MARIA: Monitoramento e Análise para Resposta Imediata a Ataques à Rede 5G no Contexto da IoT

Cleitianne Silva[1], **Carina Oliveira**[2], **Rossana Andrade**[1*]

[1]Universidade Federal do Ceará (UFC)

[2]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

cleitianne@alu.ufc.br, carina@lar.ifce.edu.br, rossana@dc.ufc.br

Abstract. *The integration of IoT with 5G networks expands the number of connected devices and the complexity of traffic, intensifying security challenges. Machine Learning techniques have been used to detect attack patterns in networks. This article presents MARIA, a solution developed to identify and propose mitigation measures for attacks targeting IoT devices in 5G networks. MARIA consists of six modules and, as part of its process, employs supervised algorithms for attack detection, enabling rapid responses. An evaluation of MARIA is conducted in a testbed. The results highlight MARIA's effectiveness in real-time detection of different types of attacks.*

Resumo. *A integração da IoT com as redes 5G amplia o número de dispositivos conectados e a complexidade do tráfego, intensificando desafios de segurança. Técnicas de Aprendizado de Máquina têm sido utilizadas para detectar padrões de ataques em redes. Este artigo apresenta o MARIA, uma solução desenvolvida para identificar e propor medidas de mitigação para ataques direcionados a dispositivos IoT em redes 5G. O Maria é composto por seis módulos e, como parte de seu processo, utiliza algoritmos supervisionados para detecção de ataques, viabilizando respostas rápidas. Uma avaliação do Maria é realizada em um testbed. Os resultados evidenciam a eficácia do MARIA na detecção em tempo real de diferentes tipos de ataques.*

1. Introdução

As redes 5G representam um marco tecnológico com o potencial de transformar a interconexão entre dispositivos e pessoas. No Brasil, embora a implementação completa das especificações do 5G ainda esteja em andamento, seus benefícios já são amplamente reconhecidos, incluindo maior cobertura, mobilidade, densidade de conexões, eficiência energética e menor latência [Nkosi and Mathonsi 2024]. Além disso, essa nova geração de redes foi projetada para viabilizar uma ampla gama de casos de uso emergentes, como veículos autônomos, telepresença holográfica, realidade estendida imersiva, automação na Indústria 4.0, telemedicina remota e a Internet das Coisas (IoT) [Farzaneh et al. 2024].

Paralelamente, a IoT se consolidou como uma evolução transformadora da Internet, ampliando suas aplicações e diversificando a natureza e a escala dos dados gerados. Essa tecnologia conecta dispositivos físicos ao mundo digital, possibilitando a

*Bolsista de Produtividade Desen. Tec. e Extensão Inovadora do CNPq - Nível 1D

comunicação entre dispositivos e pessoas por meio de sensores e atuadores integrados. Ao possibilitar maior eficiência operacional, automação de processos e decisões baseadas em dados, a IoT tem desempenhado um papel crucial na modernização de setores como saúde, indústria, agricultura e cidades inteligentes [Yang and Zhang 2023]. A convergência do 5G e da IoT cria um ecossistema inovador que integra bilhões de dispositivos em uma infraestrutura robusta e escalável, possibilitando avanços significativos em diversas aplicações. No entanto, essa integração também amplia significativamente os desafios relacionados à segurança cibernética.

Muitos dispositivos IoT são projetados com recursos limitados de memória e processamento, o que os torna especialmente vulneráveis a ataques maliciosos. Essa fragilidade facilita a criação de *botnets*, amplamente utilizadas em ataques de *Distributed Denial of Service* (DDoS) em larga escala [Saheed et al. 2022]. Um exemplo notório foi o ataque do malware Mirai, que explorou essas vulnerabilidades para comprometer mais de 1 milhão de dispositivos, incluindo câmeras de segurança e roteadores domésticos, para criar uma enorme *botnet* usada para realizar ataques DDoS que afetaram grandes plataformas como Netflix, CNN e Twitter [Kumari and Jain 2023].

Esses riscos são agravados pela falta de padronização em protocolos de segurança para IoT, amplificando as vulnerabilidades quando esses dispositivos são integrados a redes 5G [Pakmehr et al. 2024]. A aplicação de aprendizado de máquina (*Machine Learning* - ML) para identificar anomalias e ameaças em ambientes IoT é importante para uma manutenção proativa, a fim de garantir a segurança dessa infraestrutura. Os modelos de ML são capazes de aprender padrões e tomar decisões relacionadas a intrusões, permitindo a construção de sistemas automatizados de detecção que superam as limitações de tempo e custo. Além disso, o desempenho generalizado desses sistemas pode ser mantido mesmo diante de novos padrões de ataque, o que reforça o crescente interesse na automação da detecção de intrusão em redes 5G utilizando ML [Kim et al. 2022].

Embora técnicas baseadas em ML tenham se mostrado promissoras na detecção de ataques, a análise em tempo real do grande volume de dados gerados por dispositivos IoT ainda representa um desafio significativo [Kim et al. 2022]. No caso específico da detecção de ataques DDoS em redes 5G, a limitação está no processamento, pois o uso dos recursos disponíveis como entrada pode resultar em modelos de ML com desempenho insatisfatório, além de aumentar o tempo necessário para treinamento e detecção.

