi feito um mapeamento das subclasses, atribuindo um valor numérico, variando entre 0 e 4, de acordo com a classe que pertence. Na Figura 2 é possível ver a quantidade de registros de cada classe após o mapeamento no conjunto de dados de treinamento.

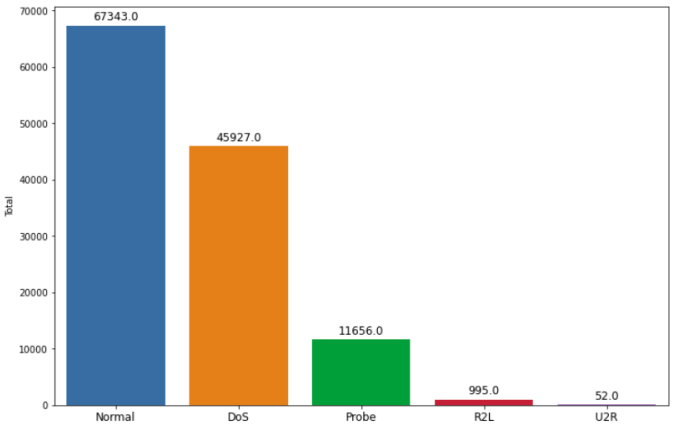

Figura 2. Registros nos de cada classe nos dados de treinamento.

B. Feature Engineering & Feature Selection

Como o *dataset* possui muitos atributos, é normal que nem todos tenham o mesmo nível de relevância para classificação do tipo de ataque. Além disso, o formato que os dados se encontram podem não estar adequados, estando, por exemplo, em escalas diferentes e formato textual, representando um bloqueio para os modelos.

Sendo assim, para se ter noção de quais atributos estão mais relacionados com a variável *target* e causam maior impacto em sua determinação, foram utilizadas duas técnicas: *Mutual information* (MI) e *Feature Importance* (FI). MI é descreve uma relação em termos de incerteza, ou seja, quanto diminui a incerteza de uma variável sabendo o valor de outra, enquanto no caso da FI, é retornado, em termos de porcentagem, o valor de quão determinante um atributo é para definir a *target*, nesse caso, treinando um modelo com todos os atributos. Para determinar o FI foi utilizado o modelo de *Random Forest*.

Com os valores de MI e FI, somados a outros resultados obtidos durante a etapa de análise exploratória, a partir de gráficos e medidas estatísticas, os atributos escolhidos para treinamento e gerar previsões foram:

- *src_bytes:* número de bytes de dados transferidos da origem para o destino em uma única conexão;
- *flag:* status da conexão;
- *protocol type:* protocolo utilizado na conexão;
- *count:* número de conexões com o mesmo *host* de destino que a conexão atual nos últimos dois segundos;
- *dst_bytes:* número de *bytes* de dados transferidos do destino para a origem em uma única conexão;
- *duration:* duração da conexão em segundos;
- *diff_srv_rate:* A porcentagem de conexões que foram para serviços diferentes, entre as conexões agregadas em *count*;
- *same_srv_rate:* A porcentagem de conexões que foram para o mesmo serviço, entre as conexões agregadas em *count*.

Com os atributos selecionados, foi construído um *transformer*, encarregado de padronizar os dados. Para valores numéricos, foi aplicado um *rescaling*, mais especificamente o *MinMaxScaling*, que converte os valores para uma escala que varia de 0 a 1, de acordo com valor máximo e mínimo encontrado. Para valores categóricos, foi utilizado o *One-Hot-Encoding*.

C. Criação dos Modelos e Previsões

Como algoritmos para construção dos modelos, foram escolhidos os de *Decision Tree* (DT), *Random Forest* (RF) e *Support Vector Machines* (SVM). Para cada um deles foi criado um *Pipeline*, composto pelo modelo em questão e pelo *transformer* criado na etapa anterior.

Para treinamento e previsão, como base nos modelos, foram criados dois cenários: no primeiro, classificação envolvendo todas as classes de ataque presentes, e no segundo, classificação binária, no qual todas as classes de ataques foram convertidas para uma única, indicando ser ataque.

Além disso, como um teste alternativo para comparações, também foi criado um terceiro cenário, convertendo, separadamente, os valores das previsões do primeiro cenário, binarizando para ficar como se fosse o segundo cenário.

D. Métricas de avaliação

Como métricas de avaliação nos testes, foram escolhidas as medidas estatísticas de:

- Acurácia: mede o total de imagens classificadas corretamente;
- Precisão: das imagens que receberam uma classe específica, indica quantas delas realmente estão corretas;

- *Recall*: das imagens que compõem uma classe específica, mostra quantas foram classificadas corretamente;
- *F1-Score*: uma média harmônica de precisão e *recall* para obter um único resultado, ponderando para o pior valor.

REFERÊNCIAS

[1] "Volume of data information created, captured, copied, and consumed worldwide from 2010 to 2025". Statista, 2022.

[2] BARRETO, Jeanine dos S.; ZANIN, Aline; SARAIVA, Maurício de O. Fundamentos de redes de computadores. Grupo A, 2018. 9788595027138.

[3] OODRICH, Michael T.; TAMASSIA, Roberto. Introdução à Segurança de Computadores. Grupo A, 2012. 9788540701939.

[4] Sultana, N.; Chilamkurti, N.; Peng, W. "Survey on SDN based network intrusion detection system using machine learning approaches".

[5] Ghori K. M.; Abbasi R. A.; Awais M.; (Member, IEEE).; Imran M.; Ullah A.; Szathmary L. "Performance Analysis of Different Types of Machine Learning Classifiers for Non-Technical Loss Detection". IEEE, 2019. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8943419

[6] Ahmad, Z.; Khan, A. S.; Shiang, C. W.; Abdullah, J.; Ahmad F. "Network intrusion detection system: A systematic study of machine learning and deep learning approaches". Wiley, 2020. https://onlinelibrary.wiley.com/doi/10.1002/ett.4150

[7] Alzahrani, A.O.; Alenazi, M.J.F. Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks. Future Internet 2021, 13, 111. https://doi.org/10.3390/fi13050111