IoTSHIELD: Sistema de Detecção de Intrusão contra *Botnets* para *Roteadores* e *Dispositivos de Borda*

Magno Ramos Guimarães[1], **Dalbert Matos Mascarenhas**[1]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ)
Petrópolis, RJ, Brasil

magno.guimaraes@aluno.cefet-rj.br, dalbert.mascarenhas@cefet-rj.br

Abstract. *The proliferation of vulnerable IoT devices has driven the creation of destructive botnets. This paper proposes an efficient Intrusion Detection System (IDS) against botnets designed for network edge devices and routers. The solution monitors local traffic and employs a Random Forest machine learning algorithm to identify anomalous patterns and isolate Mirai-infected devices. Validated on a dataset of over 318,000 network flows, the model achieved 99.94% accuracy and F1-Score. In end-to-end tests simulating restricted router hardware (1.15 GHz single-core CPU and 128 MB RAM), the IDS maintained a throughput exceeding 578,000 packets per second while consuming only 36.8 MB of RAM, proving its resource-efficiency.*

Resumo. *A proliferação de dispositivos IoT vulneráveis tem impulsionado botnets destrutivas. Este artigo propõe um Sistema de Detecção de Intrusão (IDS) eficiente contra botnets projetado para roteadores e dispositivos de borda. A solução monitora o tráfego local e emprega o algoritmo Random Forest para identificar padrões anômalos e isolar equipamentos infectados pelo malware Mirai. A metodologia foi validada com mais de 318 mil fluxos de rede, alcançando 99,94% de acurácia e F1-Score. Em testes simulando hardware restrito (monocore de 1,15 GHz e 128 MB de RAM), o IDS obteve throughput superior a 578 mil pacotes por segundo, consumindo apenas 36,8 MB de RAM, indicando viabilidade e alto desempenho na borda.*

1. Introdução

A proliferação de dispositivos da Internet das Coisas (IoT) trouxe conveniência aos usuários domésticos, mas introduziu desafios críticos de segurança. Devido a senhas inalteradas, firmwares desatualizados e ausência de *hardening*, esses equipamentos tornaram-se alvos de botnets [Antonakakis et al. 2017][Silva et al. 2013]. Sequestrados, são usados para orquestrar ataques massivos. A magnitude dessa ameaça reflete-se na botnet Aisuru que, no terceiro trimestre de 2025, infectou até 4 milhões de dispositivos globais. Essa rede lançou em média 14 ataques DDoS diários, atingindo picos sem precedentes de 29,7 Tbps e 14,1 Bpps [Cloudflare 2025].

Enquanto corporações mitigam tais ataques com Sistemas de Detecção de Intrusão (IDS) robustos, como Suricata ou Snort, as redes residenciais seguem desprotegidas. A limitação de recursos dos roteadores comuns inviabiliza a transposição dessas ferramentas. Simultaneamente, soluções comerciais (como Fing e Bitdefender BOX) restringem-se ao monitoramento passivo, falhando na contenção ativa. Embora a Inteligência Artificial

aplicada à análise comportamental seja promissora, as soluções acadêmicas de Aprendizado de Máquina esbarram na alta complexidade dos modelos e na carência de testes práticos em ambientes reais [Rahman et al. 2025].

Para superar essa lacuna, propõe-se o IoTSHIELD, um *framework* de detecção projetado para atuar na borda da rede. O objetivo é identificar atividades de botnets diretamente no roteador doméstico em seus estágios iniciais, isolando o dispositivo comprometido antes que a ameaça escale. Para garantir sua eficácia, a validação do modelo Random Forest superou o uso exclusivo de bases estáticas: o IoTSHIELD foi submetido a uma Prova de Conceito em ambiente conteinerizado (Docker), utilizando o código-fonte adaptado do malware Mirai[Lee][Gamblin].