Nesse cenário, o MARIA surge como uma solução para o monitoramento e análise de vulnerabilidades em redes 5G, com foco na identificação proativa de ameaças. Com seis módulos especializados, o MARIA foi projetado para enfrentar os desafios de segurança em redes 5G integradas ao IoT, sendo capaz de identificar e recomendar medidas de mitigação contra seis tipos de ataques cibernéticos. No seu processo de análise, o MARIA utiliza algoritmos supervisionados para a detecção de ataques, viabilizando respostas rápidas. Além de sua capacidade de detectar anomalias em tempo real, o MARIA se diferencia por enviar notificações imediatas sobre as ameaças detectadas. Dessa forma, o MARIA fortalece a proteção da infraestrutura e contribui para um ambiente IoT mais seguro e resiliente.

Para validar sua eficácia, foi aplicado um conjunto de técnicas e algoritmos de ML a uma base de dados rotulada, com o objetivo de identificar anomalias e ameaças. O expe-

rimento adotou validação cruzada como estratégia para garantir uma avaliação robusta do desempenho do modelo preditivo. No total, seis algoritmos de ML foram testados nesse processo. Dentre os algoritmos analisados, o CatBoost se destacou, alcançando 90,72% de acurácia, 93,61% de precisão, 90,72% de *recall* e 90,24% de *F1-score*. Com base nessas métricas, foi realizada a predição dos ataques. Em seguida, o desempenho do MARIA foi avaliado em um ambiente de *testbed*, considerando não apenas sua capacidade de prever ataques, mas também a sugestão de contramedidas e o envio de notificações em tempo real ao administrador da rede, permitindo uma resposta rápida e proativa. Os resultados obtidos demonstram a eficiência do modelo no contexto proposto, reforçando o potencial do MARIA como uma solução avançada para segurança em redes 5G integradas ao IoT.

No restante do artigo, a Seção 2 apresenta os trabalhos relacionados. A Seção 3 fornece uma visão geral do MARIA, detalhando os módulos que compõem sua concepção. A Seção 4 discute a avaliação do MARIA. A Seção 5 inclui a conclusão e os trabalhos futuros. Por fim, a Seção 6 fornece a disponibilidade de artefatos.

2. Trabalhos Relacionados

O estudo de [Nkosi and Mathonsi 2024] apresenta um algoritmo de segurança híbrido para proteger o ecossistema 5G-IoT, integrando fatiamento de rede 5G com redes neurais em um modelo matemático robusto. A solução emprega técnicas de detecção baseadas em conhecimento e correspondência de padrões para identificar e mitigar ameaças tanto em fatias de rede quanto em dispositivos IoT. Combina técnicas como DQN, Double Q, distribuição de Poisson e SAWS, sendo que o SAWS mitiga o impacto de ataques em duas etapas: primeiro, desabilitando a fatia de rede afetada e, em seguida, garantindo a continuidade do serviço por meio da realocação de tarefas do dispositivo IoT para outras fatias da rede. No entanto, os autores não exploram a aplicação dessas técnicas em cenários de detecção em tempo real.

O estudo de [Kumar et al. 2023] aborda redes multilocatárias 5G e Wi-Fi avançadas, projetadas para transmissões de alta velocidade e baixa latência, atendendo à crescente demanda por essas características. Os autores apresentam um algoritmo que realiza monitoramento contínuo da rede e adapta os padrões de tráfego, permitindo a identificação e mitigação de ataques DDoS em tempo real. A solução proposta inclui uma estratégia de filtragem de tráfego, que identifica e bloqueia visitantes maliciosos, enquanto gerencia o fluxo para evitar congestionamentos, garantindo maior desempenho e resiliência da rede. Além disso, os autores empregam técnicas de ML para categorização, o que acelera a identificação de ataques e permite respostas mais ágeis.

No trabalho de [Örs et al. 2021], são apresentados dois modelos de aprendizado de máquina supervisionado implantados em nós de borda para detecção de anomalias e classificação de ataques. O primeiro modelo, baseado em XGBoost, detecta anomalias e classifica ataques independentemente dos dispositivos utilizados, enquanto o segundo, que utiliza árvores extremamente randomizadas, é especializado em ataques Wi-Fi. Os modelos identificam tráfego benigno e seis tipos de ataques. Além disso, são escaláveis e compatíveis com dados criptografados, permitindo a detecção de intrusões sem comprometer a privacidade do usuário. No entanto, diferentemente do presente estudo, que além de sugerir medidas de mitigação, tem foco específico em redes 5G para cenários de IoT.

O artigo de [Aljuhani 2021] apresenta uma visão abrangente sobre o uso de

técnicas de ML e aprendizado profundo (DL) para mitigar ataques DDoS. Os autores analisam sistemas de defesa baseados em ML/DL em ambientes como computação em nuvem, SDN e NFV, classificam os tipos comuns de ataques DDoS e apresentam uma taxonomia detalhada. Além disso, exploram ataques em ambientes IoT e técnicas de mitigação como bloqueio de portas, rastreamento de IP, Honeypot, limitação de taxa e filtragem de assinatura. O estudo identifica desafios na detecção de ataques, como a falta de conjuntos de dados específicos e a limitação de métodos que reagem apenas após o início dos ataques, quando danos já foram causados. Contudo, [Aljuhani 2021] não aborda aplicação dessas técnicas em redes 5G com dispositivos IoT.

3. Visão Geral do MARIA

O MARIA tem como principal objetivo monitorar e sugerir medidas para a neutralização de ameaças em redes 5G no contexto da IoT. A Figura 1 ilustra sua arquitetura, composta por seis módulos.

