# Aplicação de Redes Neurais Convolucionais e Recorrentes na Detecção de Intrusão em Linux Baseada em Chamadas de Sistema

André Augusto Bortoli
IFPR - Instituto Federal do Paraná
Cascavel, Brasil
andreaugustobortoli@gmail.com

Thiago Berticelli Ló
IFPR - Instituto Federal do Paraná
Cascavel, Brasil
thiago.lo@ifpr.edu.br

Darlon Vasata
IFPR - Instituto Federal do Paraná
Cascavel, Brasil
darlon.vasata@ifpr.edu.br

***Abstract*—Cybersecurity in Linux systems is a growing challenge due to the increasing frequency and sophistication of attacks. Traditional Intrusion Detection Systems (IDS), based on signature detection, have proven ineffective against new threats, prompting the search for more advanced solutions. This study proposes an intrusion detection module using machine learning techniques, combining Convolutional Neural Networks and Recurrent Neural Networks, to identify malicious patterns in system calls. The model was tested with the ADFA-LD dataset, achieving a recall of 97% and a precision of 95%. These results demonstrate the proposed approach's effectiveness in detecting complex attacks. However, the model still has a false negative rate of 17.97%, indicating the need for further improvements. Future work includes implementing the module in real-world environments and expanding testing with more diverse and heterogeneous datasets.**

***Keywords*—Cybersecurity; Machine Learning; system call analysis; anomaly detection; deep neural networks; operating systems.**

***Resumo*—A segurança cibernética em sistemas Linux é um desafio crescente devido ao aumento na frequência e sofisticação dos ataques. Os Sistemas de Detecção de Intrusão (IDS) tradicionais, baseados em assinaturas, mostram-se ineficazes contra novas ameaças, motivando a busca por soluções mais avançadas. Este estudo propõe um módulo de detecção de intrusões utilizando técnicas de aprendizado de máquina, combinando Redes Neurais Convolucionais e Redes Neurais Recorrentes, para identificar padrões maliciosos em chamadas de sistema. O modelo foi testado com o conjunto de dados ADFA-LD, alcançando uma revocação de 97% e uma precisão de 95%. Esses resultados demonstram a eficácia da abordagem proposta na detecção de ataques complexos. No entanto, o modelo ainda possui uma taxa de falsos negativos de 17,97%, indicando a necessidade de melhorias. Como trabalhos futuros, planeja-se implementar o módulo em ambientes reais e expandir os testes com bases de dados mais diversas e heterogêneas.**

***Palavras-chave*—Segurança cibernética; Aprendizado de Máquina; análise de chamadas de sistema; detecção de anomalias; redes neurais profundas; sistemas operacionais.**

## I. Introdução

A segurança cibernética emergiu como uma preocupação crítica caracterizada por um aumento significativo da frequência e da severidade dos ataques digitais [1]. Os sistemas Linux, amplamente adotados em ambientes corporativos e de infraestrutura crítica, são alvos recorrentes de intrusões maliciosas, uma vulnerabilidade atribuída à sua popularidade e à complexidade inerente de suas configurações de segurança [2].

A relevância desta problemática vai além dos desafios técnicos, impactando diretamente a vida de usuários comuns e a integridade de instituições. Incidentes de segurança de grande escala têm demonstrado repetidamente a vulnerabilidade de sistemas e a necessidade urgente de soluções de segurança mais eficazes. O vazamento de dados da empresa americana Rockyou, em 2009, que expôs 32 milhões de senhas de contas de usuários na plataforma, foi um marco na conscientização sobre a inadequação de práticas de armazenamento de senhas [3]. Mais recentemente, o vazamento de dados do Facebook em 2021 comprometeu informações pessoais de mais de 530 milhões de usuários em todo o mundo [4]. No contexto brasileiro, o vazamento de dados do Sistema Único de Saúde (SUS), em 2020, expôs informações sensíveis de aproximadamente 243 milhões de brasileiros [5]. Estes incidentes ressaltam a necessidade urgente de sistemas de segurança mais avançados e adaptáveis.

Neste cenário, os Sistemas de Detecção de Intrusão (IDS[1]) — que são definidos como dispositivos ou programas passivos que monitoram pontos de vulnerabilidade expostos para identificar tentativas de intrusão [6] — desempenham um papel fundamental na identificação e na mitigação de ameaças em tempo real [7], [8]. No entanto, os IDS tradicionais, baseados

[1]Do inglês *Intrusion Detection System* - Tradução livre.

em assinaturas, enfrentam limitações significativas, como a incapacidade de detectar ataques desconhecidos ou variantes dos ataques conhecidos e a necessidade constante de atualizações manuais [9].

Para enfrentar os desafios de segurança cibernética modernos, que estão em constante evolução, a implementação de técnicas de Aprendizado de Máquina (ML[2]), incluindo o Aprendizado Profundo (DL[3]), tem se mostrado promissora. Estas abordagens possibilitam o desenvolvimento de modelos capazes de aprender e reconhecer padrões complexos — inclusive em sequências de chamadas de sistema — aumentando significativamente a capacidade de detecção de ataques, especialmente variantes de ameaças existentes [7], [8], [10]–[14]. A eficácia desses modelos baseia-se na sua habilidade de generalizar a partir de dados de ataques conhecidos para identificar atividades maliciosas com padrões semelhantes, superando as limitações inerentes aos sistemas baseados em assinaturas [9].

Neste contexto de crescente necessidade de avanços em soluções de segurança, o presente trabalho tem como objetivo principal desenvolver um módulo de identificação de intrusão para sistemas Linux fundamentado em técnicas de ML. Este módulo foi projetado para operar de forma integrada ao sistema, analisando chamadas de sistema em tempo real para detectar intrusões em tempo real.

Os objetivos específicos do trabalho incluem:

1) Implementar e comparar diferentes algoritmos de ML, incluindo Redes Neurais Convolucionais (CNN[4]) e Redes Neurais Recorrentes (RNN[5]) para a detecção de intrusões nos sistemas monitorados;
2) Avaliar a eficácia do módulo proposto com base em métricas de precisão, revocação e taxa de falsos negativos.