As principais contribuições deste artigo são:

- O desenvolvimento de um IDS de baixo custo computacional e alto *throughput*, capaz de processar mais de 578 mil pacotes por segundo utilizando hardware restrito (arquitetura monocore emulada a 1,15 GHz)[Guimarães 2026];
- A validação prática em ambiente de laboratório isolado (Docker) com tráfego malicioso real gerado pelo malware Mirai;
- A criação e disponibilização de um ecossistema de código aberto para a geração de modelos IDS otimizados baseados em Random Forest[Guimarães 2026].

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 detalha a metodologia de construção do modelo e as *features* escolhidas. A Seção 4 detalha a topologia do ambiente Docker usado em testes. A Seção 5 apresenta e discute os resultados do experimento. Por fim, a Seção 6 conclui o trabalho e delineia direcionamentos para pesquisas futuras.

2. Trabalhos Relacionados

A revisão de Elrawy et al. (2018) mapeia abordagens de IDS em ambientes inteligentes, apontando que sistemas baseados em assinaturas exigem bancos de dados inviáveis para roteadores [Elrawy et al. 2018]. O IoTSHIELD supera isso atuando na borda como um IDS de anomalias: elimina assinaturas pesadas e utiliza atributos agnósticos à carga útil, preservando a privacidade do usuário e adequando-se às limitações de hardware.[Rahman et al. 2025]

Sobre o treinamento de modelos, Kumar et al. (2019) destacam a obsolescência de bases clássicas como a KDD99 [Kumar et al. 2019]. O IoTSHIELD compartilha dessa necessidade de curadoria moderna, mas focada em roteadores, combinando tráfego benigno puro (CICIDS-2017) e capturas reais do malware Mirai (IoT-23) [Stratosphere Laboratory][Canadian Institute for Cybersecurity 2017]. Além disso, enquanto a literatura foca em testes estáticos, o IoTSHIELD utilizou contêineres Docker para simulação dinâmica e validação de ataques em tempo real.

Buscando maior discriminação, Awajan (2023) propôs um IDS baseado em Deep Learning de quatro camadas [Awajan 2023]. Apesar da boa acurácia, a sobrecarga matemática dessas redes é inviável em roteadores comerciais. O IoTSHIELD segue uma filosofia oposta para garantir viabilidade na borda: emprega um modelo Random Forest otimizado e transpilado para código C nativo, eliminando a alta densidade matemática com desempenho superior.

Por fim, Bhavsar et al. (2023) usaram o Coeficiente de Pearson para selecionar atributos antes de aplicá-los a uma Rede Neural Convolucional (CNN) [Bhavsar and Jhaveri 2023]. Embora reduza a redundância, a arquitetura mantém o gargalo computacional e a avaliação estática. O IoTSHIELD, em contraste, extrai apenas 17 *features* temporais e volumétricas, mitigando a complexidade sem o custo de uma CNN. É uma solução ágil de alto rendimento, atuando como gatilho responsivo integrado às regras nativas do roteador (como iptables ou VLAN).

3. Metodologia

Para viabilizar a detecção de anomalias em roteadores residenciais com recursos computacionais severamente limitados, a arquitetura do IoTSHIELD foi projetada com uma separação estrita entre a fase de treinamento e a fase de inferência. A metodologia baseia-se em um *pipeline* de quatro estágios: (i) Extração de dados, (ii) Balanceamento, (iii) Treinamento do modelo matemático e (iv) Transpilação para código nativo C. O treinamento ocorre em um ambiente robusto controlado, enquanto apenas o produto final otimizado é embarcado no dispositivo de borda.

3.1. Extração de Tráfego e Engenharia de Atributos

A primeira etapa do *pipeline* consiste em converter pacotes de rede brutos (arquivos .pcap) em fluxos de comunicação bidirecionais (sessões). Para garantir a integridade da memória durante o processamento de grandes volumes de tráfego, adotou-se a técnica de *Pcap Splitting* combinada com multiprocessamento na linguagem Python, utilizando a biblioteca Scapy para a dissecção dos protocolos e o NumPy para a estruturação massiva das matrizes de dados.