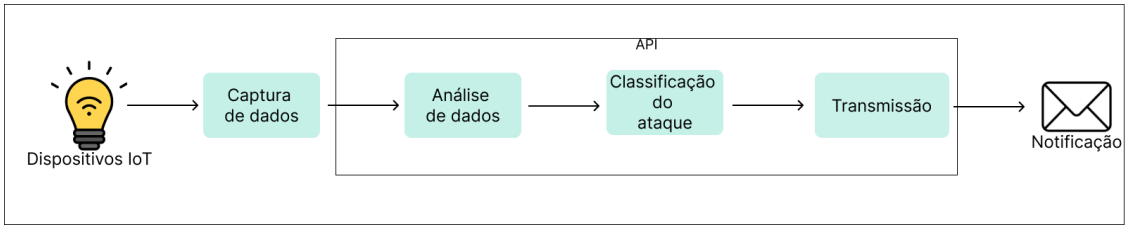


Figura 1. Arquitetura do MARIA.

O módulo **Dispositivos IoT** evidencia o crescimento do número de dispositivos conectados, com estimativas indicando que esse número atingirá 30,9 bilhões até 2025 [Nascita et al. 2022]. Esse aumento acelerado resulta em um tráfego de rede cada vez mais intenso. Nesse contexto, o primeiro módulo, responsável pelo gerenciamento dos dispositivos IoT conectados à rede 5G, processa um fluxo contínuo de dados. No entanto, esses dispositivos geralmente apresentam restrições de recursos, como limitações de CPU, consumo de energia e capacidade de bateria, o que os torna vulneráveis a ataques cibernéticos. Devido a essas limitações, ataques como DDoS exploram o tráfego gerado por esses dispositivos para comprometer serviços e degradar o desempenho da rede. A ampla disseminação e a conectividade massiva dos dispositivos IoT favorecem sua utilização em *botnets* maliciosas, tornando-os alvos estratégicos para ciberataques.

O módulo de **Captura de dados** monitora o tráfego da rede e captura os dados transmitidos pelos dispositivos IoT, encaminhando-os para o módulo de análise dos dados. Durante esse processo, os pacotes que trafegam na rede são coletados e analisados, permitindo a extração de informações essenciais. Com essas informações, os pacotes são submetidos à `API` de predição que contém o melhor modelo mais eficiente para classificá-los como benignos ou maliciosos.

No módulo **Análise de dados**, os dados capturados passam por um processo de tratamento e preparação para o modelo preditivo. Nessa etapa, são aplicadas técnicas de pré-processamento, incluindo o tratamento de dados não numéricos e a correção de valores ausentes, garantindo que os pacotes processados pelo módulo de captura estejam adequados para análise.

Após a adequação dos dados, o módulo de **Classificação do ataque** utiliza um modelo preditivo previamente treinado para classificar os pacotes como benignos ou pertencentes a uma das cinco categorias de ataques.

O módulo de **Transmissão** interpreta a classificação gerada e executa as ações apropriadas com base nos resultados. Esse módulo está integrado a um banco de dados que armazena os pacotes transmitidos na rede, assim como a classificação predita para cada um deles. Esse registro possibilita o monitoramento contínuo dos ataques identificados e auxilia na obtenção de uma visão abrangente do tráfego na rede.

Por fim, ao detectar um possível ataque, o módulo de **Notificação** é acionado, enviando alertas aos administradores da rede sobre a ameça em andamento. Além disso, essas notificações incluem sugestões de contramedidas adequadas para cada tipo de ataque analisado neste estudo.

4. Avaliação do MARIA

Nesta seção, apresentamos a avaliação detalhada de cada módulo do MARIA, destacando seus resultados, desafios encontrados e melhorias sugeridas.

4.1. Módulos 1 e 2

O módulo de **Dispositivos IoT** é responsável por enviar pacotes de dados contendo informações sobre o tráfego de rede ao segundo módulo, **Captura dos Dados**, que envolve a captura dos dados enviados pelos dispositivos IoT.

Nesta etapa de coleta, utilizamos a biblioteca *Scapy*, implementada em *Python*. *Scapy* é uma biblioteca interativa de manipulação de pacotes, capaz de decodificar grandes números de protocolos, enviá-los pela rede, capturá-los e corresponder solicitações e respostas. Para capturar o tráfego de dados em tempo real, utiliza-se a função `sniff()`. Dessa forma, os pacotes que trafegam na rede são capturados e analisados, permitindo a coleta dos dados necessários. Com essas informações, os pacotes são submetidos à `API` de predição que contém o melhor modelo para classificar o pacote como benigno ou malicioso. Para treinar e validar o modelo com dados representativos do ambiente 5G-IoT, foi essencial construir um conjunto de dados adequado.

4.1.1. Características do Conjunto de Dados

Para avaliar a eficácia dos mecanismos de segurança adequadamente, é imprescindível dispor de um conjunto de dados. Devido à lacuna na literatura referente a conjuntos de dados com diferentes tipos de ataques em um ambiente IoT dentro de uma rede 5G, tronou-se necessário construir um novo *dataset* a partir de bases de dados já existentes. Especificamente, realizamos a junção de um *dataset* 5G e um *dataset* IoT. Todos os procedimentos para unificar esses conjuntos de dados foram cuidadosamente executados, resultando em um novo *dataset* que incorpora seis tipos diferentes de ataques. Dessa forma, a combinação do IoT-23 [Garcia et al. 2021] com o 5G-NIDD [Samarakoon et al. 2022] resulta em um *dataset* integrado, que oferece uma diversidade de dados, abrangendo tanto ambientes IoT quanto 5G, facilitando a pesquisa e o teste de soluções de segurança relacionados a rede 5G em cenário de IoT.