Para garantir a relevância da solução proposta, a avaliação do módulo foi realizada utilizando o *Australian Defense Force Academy Linux Dataset* (ADFA-LD), amplamente utilizado na literatura como padrão de comparação em estudos de detecção de intrusões [15]–[17]. Os testes em ambientes Linux reais serão realizados como continuidade deste projeto, por meio da implementação de um *daemon* dedicado para integração do módulo.

## II. Trabalhos Relacionados

A evolução dos IDS tem sido marcada por avanços significativos nas últimas décadas, impulsionados pela crescente sofisticação das ameaças cibernéticas e pela necessidade de proteção mais eficaz dos sistemas computacionais [8], [18], [19]. Esta seção apresenta uma revisão dos trabalhos relacionados, abordando sistemas tradicionais, aplicações de aprendizado de máquina em IDS e métodos baseados na análise de chamadas de sistema.

### A. Sistemas de Detecção de Intrusão Tradicionais

Os IDS tradicionais, fundamentados principalmente em abordagens baseadas em assinaturas, representaram um marco inicial na defesa contra intrusões [7], [10]. Esses sistemas funcionam comparando padrões de tráfego ou comportamento com uma base de dados contendo assinaturas conhecidas de ataques. No entanto, as limitações dos sistemas baseados em assinaturas tornaram-se evidentes à medida que os ataques se tornaram mais complexos e variados [8], [9]. Buczak e Guven [9] destacam a ineficácia desses sistemas na detecção de variantes de ataques existentes ou modificados, uma vez que não há assinaturas predefinidas para esses novos padrões.

Um modelo alternativo de IDS tradicional foi proposto por Ilgun et al. em 1995 [20], que se baseia na análise de transição de estado. Este modelo baseia-se na modelagem de invasões como uma série de mudanças de estado, estabelecendo um paradigma influente no desenvolvimento de IDS [21], [22]. Embora esta abordagem permita uma representação mais dinâmica das intrusões, ela ainda enfrenta desafios na detecção de ataques que não seguem padrões previamente conhecidos.

### B. Aplicações de Aprendizado de Máquina em IDS

A aplicação de técnicas de ML e DL abriu novas fronteiras na detecção de intrusões, demonstrando potencial significativo para superar as limitações dos sistemas tradicionais [1], [9], [13], [19]. Diro e Chilamkurti [13] evidenciaram em 2018 a superioridade dos modelos de DL em comparação com abordagens tradicionais de ML, particularmente na detecção de ataques em ambientes de Internet das Coisas (IoT[6]). Seu estudo revelou uma melhoria significativa na taxa de detecção de ataques, com modelos de DL alcançando uma acurácia de 98,27% em comparação com 96,75% dos métodos tradicionais de ML.

Laghrissi et al. [1] exploraram o uso de RNNs, especificamente *Long Short-Term Memory* (LSTM), para detecção de intrusões, demonstrando melhorias significativas na acurácia de detecção. Utilizando o conjunto de dados KDD99, os modelos baseados em LSTM alcançaram uma acurácia de 99,44% para classificação binária e 99,39% para classificação multiclasse.

Vijayanand e Devaraj [16] propuseram um método de seleção de características baseado em algoritmo de otimização de baleias (WOA[7]) modificado, melhorando a eficácia de IDS em

[2]Do inglês *Machine Learning* - Tradução livre.
[3]Do inglês *Deep Learning* - Tradução livre.
[4]Do inglês *Convolutional Neural Networks* - Tradução livre.
[5]Do inglês *Recurrent Neural Networks* - Tradução livre.
[6]Do inglês *Internet of Things* - Tradução livre.
[7]Do inglês *Whale Optimization Algorithm* - Tradução livre.

redes *mesh* sem fio. Utilizando o conjunto de dados ADFA-LD, seu método alcançou uma taxa de detecção de ataques de 94,44%.

### C. IDS Baseados em Análise de Chamadas de Sistema

A análise de chamadas de sistema tem se mostrado uma abordagem promissora para Sistemas de Detecção de Intrusão Baseados no Hospedeiro (HIDS[8]), oferecendo dados valiosos sobre o comportamento de processos a nível de *kernel* [10], [12], [23]. O trabalho pioneiro de Forrest et al. [10] estabeleceu as bases para muitas pesquisas subsequentes, propondo um método de detecção de anomalias baseado em correlações de curto alcance nas chamadas de sistema.

Um desafio notável da análise de chamadas de sistema é o alto volume de dados gerados por processos em execução no sistema hospedeiro. Diante disso, Vyšniūnas et al. [23] propuseram um método de agrupamento de sequências de chamadas de sistema baseado em risco, atribuindo valores de risco a funções específicas. Esta abordagem resultou em melhorias significativas na acurácia de classificação, com um aumento de 87,6% para 96,8% em Máquinas de Vetores de Suporte (SVM[9]) e de 73,4% para 87,6% em Árvores de Decisão (DT[10]), em comparação com resultados anteriores obtidos utilizando os mesmos métodos e dados.

## III. Metodologia

Esta seção detalha as abordagens metodológicas adotadas para o desenvolvimento do módulo de detecção de intrusão proposto. A metodologia abrange a arquitetura do sistema no qual o módulo será inserido como contuidade deste trabalho e os algoritmos de DL utilizados no processo.

### A. Arquitetura do Sistema

A arquitetura proposta para a implementação do módulo de detecção de intrusão é ilustrada na Figura 1. Ela consiste em três camadas principais: a camada de aplicação, o sistema operacional com o mecanismo de interceptação e a camada de detecção de intrusão.

[8]Do inglês *Host-based Intrusion Detection System* - Tradução livre.
[9]Do inglês *Support-Vector Machine* - Tradução livre.
[10]Do inglês *Decision Tree* - Tradução livre.