Cada fluxo isolado é analisado para a geração de uma tabela (DataFrame manipulado com Pandas), onde cada linha representa uma sessão de comunicação e as colunas contêm atributos puramente comportamentais. Para garantir que o modelo aprenda a assinatura holística do ataque e não memorize o ambiente de laboratório, aplicaram-se duas técnicas fundamentais:

- **Cegueira Intencional (*Data Leakage Removal*):** Atributos identificadores, como TTL (Time-to-Live), endereços IP e portas de origem e destino, foram sumariamente excluídos. Isso força o modelo a avaliar o "como" a comunicação ocorre, tornando o IDS agnóstico à topologia da rede[Bouke et al. 2024].
- **Normalização de Contexto e *Clipping*:** Para evitar que a Inteligência Artificial decore a intensidade absoluta do tráfego do laboratório, aplicou-se a remoção de outliers (limites de teto para métricas agressivas) e a conversão de valores brutos em taxas matemáticas (como a porcentagem de pacotes com a *flag* SYN), garantindo a eficácia do modelo.

3.2. Engenharia e Seleção de Atributos Comportamentais

O IoTSHIELD exclui a Inspeção Profunda de Pacotes (DPI) para contornar gargalos em CPUs monocore e preservar a privacidade do usuário. Em substituição à análise de *payload*, o modelo atua sobre um vetor otimizado de 17 atributos (*features*) extraídos de cada fluxo, concebidos para refletir a semântica da comunicação em vez de seu conteúdo. Para o treinamento e inferência, as 17 características foram consolidadas em seis dimensões comportamentais[Kononenko and Hong 1997, Livadas et al. 2006, Yuan et al. 2023, Awajan 2023]:

- **Métricas de Vida Útil (*Duration*):** Composto por `bidirectional_duration_ms`, `src2dst_duration_ms` e `dst2src_duration_ms`. Essas variáveis avaliam o tempo total de sobrevivência da conexão. Sessões humanas legítimas tendem a ser persistentes, enquanto malwares em varredura abrem e fecham portas em milissegundos.
- **Cadência de Disparo (*Packet Inter-Arrival Time* - PIAT):** Engloba o tempo de chegada entre os pacotes da sessão (`bidirectional_max_piat_ms`, `bidirectional_mean_piat_ms`, `src2dst_max_piat_ms`, `dst2src_max_piat_ms`, `dst2src_mean_piat_ms`). O tempo máximo e a média de "recarga" entre os pacotes são fundamentais para distinguir o determinismo das máquinas da aleatoriedade humana. Botnets apresentam ritmos de disparo mecanicamente constantes, contrastando com os comportamentos gerados por humanos.
- **Agressividade e Paralelismo (*Concurrent Flows*):** Formado por `src2dst_concurrent_flows`, `dst2src_concurrent_flows` e `src2dst_flow_rate`. Avalia a quantidade de conexões sobrepostas e a taxa de abertura de novos fluxos por segundo, identificando a agressividade do host. Para garantir que o modelo não memorizasse a intensidade artificial do laboratório, essas métricas sofreram um teto matemático (*clipping* limitador), forçando a Inteligência Artificial a focar na proporção do paralelismo e não no volume absoluto.
- **Assinaturas de Abertura TCP:** Contagem bruta de pacotes com a *flag* de sincronização (`src2dst_syn_packets` e `bidirectional_syn_packets`). A contagem de solicitações de abertura é essencial para a identificação primária de ataques de exaustão de estado de conexão.
- **Engenharia Comportamental e Proporções (*Engineered Features*):** Em vez de utilizar apenas valores brutos fornecidos pela rede, o IoTSHIELD introduz atributos derivados de cálculo matemático em tempo real, tornando a identificação mais precisa para certos ataques:
  - `in_out_packet_ratio`: Mede a simetria dividindo pacotes enviados por recebidos. Tráfegos maliciosos (como inundações DDoS) apresentam forte assimetria: milhares de pacotes vão, mas nenhum volta, diferentemente de fluxos TCP benignos, que são equilibrados.
  - `syn_to_total_ratio`: Taxa percentual de pacotes SYN em relação ao total do fluxo. Se um fluxo possui uma razão próxima a 1,0 (100% dos pacotes são requisições SYN), evidencia-se a intenção de inabilitação (SYN Flood) em vez de comunicação legítima.
  - `is_constant_payload`: Sinalizador binário baseado no desvio padrão do tamanho dos pacotes (em bytes). Sinaliza como verdadeiro (1,0) se a variação for próxima a zero, identificando malwares como o Mirai, que disparam grandes rajadas de pacotes de tamanho idêntico para maximizar o dano volumétrico.
- **Contexto de Protocolo:** A característica derivada `is_discovery_protocol`. Um desafio particular em redes locais residenciais é o alto volume de tráfego de descoberta (como SSDP, mDNS e DHCP), que é barulhento e compartilha semelhanças volumétricas com varreduras maliciosas. Esta *flag* atua como uma máscara de contexto para identificar tais protocolos. Embora tenha figurado