Após a unificação das diferentes bases de dados, a estrutura resultante consistiu em um total de 10 atributos. Destes, 8 atributos foram designados como variáveis independentes ou *features* de entrada, enquanto os 2 atributos restantes foram considerados rótulos. Especificamente, esses rótulos indicam se um pacote faz parte de um ataque (representado pelo atributo `Label`) e qual é o tipo de ataque (identificado pelo atributo `AttackType`). Dos 10 atributos, 7 são numéricos e 3 não numéricos, sendo esses os atributos `Proto`, que representa o protocolo utilizado, e as duas classes `Label` e `AttackType`. O *dataset* gerado resultou em um total de 58.175.447 registros distribuídos em `Malicious` um total de 49122106 e, `Benign` com 9053334.

4.2. Módulo 3

Os módulos subsequentes **Análise, Classificação do Tráfego, Transmissão e Notificação** estão integrados em uma única `API`. Esta `API` foi desenvolvida utilizando o framework *Flask*[1], e contém um endpoint denominado `/predict (POST)`, que segue o modelo Curl do contrato da `API` e opera por meio do protocolo `HTTP`. A `API` recebe parâmetros com base nas *features* dos modelos. Após a recepção dos dados, realizou-se o processo de validação e pré-processamento para garantir que estes dados estejam prontos para serem submetidos ao modelo. Isso inclui o tratamento para ajustar os dados ao formato esperado pelo modelo. Particular atenção foi dada ao tratamento de valores não numéricos, como o atributo `proto`, bem como ao manejo de valores ausentes. Após essa análise e tratamento, os dados foram submetidos ao modelo de ML presente na API, a fim de classificar o dado recebido.

4.2.1. Pré-processamento

O pré-processamento dos dados tem como principal finalidade tratar o conjunto de dados para otimizar o desempenho dos algoritmos avaliados. Diversas técnicas de pré-processamento foram aplicadas, incluindo a seleção de ataques, o tratamento de valores ausentes e não numéricos, bem como o balanceamento e o particionamento dos dados para treinamento e validação dos modelos. Essas etapas são importantes para garantir que o *dataset* esteja em condições ideais para a modelagem e análise subsequente, permitindo que os algoritmos de ML alcancem o máximo do seu desempenho.

Embora o *dataset* seja volumoso, observa-se uma discrepância nos valores contidos na coluna `AttackType`. Diante disso, foram selecionados os seis principais `AttackType`, considerando tanto registros benignos quanto tipos de ataques, em termos de volume de dados. Essa seleção visa otimizar a precisão dos algoritmos que serão utilizados para treinamento, garantindo que as análises se concentrem nos dados mais relevantes.

Em seguida, procedeu-se à separação dos dados de entrada e valores alvo, um procedimento usual para viabilizar o treinamento dos modelos. Além disso, foi necessário realizar o tratamento de valores ausentes. Para isso, utilizou-se a classe `SimpleImputer` [2] do pacote `sklearn` do *Python*, que emprega estatísticas descritivas para substituir va-

[1]Flask é um micro-framework escrito em *Python* que é amplamente utilizado para criar `APIs` leves e aplicações web.

[2]https://scikitlearn.org/1.5/modules/generated/sklearn.impute.SimpleImputer.html

lores faltantes. Neste trabalho, foi adotada a estratégia da média, que substitui os valores ausentes pela média dos demais valores da coluna. A etapa seguinte envolveu o tratamento de valores não numéricos.

Após a seleção dos valores mais relevantes, ainda persistiu um grande número de desbalanceamento entre as classes. Para abordar essa questão, decidiu-se utilizar uma técnica de balanceamento. Para realizar o balanceamento, foi empregada a classe `RandomUnderSampler`[3], que iguala as classes aleatoriamente com base na classe com o menor valor. A Figura 2 apresenta os valores após o processo de balanceamento.

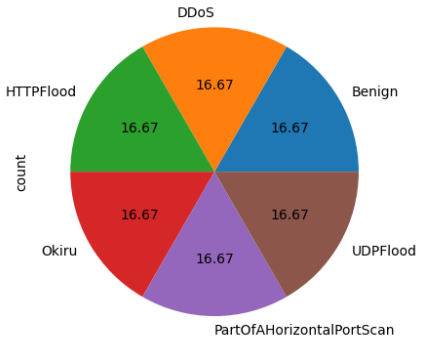


Figura 2. Total Final do atributo `AttackType` após balanceamento

Por fim, realizou-se a divisão entre os conjuntos de treino e teste, adotando a metodologia 80/20. Esta abordagem consiste em separar 80% dos dados para o treinamento dos modelos e 20% para os testes e validação. A divisão foi efetuada utilizando a classe `train_test_split` da biblioteca *sklearn*[4]. Com esses passos, os dados estavam preparados para serem utilizados no treinamento dos algoritmos, visando encontrar o modelo com melhor desempenho na classificação de ataques.

4.2.2. Treinamento

Após o pré-processamento dos dados, iniciou-se a etapa de treinamento. Foram gerados seis modelos a partir do treinamento da base de dados com seis algoritmos de ML: *Decision Tree, Random Forest, AdaBoost, MLP, CatBoost e KNN*. Para cinco dos algoritmos, utilizou-se as implementações presentes na biblioteca *sklearn*: *DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, MLP e KNN*. Para o algoritmo *CatBoost*, foi utilizada a implementação *CatBoostClassifier* do pacote *Catboost*[5].