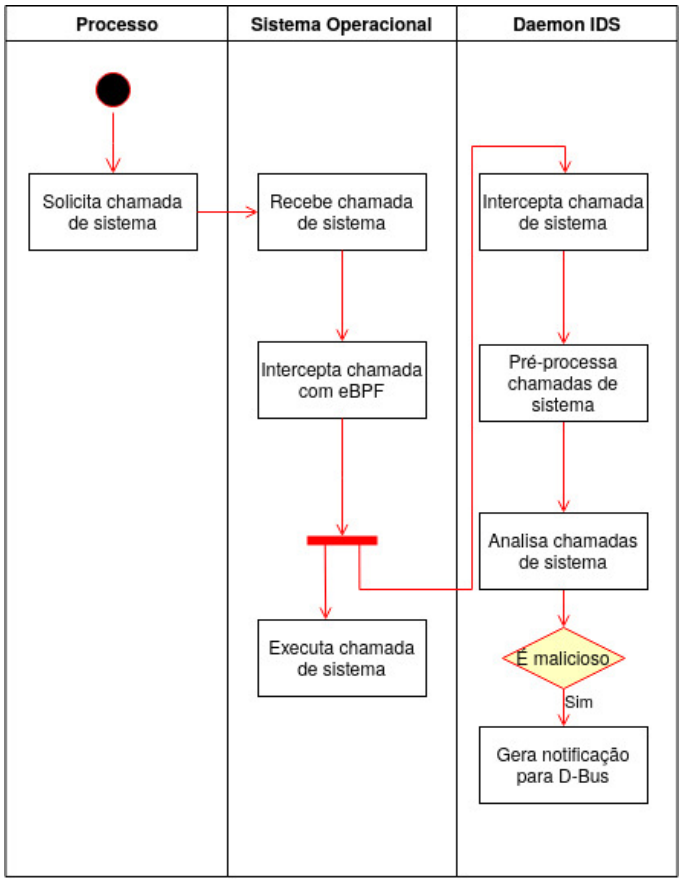


Fig. 1. Arquitetura do sistema proposto para detecção de intrusão, mostrando o fluxo de dados entre as camadas de aplicação, sistema operacional e detecção de intrusão.

Na primeira camada, processos de usuário iniciam e fazem requisições de chamadas ao sistema operacional. Na segunda camada, o sistema operacional recebe essas chamadas e utiliza o eBPF (*extended Berkeley Packet Filter*) para interceptá-las, realizando uma ramificação da execução. Esta ramificação permite que a chamada de sistema seja executada normalmente, enquanto, paralelamente, informações da chamada são enviadas ao *daemon* de detecção de intrusão na terceira camada.

O módulo de detecção de intrusão atua nesta terceira camada, na qual as chamadas de sistema interceptadas são pré-processadas e agrupadas em janelas deslizantes iterativas. Este pré-processamento organiza as sequências de chamadas de sistema em estruturas adequadas para serem analisadas pelo modelo de DL. O modelo atribui uma pontuação a cada sequência, indicando a probabilidade de comportamento malicioso. Durante a execução do processo, essas pontuações são monitoradas continuamente para avaliar se o comportamento observado é indicativo de uma intrusão. Caso o modelo identi-

fique uma atividade suspeita que ultrapasse um limiar definido, uma notificação é gerada para alertar os administradores sobre a possível ameaça.

### B. Conjunto de Dados

Para treinar e avaliar os modelos propostos, foi utilizado o *Australian Defense Force Academy Linux Dataset* (ADFA-LD), amplamente reconhecido e utilizado na literatura para a detecção de intrusões baseadas em chamadas de sistema em ambientes Linux [12]. O ADFA-LD contém sequências de chamadas de sistema provenientes de atividades normais e maliciosas, representando um ambiente operacional realista.

O conjunto de dados apresenta um desbalanceamento significativo entre as classes, com um total de 5.205 processos benignos e 756 processos maliciosos. Este desbalanceamento pode prejudicar o desempenho dos modelos de aprendizado de máquina, levando-os a serem tendenciosos em favor da classe majoritária [24].

Para mitigar o problema do desbalanceamento, foi empregado o algoritmo *Synthetic Minority Over-sampling Technique* (SMOTE) [25]. O SMOTE gera novas instâncias sintéticas da classe minoritária (maliciosa) interpolando entre os exemplos existentes, equilibrando a distribuição das classes.

A divisão dos dados foi realizada em 64% para treinamento, 16% para validação e 20% para teste. O SMOTE foi aplicado somente nos conjuntos de treinamento e validação, enquanto o conjunto de teste permaneceu inalterado.

Esta estratégia tem se mostrado eficaz em melhorar a capacidade dos modelos de aprender padrões da classe minoritária durante o treinamento, resultando em melhor desempenho na detecção de intrusões [26].

### C. Métricas Monitoradas

A avaliação do desempenho dos modelos de detecção de intrusão requer o uso de métricas que reflitam adequadamente sua capacidade de distinguir entre atividades normais e maliciosas. Conforme destacado por Axelsson [27], a presença de falsos positivos e falsos negativos pode afetar significativamente a eficácia operacional de um IDS, pois o fenômeno da falácia da taxa-base demonstra que, em ambientes com baixa frequência de intrusões, mesmo sistemas com alta taxa de detecção acabam gerando muitos alarmes falsos, comprometendo a confiança nas detecções e sobrecarregando os operadores com alertas irrelevantes.