Após determinar a configuração ideal para cada algoritmo, o treinamento foi executado com o intuito de gerar os modelos finais. Para este processo, foi utilizado o método `fit` presente em cada um dos classificadores. Por fim, a integração e implementação dos módulos mencionados anteriormente resultaram na versão completa do MARIA, um mecanismo inteligente projetado para capturar, analisar, classificar e sugerir contramedidas a diferentes tipos de ataques em redes 5G em ambientes de IoT, conforme apresentado na

[3]https://imbalanced-learn.org/stable/references/generated/imblearn.under$_sampling.RandomUnderSampler.html$
[4]https://scikit-learn.org/1.5/modules/generated/sklearn.model$_selection.train_test_split.html$
[5]https://catboost.ai/en/docs/concepts/python-quickstart

Figura 3. Assim, o MARIA visa funcionar como uma ferramenta de resposta imediata a anomalias na rede, contribuindo para a mitigação dos danos que possam ser causados por ataques a dispositivos IoT conectados a essa infraestrutura.

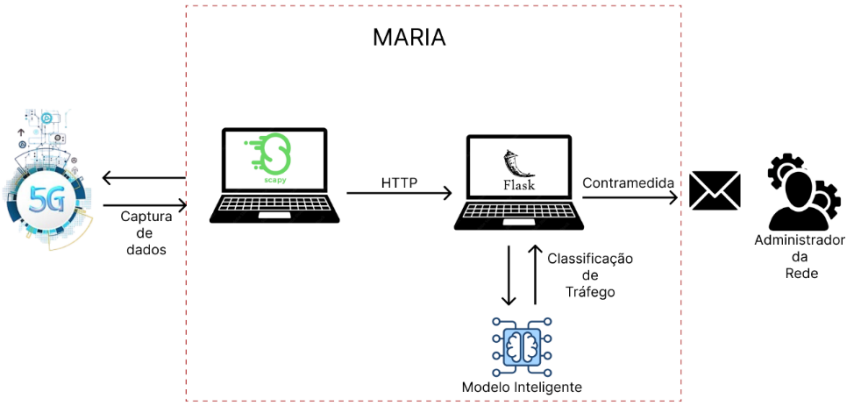


Figura 3. Fluxo Operacional do MARIA.

4.2.3. Execução dos algoritmos

Após o treinamento dos algoritmos e a construção dos modelos, foi realizada uma avaliação detalhada para identificar o modelo com melhor desempenho entre os seis analisados. Os dados foram divididos na proporção 80/20 para treinamento e teste dos modelos, respectivamente. Dessa forma, os dados de validação foram submetidos aos modelos, permitindo a classificação e a comparação com seus reais valores. Para este fim, utilizou-se o método `predict()`, presente em todos os modelos avaliados.

Com a classificação da base de teste realizada por todos os seis modelos, procedeu-se à comparação das predições com os valores reais da base de teste. Esse procedimento possibilitou a extração de diversas métricas, fundamentais para uma análise comparativa do desempenho entre os modelos. As métricas escolhidas para essa avaliação e comparação incluíram Acurácia, Recall, F1-Score e Precisão.

Com a extração das métricas foi observado que o *CatBoost* apresentou o melhor desempenho dentre as quatro métricas avaliadas. O fato de um modelo demonstrar valores superiores de forma unânime em todas as métricas sinaliza um bom desempenho em comparação com os demais. No entanto, decidiu-se realizar uma nova rodada de experimentos para analisar se houve, de fato, um aprendizado generalista por parte dos modelos ou se, em contrapartida, eles se adaptaram apenas ao conjunto de dados utilizado, resultando em um fenômeno conhecido como *overfitting* [Dietterich 1995].

Para conduzir o novo experimento, foi escolhida uma das técnicas de validação cruzada para avaliar o desempenho dos modelos. A validação cruzada é um método de reamostragem amplamente utilizado na seleção e avaliação de modelos de ML. Esse método serve para avaliar a capacidade de generalização dos modelos preditivos e para mitigar o problema de *overfitting* [Berrar 2019]. O método selecionado para a nova rodada de validação foi o `k-fold`, que consiste em dividir o conjunto de dados em `k` partes

distintas. O modelo é treinado `k` vezes, utilizando `k-1` partes para o treinamento e a parte restante para validar o desempenho. Após as `k` iterações, calcula-se a média das métricas para determinar qual algoritmo apresentou o melhor desempenho [Berrar 2019].

Para realizar a validação cruzada, foi utilizada a implementação do `cross_val_predict` [6] do pacote sklearn. Definiu-se `k=10` para o experimento, um valor amplamente utilizado em estudos acadêmicos e experimentações práticas [Kohavi 1995]. A Tabela 1 apresenta a média dos resultados das métricas para as dez iterações executadas durante a validação cruzada.

Tabela 1. Comparação dos Modelos Utilizando Validação Cruzada.

Modelo	Acurácia	Recall	F1-Score	Precisão
Decision Tree	90.54	90.54	90.08	93.23
Random Forest	90.65	90.65	90.18	93.45
MLP	79.36	79.36	78.44	89.60
AdaBoost	70.35	70.35	71.57	78.21
KNN	63.95	63.95	60.91	71.33
CatBoost	**90.72**	**90.72**	**90.24**	**93.61**

Ao analisar os resultados da validação cruzada, observou-se que o *CatBoost* apresentou, mais uma vez, o melhor desempenho entre todas as métricas avaliadas, reforçando a eficácia do modelo e sua capacidade de generalização. Um aspecto interessante foi a queda de desempenho do modelo *AdaBoost*, sugerindo um possível overfitting durante o primeiro experimento. Isso destaca a importância da validação cruzada para mitigar distorções nas análises de modelos de ML. Após a avaliação dos experimentos, constatou-se que o modelo mais adequado para resolver o problema proposto foi o *CatBoost*, que demonstrou desempenho superior tanto na metodologia 80/20 quanto na validação cruzada com `k-fold`. Assim, optou-se por utilizar o *CatBoost* como parte do mecanismo de detecção de ataques do sistema MARIA proposto no trabalho.

4.3. Módulo 4

No módulo de **Classificação do Tráfego**, a ferramenta *Joblib*[7] é empregada para serializar o modelo de ML selecionado, ou seja, aquele que apresentou o melhor desempenho. Esse modelo é então carregado na API, permitindo sua utilização como uma função dentro do sistema. Com os dados de entrada, o método `predict` é aplicado para realizar a classificação do tráfego, identificando possíveis anomalias ou padrões suspeitos. O resultado obtido é, em seguida, encaminhado ao Módulo de Transmissão, garantindo a integração fluida entre os componentes do sistema.

4.4. Módulo 5

O Módulo **Transmissão** é responsável por interpretar a classificação gerada e executar ações adequadas com base nos resultados. Este módulo opera utilizando o valor predito pelo modelo. Se o resultado da predição for `Benign`, o tráfego é considerado seguro e

[6] https://scikitlearn.org/1.5/modules/generated/sklearn.model_selection.cross_val_predict.html

[7] Joblib é uma biblioteca Python utilizada para a serialização eficiente de objetos grandes, como modelos de ML, oferecendo uma maneira rápida e robusta de salvá-los e carregá-los.

nenhuma ação adicional é tomada. No entanto, quando a *label* é diferente de `Benign`, o módulo identifica a presença de um ataque. As possíveis classificações de ataque incluem `DDoS`, `PartOfAHorizontalPortScan`, `Okiru`, `HTTPFlood` e `UDPFlood`. A tabela 2 descreve cada um dessas *label* bem como apresenta sugestões de contramedidas para cada uma delas.

Tabela 2. Sugestões de Contramedidas.

Label	Descrição	Sugestão de Contramedida
DDoS	Ataque de negação de serviço distribuído em que um dispositivo infectado lança uma atividade maliciosa para penetrar em outro dispositivo.	Balanceamento de carga; Controle de fluxo; *Drop request*; Análise de padrões [Ramanauskaite and Cenys 2011].
HTTP *Flood*	HTTP *flooding* pode ser iniciado enviando um grande número de solicitações GET ou POST para um servidor web alvo.	[Mohammadi et al. 2023] apresenta um controlador SDN que aplica algoritmos de ML. Caso um fluxo HTTP seja detectado como ataque, ele primeiro identifica e, em seguida, bloqueia a origem instalando regras no switch de borda conectado ao(s) invasor(es).
UDP *Flood*	Ataque no qual um grande número de pacotes UDP é enviado para um servidor.	Regras de firewall para restringir o acesso de IPs suspeitos; Implementação de um limitador de taxa para restringir a quantidade de solicitações de IP dentro de um período específico [Rao and Subbarao 2023].
PartOfAHorizontalPortScan	Indicação de conexão usada para fazer varredura de portas de forma horizontal para executar o ataque.	Ferramentas de detecção de portas; Firewall.
Okiru	Conexões com características de uma botnet Okiru.	Métodos de ML para identificação de anomalias [Khan and Sharma 2023].
Benign	Dispositivo sem atividades suspeitas ou maliciosas detectadas na rede.	-

4.5. Módulo 6

Para a implementação do módulo de **Notificação**, utilizou-se a biblioteca `smtplib`[8] do *Python*, que facilita o envio automático de *e-mails* assim que um ataque é identificado e classificado. No interior deste módulo, foi configurado um endereço de *e-mail* dedicado para atuar como remetente das mensagens de alerta. Quando um ataque é detectado pela `API`, uma notificação é enviada para o(s) destinatário(s), contendo informações sobre o tipo de ataque identificado e quaisquer recomendações relevantes para mitigar a ameaça. Este envio de notificação em tempo real é um elemento essencial para permitir respostas rápidas a incidentes de segurança, reduzindo o tempo de mitigação e minimizando os impactos de atividades maliciosas na infraestrutura da rede.

[8]https://docs.python.org/3/library/smtplib.html

4.6. *Testbed* do MARIA

Para avaliar o MARIA, foi configurado um cenário de teste destinado a replicar uma situação de ataque DDoS em um ambiente IoT em tempo real, conforme ilustrado na Figura 4.

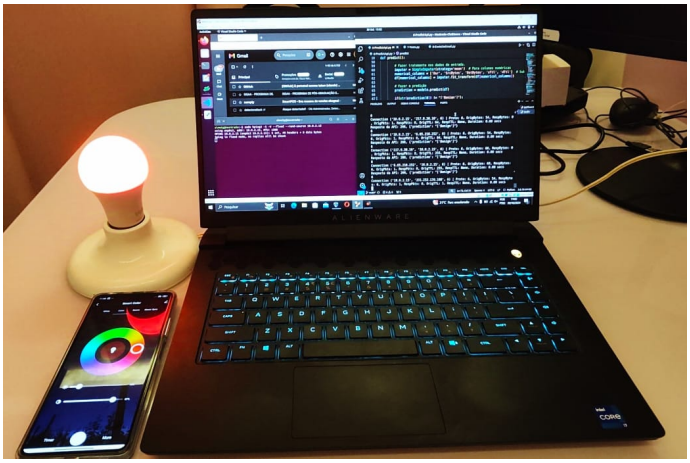


Figura 4. Cenário de teste.