Considerando este aspecto, as métricas monitoradas durante o treinamento incluem:

- **Acurácia**: A acurácia é uma medida de desempenho do modelo que calcula a proporção de previsões corretas em relação ao total de previsões feitas. Ela é definida como:

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN}$$

  onde:
  - $VP$: número de instâncias positivas corretamente classificadas (verdadeiros positivos),
  - $VN$: número de instâncias negativas corretamente classificadas (verdadeiros negativos),
  - $FP$: número de instâncias negativas incorretamente classificadas como positivas (falsos positivos),
  - $FN$: número de instâncias positivas incorretamente classificadas como negativas (falsos negativos).
- **Precisão**: A precisão indica a proporção de previsões positivas que são de fato corretas. Ela é definida como:

$$\text{Precisão} = \frac{VP}{VP + FP}$$

  onde:
  - $VP$: número de instâncias positivas corretamente classificadas (verdadeiros positivos),
  - $FP$ número de instâncias negativas incorretamente classificadas como positivas (falsos positivos).
- **Revocação**: A revocação mede a capacidade do modelo de identificar corretamente todas as instâncias positivas. A sua fórmula é:

$$\text{Revocação} = \frac{VP}{VP + FN}$$

  onde:
  - $VP$: número de instâncias positivas corretamente classificadas (verdadeiros positivos),
  - $FN$: número de instâncias positivas incorretamente classificadas como negativas (falsos negativos).
- **Área Sob a Curva ROC (AUC)**: A AUC mede a capacidade do modelo de distinguir entre classes positivas e negativas em diferentes limiares de decisão. Ela é a área sob a curva ROC, que representa graficamente a taxa de verdadeiros positivos em função da taxa de falsos positivos. Quanto mais a AUC se aproxima de 1, melhor é o desempenho de classificação do modelo.
- **Perda**: A perda, ou função de custo, calcula o erro médio entre as previsões do modelo e os valores reais. Durante o treinamento, o objetivo é minimizar a perda para que o modelo aprenda a fazer previsões mais precisas. A função de perda utilizada neste trabalho é a entropia cruzada binária[11], definida como:

$$\text{Perda} = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \ln 1 - p_i]$$

[11] A entropia cruzada binária é uma função de perda que mede a divergência entre as distribuições previstas pelo modelo e as distribuições reais das classes, sendo apropriada para problemas de classificação binária [28].

onde:

- $N$: número de exemplos no conjunto de dados,
- $y_i$: rótulo verdadeiro da instância $i$ (0 ou 1),
- $p_i$: probabilidade prevista pelo modelo para a classe positiva na instância $i$.

A escolha dessas métricas se justifica pela necessidade de avaliar não apenas a proporção de previsões corretas (acurácia), mas também a qualidade das previsões positivas (precisão), a capacidade do modelo de identificar corretamente as instâncias positivas (revocação) e a eficácia geral na distinção entre comportamentos normais e maliciosos (AUC).

### D. Algoritmos Utilizados

A escolha dos algoritmos de DL foi baseada nos resultados encontrados na literatura, que evidenciam a eficácia de modelos como CNNs e RNNs, especificamente 1D-CNN e LSTM, na detecção de intrusões em sistemas similares [1], [12], [17]. Nesta subseção, são apresentados os modelos empregados no módulo de detecção de intrusão, detalhando seu funcionamento e arquitetura.

*1) 1D-CNN:* As CNNs são modelos de DL que utilizam operações de convolução para extrair características de dados com estruturas locais, como sequências ou imagens [29]. A convolução é uma operação matemática que envolve o deslizamento de um filtro sobre os dados de entrada, permitindo a detecção de padrões locais. No contexto da detecção de intrusões, as CNNs podem identificar padrões em sequências unidimensionais de chamadas de sistema que correspondem a comportamentos maliciosos, mesmo que esses padrões não estejam imediatamente adjacentes.

O modelo 1D-CNN proposto para o módulo de detecção de intrusão é treinado utilizando o otimizador Adam e composto pelas seguintes camadas:

- **Camada de *Embedding***: converte as sequências de inteiros que representam as chamadas de sistema em vetores densos de dimensão fixa, permitindo uma representação vetorial das chamadas [30].
- **Camadas Convolucionais**: duas camadas convolucionais sequenciais aplicam filtros sobre as sequências de embeddings, realizando operações de convolução que extraem características locais relevantes. Cada camada é seguida por normalização em lote (*batch normalization*), *pooling* e *dropout* para estabilizar o treinamento e reduzir o risco de sobreajuste.
- **Camada de Atenção**: implementa um mecanismo de atenção que permite ao modelo enfatizar partes relevantes da sequência, destacando padrões significativos para a detecção [31].
- **Camada Densa**: após o achatamento da saída anterior, uma camada totalmente conectada processa as características extraídas.
- **Camada de Saída**: utiliza a função de ativação sigmoidal para produzir uma probabilidade entre 0 e 1, induzindo a classificação binária entre comportamento normal e malicioso.

*2) LSTM:* As RNNs são modelos projetados para lidar com dados sequenciais, pois possuem conexões recorrentes que permitem a transmissão de informações de etapas anteriores para etapas posteriores na sequência [32]. Essa recorrência permite que o modelo mantenha uma memória interna do que foi processado, tornando-as adequadas para tarefas como processamento de linguagem natural e séries temporais. No entanto, as RNNs tradicionais enfrentam dificuldades no aprendizado de dependências de longo prazo devido ao problema do desvanecimento ou explosão do gradiente, cujo resultado é a diminuição ou aumento exponencial dos parâmetros de entrada durante o processo de treinamento, dificultando a atualização efetiva dos pesos [33].

Para mitigar esse problema, foram introduzidas as unidades de LSTM, que incorporam portas de entrada, saída e esquecimento, permitindo o controle do fluxo de informações por meio da célula de memória [32]. Isso permite que as LSTMs capturem dependências de longo alcance nas sequências, mantendo informações relevantes por períodos mais longos.

O modelo LSTM proposto para o módulo de detecção de intrusão é treinado utilizando o otimizador Adam e composto pelas seguintes camadas:

- **Camada de *Embedding***: Representa as chamadas de sistema como vetores densos, similarmente ao modelo CNN.
- **Camada LSTM Bidirecional**: Processa a sequência em ambas as direções (anterior e posterior), combinando informações passadas e futuras para capturar contextos.
- **Camada de Atenção**: Implementa um mecanismo de atenção que permite ao modelo enfatizar partes relevantes da sequência, destacando padrões significativos para a detecção.
- **Camada Densa**: Após o achatamento, uma camada densa processa as características agregadas.
- **Camada de Saída**: Utiliza a função de ativação sigmoidal para a classificação binária.

## IV. Resultados e Discussão

Nesta seção são apresentados e discutidos os resultados experimentais obtidos a partir da implementação dos modelos de detecção de intrusão baseados em técnicas de aprendizado de máquina.

### A. *Desempenho dos Modelos*

Os modelos propostos foram avaliados utilizando o conjunto de dados ADFA-LD, com foco na capacidade de detectar sequências maliciosas de chamadas de sistema. Durante o treinamento, os dados foram balanceados utilizando o algoritmo SMOTE para mitigar o desbalanceamento entre classes.

As Figuras 2, 3 e 4 mostram os gráficos de acurácia, AUC, precisão, revocação e perda ao longo das épocas de treinamento para os melhores modelos 1D-CNN, LSTM e Mesclado, respectivamente.

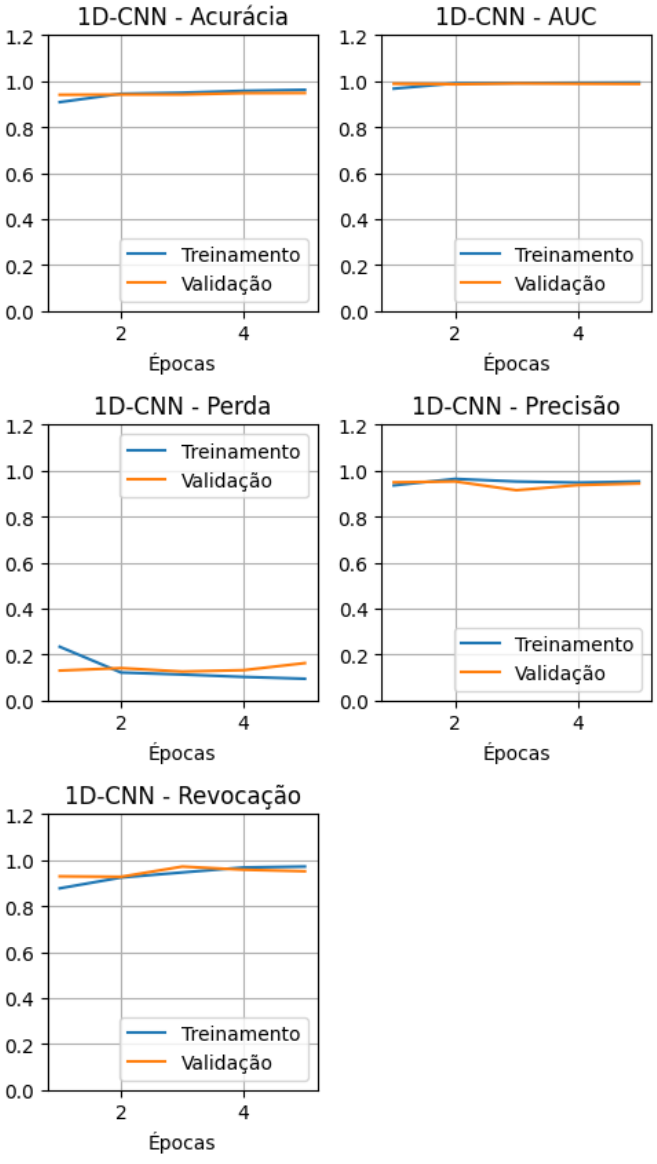


Fig. 2. Métricas de Desempenho para o Melhor Modelo 1D-CNN

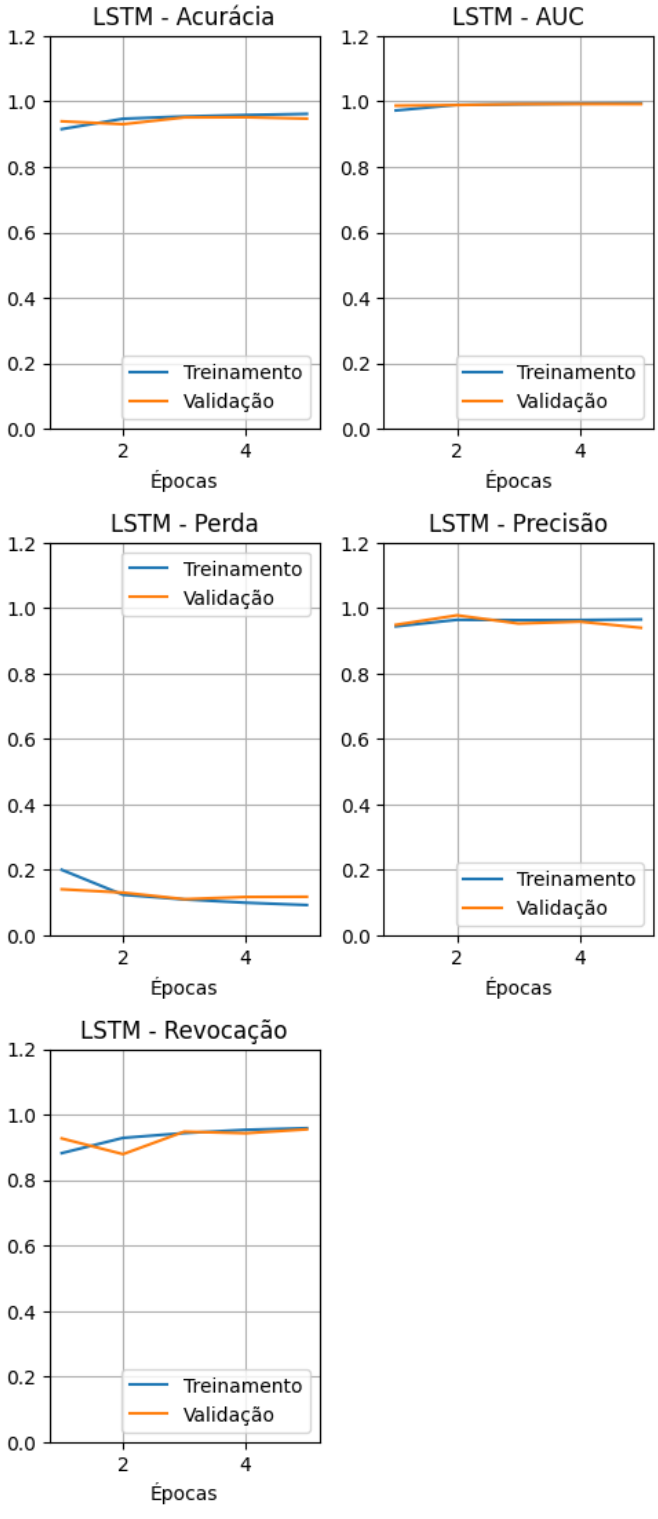


Fig. 3. Métricas de Desempenho para o Melhor Modelo LSTM

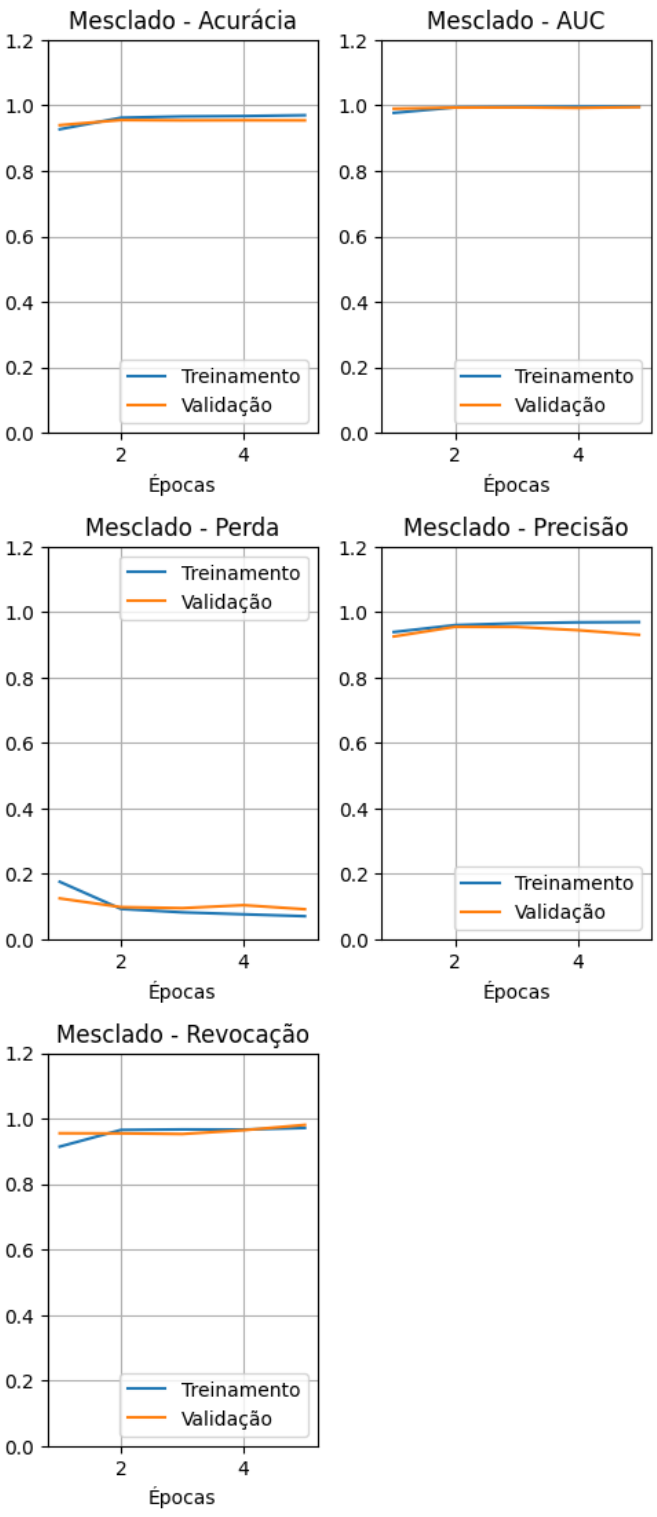


Fig. 4. Métricas de Desempenho para o Melhor Modelo Mesclado

Os gráficos indicam que todos os modelos mostram uma convergência estável durante o treinamento, com redução gradual da perda e melhoria das métricas de desempenho nos conjuntos de treinamento e validação.

No caso do modelo 1D-CNN (Figura 2), a acurácia atingiu aproximadamente 95% nos dados de validação, sugerindo que o modelo faz previsões corretas em grande parte dos casos. A precisão e a revocação indicam um equilíbrio entre a qualidade das previsões positivas e a capacidade de identificar eventos maliciosos, embora possa haver espaço para melhorias na detecção de falsos positivos e negativos.

Já o modelo LSTM (Figura 3) alcançou uma AUC de 99,19% nos dados de validação, o que sugere uma capacidade significativa de distinguir entre classes positivas e negativas em diferentes limiares de decisão. Com uma revocação de 95,48%, o modelo demonstrou boa capacidade de identificar instâncias maliciosas, apesar de a precisão ligeiramente menor sugerir a possibilidade de ocorrência de alguns falsos positivos.

Já o modelo LSTM (Figura 3) alcançou uma AUC de 99,19% nos dados de validação, o que sugere uma capacidade significativa de distinguir entre classes positivas e negativas em diferentes limiares de decisão. Com uma revocação de 95,48%, o modelo demonstrou boa capacidade de identificar instâncias maliciosas.