No centro deste ambiente de teste, um *hotspot* com rede 5G conecta todos os elementos do cenário. Entre esses elementos, está um *smartphone* que se comunica com um dispositivo IoT, uma lâmpada inteligente. Além disso, uma máquina de monitoramento está ligada a `API` de detecção. Ao detectar um possível ataque, a `API` notifica o administrador da rede por *e-mail*, contendo informações sobre o tipo de ataque, bem como contramedidas em tempo real. Este cenário também inclui uma máquina atacante. A máquina de monitoramento capturou todos os dados que transitam pela rede através da função `sniff` do *Python*.

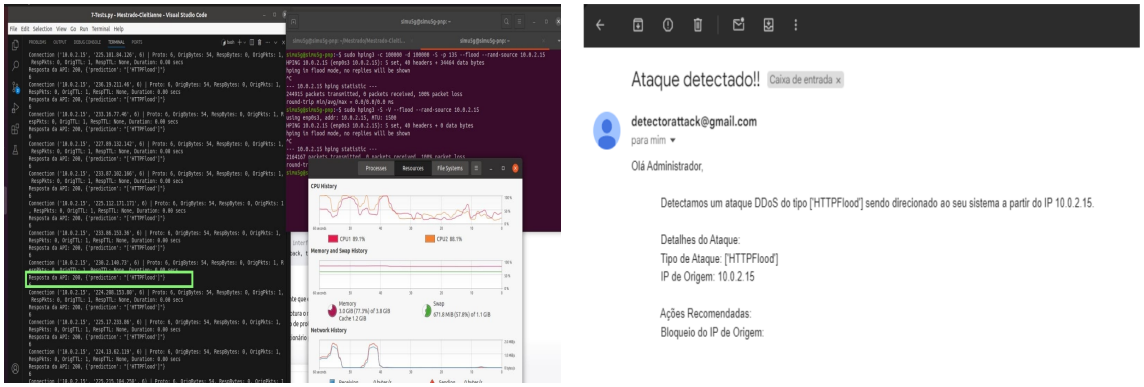
Durante o experimento, o dispositivo IoT gerou um fluxo constante de dados para a máquina de monitoramento. Simultaneamente, a máquina atacante transmitiu dados para o mesmo destino e implementou um ataque ao dispositivo IoT, simulando um ataque em um ambiente controlado. Na máquina atacante, foi utilizada a ferramenta `hping` para a execução dos ataques. O Hping3 é uma ferramentas de penetração TCP, amplamente utilizada para testes de rede e *host*. Além disso, é eficaz em avaliações de *firewall*, realização de traceroutes em diferentes protocolos, escaneamento de portas e geração de pacotes. A Figura 5 ilustra um ataque DoS em andamento, exibindo a estatística final gerada pelo Hping3.

A Figura 6a apresenta o testbed em execução, exibindo o ataque em andamento, os logs da API em tempo real e a porcentagem de uso da CPU. Quando o mesmo endereço IP realiza uma requisição, sendo registrado em um banco de dados local. Foi implementada uma regra que envia um e-mail ao administrador da rede. A Figura 6b demonstra o processo de envio de e-mail para o administrador.