Por fim, o modelo Mesclado (Figura 4) obteve uma AUC de 99,44%, uma acurácia de 96% e uma revocação de 97,01% nos dados de validação. Esses resultados indicam que o modelo é eficaz na distinção entre comportamentos normais e maliciosos, sendo capaz de identificar corretamente a maior parte das intrusões e manter uma proporção elevada de previsões positivas corretas.

### *B. Sumário das Métricas de Validação*

A Tabela I apresenta um resumo das métricas de validação finais para os três modelos avaliados: 1D-CNN, LSTM e Mesclado. As métricas incluem acurácia, AUC, perda, precisão e revocação.

TABELA I
MÉTRICAS DE VALIDAÇÃO DOS MODELOS TESTADOS

| Modelo | Acurácia | AUC | Perda | Precisão | Revocação |
|---|---|---|---|---|---|
| 1D-CNN | 94,81% | 98,77% | 16,24% | 94,37% | 95,14% |
| LSTM | 94,75% | 99,19% | 11,78% | 93,96% | 95,48% |
| Mesclado | 96,00% | 99,44% | 09,06% | 94,99% | 97,01% |

Embora o modelo 1D-CNN tenha apresentado alta acurácia, é importante analisar esse resultado com cautela, pois em contextos de segurança cibernética o impacto de falsos negativos (falha em detectar uma intrusão) é crítico.

O modelo LSTM apresentou uma precisão ligeiramente menor na validação, mas isso não significa necessariamente que ele gerará um número maior de falsos positivos no conjunto de teste desbalanceado.

O modelo Mesclado superou os demais em termos de AUC e revocação, indicando uma capacidade superior de distinguir comportamentos normais de comportamentos maliciosos nos dados de validação balanceados.

### C. Matrizes de Confusão

Para avaliar a capacidade dos modelos em um cenário mais realista, eles foram testados no conjunto de teste original, que é desbalanceado e não foi alterado pelo SMOTE. As matrizes de confusão apresentadas na Tabela II refletem o desempenho dos modelos nesse conjunto.

TABELA II
MATRIZES DE CONFUSÃO DOS MODELOS TESTADOS

| Modelo | Classe | Predito Positivo | Predito Negativo |
|---|---|---|---|
| 1D-CNN | Positivo | 73 | 94 |
| | Negativo | 11 | 1013 |
| LSTM | Positivo | 62 | 105 |
| | Negativo | 10 | 1014 |
| Mesclado | Positivo | 137 | 30 |
| | Negativo | 24 | 1000 |

Os resultados das matrizes de confusão mostram que o modelo Mesclado apresenta o maior número de verdadeiros positivos (137) e o menor número de falsos negativos (30), indicando uma melhor capacidade de identificar comportamentos maliciosos. No entanto, ele também apresentou um número maior de falsos positivos (24) em comparação com os modelos 1D-CNN (11) e LSTM (10), o que pode levar a mais alarmes falsos em um ambiente operacional.

O modelo LSTM, embora tenha identificado menos verdadeiros positivos (62) e tenha um maior número de falsos negativos (105), teve o menor número de falsos positivos (10), sugerindo uma maior precisão na identificação de atividades normais.

Para uma análise mais detalhada, foram calculadas as taxas de falsos negativos (TFN) e falsos positivos (TFP) para cada modelo:

- **Modelo Mesclado**:
  - TFN = $\frac{30}{30+137} \approx 17,97\%$
  - TFP = $\frac{24}{24+1000} \approx 2,34\%$
- **Modelo LSTM**:
  - TFN = $\frac{105}{105+62} \approx 62,87\%$
  - TFP = $\frac{10}{10+1004} \approx 0,99\%$
- **Modelo 1D-CNN**:
  - TFN = $\frac{94}{94+73} \approx 56,29\%$
  - TFP = $\frac{11}{11+1013} \approx 1,07\%$

Esses resultados indicam que o modelo Mesclado tem a menor taxa de falsos negativos, mas uma taxa de falsos positivos ligeiramente maior. O modelo LSTM tem a menor taxa de falsos positivos, mas a maior taxa de falsos negativos, o que pode ser problemático em detecção de intrusão, considerando que neste contexto é crucial identificar o máximo possível de atividades maliciosas. O modelo 1D-CNN apresenta taxas intermediárias em ambas as métricas.

Portanto, para trabalhos futuros, pretende-se focar em aprimorar o modelo Mesclado para reduzir tanto os falsos negativos quanto os falsos positivos, possivelmente explorando o uso de bases de dados maiores e mais diversificadas, como o *Leipzig Intrusion Detection - Data Set* (LID-DS) 2021, técnicas de pré-processamento adicionais e uma maior diversidade de algoritmos.

## V. CONCLUSÃO

Neste trabalho, foi desenvolvido um módulo de detecção de intrusão para sistemas Linux utilizando técnicas de aprendizado de máquina, com ênfase em Redes Neurais Convolucionais (CNN) e Redes Neurais Recorrentes (RNN). Embora os resultados obtidos indiquem que o uso de modelos de aprendizado profundo, especialmente o modelo mesclado que combina características de CNNs e LSTMs, tenha um desempenho promissor na detecção de intrusões, há ainda um espaço considerável para melhorias.

Apesar de o modelo ter demonstrado um equilíbrio entre acurácia, taxa de detecção e taxa de falsos negativos, a presença de 17,97% de falsos negativos sugere que o sistema ainda pode falhar em detectar algumas ameaças críticas e gerar alertas incorretos. Essas limitações apontam para a necessidade de melhorias adicionais para aumentar a robustez e a precisão do modelo, tornando-o mais confiável para uso em ambientes reais.

Uma das abordagens para superar essas limitações envolve a realização de testes com bases de dados maiores e menos homogêneas, como o LID-DS 2021, que pode fornecer uma visão mais abrangente e permitir a captura de padrões de ataque mais complexos. Além disso, o uso de técnicas de pré-processamento além da janela deslizante pode contribuir para melhorar a eficácia do modelo.