Este experimento tem por objetivo evidenciar o impacto que um ataque DoS pode causar em uma rede 5G integrada a um cenário IoT. Além disso, busca avaliar o desempenho da `API` de detecção durante a ocorrência do ataque, testando sua eficácia e capacidade de resposta em condições reais.

```
simu5g@simu5g-pnp:~$ sudo hping3 -S -V --flood --rand-source 10.0.2.15
[sudo] password for simu5g:
using enp0s3, addr: 10.0.2.15, MTU: 1500
HPING 10.0.2.15 (enp0s3 10.0.2.15): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.2.15 hping statistic ---
1451426415 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figura 5. Estatística Final do Ataque DoS usando Hping3.



(a) Execução do testbed. **(b) Envio de Notificação em tempo real.**

Figura 6. Testbed do MARIA.

5. Conclusão e Trabalhos Futuros

Este artigo apresentou o MARIA, uma solução desenvolvida para o monitoramento e análise de vulnerabilidades em redes 5G, possibilitando a resposta em tempo real a essas ameaças. A integração da infraestrutura de redes 5G com a IoT promove uma interconexão entre dispositivos inteligentes, proporcionando maior cobertura, alta mobilidade e baixa latência. No entanto, o crescimento exponencial do volume de dados móveis e a proliferação de dispositivos IoT tornam esse ecossistema mais suscetível a ataques cibernéticos.

Diante desse cenário, a presente pesquisa resultou no desenvolvimento do MARIA, capaz de identificar e sugerir medidas de mitigação para seis tipos distintos de ataques em redes 5G no ambiente IoT. Além disso, a solução também realiza a identificação de tráfego benigno.

No desenvolvimento do modelo preditivo, aplicaram-se técnicas de ML em uma base de dados rotulada, contendo os tipos de ataques mencionados. Os experimentos resultaram em acurácia de 90,72%, precisão de 93,61%, recall de 90,72% e F1-score de 90,24% no modelo treinado utilizando o algoritmo CatBoost. Além disso, foi construído um cenário de Testbed para validar a solução em um ambiente real de um ataque a um dispositivo IoT em uma rede 5G. O teste demonstrou a capacidade do MARIA em identificar, notificar e sugerir contramedidas em tempo real frente aos ataques. A pesquisa destacou o uso de ML para identificar anomalias e ameaças, aprimorando a detecção e mitigação de ataques.

A avaliação do MARIA, foi conduzida por meio da divisão dos dados em uma proporção de 80/20, sendo 80% destinadas ao treinamento dos modelos e 20% à fase de testes. Além disso, empregou-se validação cruzada `k-fold`, segmentando o conjunto de dados em `k` partes distintas, realizando k experimentos com `k-1` partes para treinamento e `1` parte para teste do modelo. Esse procedimento permitiu a análise das métricas de desempenho, viabilizando a seleção do modelo mais eficaz e o cálculo da média dos resultados obtidos na validação cruzada.

Para trabalhos futuros, pretende-se realizar uma avaliação com outros modelos de ML, bem como a implementação de estratégias que aprimorem a eficiência e a precisão na detecção de ataques em redes 5G.

6. Disponibilidade de Artefatos

Os artefatos desenvolvidos nesta pesquisa estão disponíveis para garantir a replicabilidade do trabalho. O notebook do Google Colab apresenta o processo de construção da MARIA, abrangendo preparação do conjunto de dados, pré-processamento, treinamento do modelo e análise dos resultados. Dessa forma, ele permite que outros pesquisadores e profissionais reproduzam e validem os experimentos, contribuindo para a transparência e evolução da pesquisa. Link do Google Colab: `https://colab.research.google.com/drive/1ITzo10XapY42H_nhkU8dVDFQFcv9DAk8?usp=sharing`

Referências

Aljuhani, A. (2021). Machine learning approaches for combating distributed denial of service attacks in modern networking environments. *IEEE Access*, 9:42236–42264.

Berrar, D. (2019). Cross-validation. *Encyclopedia of Bioinformatics and Computational Biology*, 1(April):542–545.

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327.

Farzaneh, B., Shahriar, N., Al Muktadir, A. H., Towhid, M. S., and Khosravani, M. S. (2024). DTL-5G: Deep transfer learning-based DDoS attack detection in 5G and beyond networks. *Computer Communications*, 228:107927.

Garcia, S., Parmisano, A., and Erquiaga, M. J. (2021). Iot-23: a labeled dataset with malicious and benign iot network traffic (2020). *URL: https://www. stratosphereips. org/datasets-iot23. doi: http://doi. org/10.5281/zenodo*, 4743746.

Khan, A. and Sharma, I. (2023). Tackling okiru attacks in iot with ai-driven detection and mitigation strategies. In *International Conference on Power Energy, Environment & Intelligent Control (PEEIC)*, pages 336–341. IEEE.

Kim, Y.-E., Kim, Y.-S., and Kim, H. (2022). Effective feature selection methods to detect iot ddos attack in 5g core network. *Sensors*, 22(10):3819.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Morgan Kaufman Publishing*.

Kumar, S., Kumar, J. R. R., Mondal, D., Hemelatha, S., Diwakar, M. P., et al. (2023). A novel optimization strategies for detecting and preventing distributed denial of service attacks in wireless networks. In *International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, pages 1–6. IEEE.

Kumari, P. and Jain, A. K. (2023). A comprehensive study of ddos attacks over iot network and their countermeasures. *Computers & Security*, 127:103096.

Mohammadi, R., Lal, C., and Conti, M. (2023). Httpscout: a machine learning based countermeasure for http flood attacks in sdn. *International Journal of Information Security*, 22(2):367–379.

Nascita, A., Cerasuolo, F., Di Monda, D., Garcia, J. T. A., Montieri, A., and Pescape, A. (2022). Machine and deep learning approaches for iot attack classification. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE.

Nkosi, Z. G. and Mathonsi, T. E. (2024). A hybrid security algorithm for 5g-internet of things. In *International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pages 1–6. IEEE.

Örs, F. K., Aydın, M., Boğatarkan, A., and Levi, A. (2021). Scalable wi-fi intrusion detection for iot systems. In *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–6. IEEE.

Pakmehr, A., Aßmuth, A., Taheri, N., and Ghaffari, A. (2024). Ddos attack detection techniques in iot networks: a survey. *Cluster Computing*, 27(10):14637–14668.

Ramanauskaite, S. and Cenys, A. (2011). Taxonomy of dos attacks and their countermeasures. *Central European Journal of Computer Science*, 1:355–366.

Rao, G. S. and Subbarao, P. K. (2023). A novel framework for detection of dos/ddos attack using deep learning techniques, and an approach to mitigate the impact of dos/ddos attack in network environment. *International Journal of Intelligent Systems and Applications in Engineering*, 12(1):450–466.

Saheed, Y. K., Abiodun, A. I., Misra, S., Holone, M. K., and Colomo-Palacios, R. (2022). A machine learning-based intrusion detection for detecting internet of things network attacks. *Alexandria Engineering Journal*, 61(12):9395–9409.

Samarakoon, S., Siriwardhana, Y., Porambage, P., Liyanage, M., Chang, S.-Y., Kim, J., Kim, J., and Ylianttila, M. (2022). 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network.

Yang, M. and Zhang, J. (2023). Data anomaly detection in the internet of things: A review of current trends and research challenges. *International Journal of Advanced Computer Science and Applications*, 14(9).