### A. Trabalhos Futuros

Para trabalhos futuros, planeja-se implementar o módulo de detecção em um *daemon* de código aberto e realizar testes em ambientes Linux reais, com o objetivo de avaliar o desempenho

do sistema em cenários práticos e controlados, otimizando métricas como o uso de CPU e memória.

Além disso, considerando que o modelo ainda apresenta uma taxa considerável de falsos positivos e negativos, a pesquisa deverá continuar com foco na minimização dessas métricas. Algumas abordagens que poderão ser exploradas incluem o uso de bases de dados maiores e mais diversificadas, como o LID-DS 2021, que oferece um conjunto de dados mais heterogêneo; a exploração de técnicas de pré-processamento adicionais; e a avaliação de uma maior diversidade de algoritmos, fatores que podem contribuir para aumentar a robustez do sistema em diferentes cenários de ataque.

Por fim, a investigação de dados adicionais, como logs de rede e comandos de usuário, poderá aumentar a capacidade do sistema de identificar intrusões que utilizem diferentes vetores de ataque. Dessa forma, espera-se que o sistema se torne mais adaptável a ambientes dinâmicos.

## Agradecimentos

Os autores agradecem ao Instituto Federal do Paraná (IFPR) pelo suporte institucional e aos amigos e familiares pelas valiosas discussões e sugestões que contribuiram para o desenvolvimento deste trabalho.

## Referências

[1] F. E. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, "Intrusion detection systems using long short-term memory (lstm)," *Journal of Big Data*, vol. 8, 2021.

[2] G. Amarchand, P. Brown, and T. Mahoney, "Linux security," *Advances in Engineering Innovation*, vol. 2, pp. 17–20, 10 2023.

[3] A. Imperva, "Consumer password worst practices," 2010. [Online]. Available: https://www.imperva.com/docs/gated/WP_Consumer_Password_Worst_Practices.pdf

[4] T. Hunt, "Have i been pwned?" 2019. [Online]. Available: https://haveibeenpwned.com/PwnedWebsites

[5] G1, "Nova falha do ministério da saúde expõe dados de 243 milhões de brasileiros na internet, diz jornal," 2020. [Online]. Available: https://g1.globo.com/economia/tecnologia/noticia/2020/12/02/nova-falha-do-ministerio-da-saude-expoe-dados-de-243-milhoes-de-brasileiros-na-internet-diz-jornal.ghtml

[6] C. P. Pfleeger, *Security in computing 5th Edition*. Prentice-Hall, Inc., 2015.

[7] S. A. V. Jatti and V. J. K. Sontif, "Intrusion detection systems," *International Journal of Recent Technology and Engineering*, vol. 8, pp. 3976–3983, 9 2019.

[8] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, 2019.

[9] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys and Tutorials*, vol. 18, 2016.

[10] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "Sense of self for unix processes," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1996.

[11] M. Xie and J. Hu, "Evaluating host-based anomaly detection systems: A preliminary analysis of adfa-ld," *Proceedings of the 2013 6th International Congress on Image and Signal Processing, CISP 2013*, vol. 3, pp. 1711–1716, 2013.

[12] G. Creech and J. Hu, "A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns," *IEEE Transactions on Computers*, vol. 63, pp. 807–819, 2014.

[13] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018. [Online]. Available: http://dx.doi.org/10.1016/j.future.2017.08.043

[14] T. Lu, X. Liu, J. Chen, N. Hu, and B. Liu, "Afcgdroid: Deep learning based android malware detection using attributed function call graphs," *Journal of Physics: Conference Series*, vol. 1693, 2020.

[15] B. S. Khater, A. W. B. A. Wahab, M. Y. I. B. Idris, M. A. Hussain, and A. A. Ibrahim, "A lightweight perceptron-based intrusion detection system for fog computing," *Applied Sciences (Switzerland)*, vol. 9, 2019.

[16] R. Vijayanand and D. Devaraj, "A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network," *IEEE Access*, vol. 8, 2020.

[17] Z. Wang, Y. Liu, D. He, and S. Chan, "Intrusion detection methods based on integrated deep learning model," *Computers and Security*, vol. 103, 2021.

[18] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments," *Energy Reports*, vol. 7, 2021.

[19] H. Satilmis, S. Akleylek, and Z. Y. Tok, "A systematic literature review on host-based intrusion detection systems," *IEEE Access*, vol. 12, 2024.

[20] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: A rule-based intrusion detection approach," *IEEE Transactions on Software Engineering*, vol. 21, 1995.

[21] Z. Liu, S. M. Bridges, and R. B. Vaughn, "Combining static analysis and dynamic learning to build accurate intrusion detection models," in *Proceedings - 3rd IEEE International Workshop on Information Assurance, IWIA 2005*, 2005.

[22] K. S. Ganesh, M. R. Sekar, and V. Vaidehi, "Semantic intrusion detection system using pattern matching and state transition analysis," in *International Conference on Recent Trends in Information Technology, ICRTIT 2011*, 2011.

[23] T. Vyšniūnas, D. Čeponis, N. Goranin, and A. Čenys, "Risk-based system-call sequence grouping method for malware intrusion detection," *Electronics (Switzerland)*, vol. 13, 1 2024.

[24] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, 2009.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, 2002.

[26] A. Fernández, S. García, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," 2018.

[27] S. Axelsson, "Base-rate fallacy and its implications for the difficulty of intrusion detection," in *Proceedings of the ACM Conference on Computer and Communications Security*. ACM, 1999, pp. 1–7.

[28] P. B. Le and Z. T. Nguyen, "Roc curves, loss functions, and distorted probabilities in binary classification," *Mathematics*, vol. 10, 2022.

[29] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, 2021.

[30] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," in *Journal of Machine Learning Research*, vol. 3, 2003.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017.

[32] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, 2020.

[33] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *30th International Conference on Machine Learning, ICML 2013*, 2013.