com a menor importância decisória global no modelo (cerca de 1,4% de Redução Média de Impureza), sua introdução foi arquiteturalmente vital: ela reduziu os Falsos Positivos (tráfego benigno bloqueado) de quase 900 fluxos perdidos para apenas 47, preservando a experiência do usuário sem degradar a detecção das ameaças reais.

A Tabela 1 detalha a importância de cada atributo na decisão final do modelo, evidenciando a dominância das métricas de paralelismo e proporção.

Tabela 1. Importância das *features* na tomada de decisão do modelo (Redução Média de Impureza).

Rank	Nome do Atributo (*Feature*)	Importância (%)
1	`src2dst_concurrent_flows`	12,02
2	`src2dst_flow_rate`	10,50
3	`in_out_packet_ratio`	10,44
4	`syn_to_total_ratio`	10,35
5	`bidirectional_max_piat_ms`	6,12
6	`bidirectional_mean_piat_ms`	5,86
7	`dst2src_duration_ms`	5,41
8	`bidirectional_duration_ms`	5,27
9	`dst2src_concurrent_flows`	4,81
10	`src2dst_duration_ms`	4,72
11	`src2dst_max_piat_ms`	4,62
12	`src2dst_syn_packets`	4,20
13	`is_constant_payload`	4,13
14	`dst2src_mean_piat_ms`	3,67
15	`dst2src_max_piat_ms`	3,25
16	`bidirectional_syn_packets`	3,15
17	`is_discovery_protocol`	1,40

3.3. Balanceamento e Otimização do Modelo

O classificador foi construído utilizando o algoritmo Random Forest, implementado através da biblioteca Scikit-Learn. A escolha por agrupamentos de árvores de decisão em detrimento de Redes Neurais Profundas (*Deep Learning*) foi estratégica: a ausência de multiplicações complexas de matrizes facilita a execução em hardware restrito, e a votação da floresta previne a dependência de atributos ruidosos[Elrawy et al. 2018, Batista et al. 2004, Barkah et al. 2023].

Para garantir um treinamento imparcial, a base de dados foi nivelada utilizando a técnica de *Undersampling* não apenas entre as classes Benigna e Maliciosa, mas também pela inclusão balanceada de diferentes tipos de tráfego benigno como telemetria, navegação web, *streaming* e comportamentos padrão de roteadores (e.g., protocolos de descoberta), evitando que o modelo crie vieses algorítmicos[Barkah et al. 2023]. O algoritmo utilizou a limitação de profundidade de poda (*Pruning* máximo de 8 passos) para impedir o overfitting. A robustez da precisão de 99,94% foi homologada utilizando Validação Cruzada K-Fold com 5 partições, combinada com o método de Amostragem em

Reservatório (*Reservoir Sampling*) para capturar comportamentos homogêneos de todas as fases do ataque (do reconhecimento à inundação).

3.4. Transpilação e Implantação na Borda

O modelo Random Forest, originalmente treinado no ambiente Python, é submetido a um processo de transpilação automatizada utilizando o gerador `m2cgen`.

Todo o conhecimento matemático do modelo é traduzido em estruturas condicionais nativas da linguagem C, dispensando a necessidade de bibliotecas externas pesadas e garantindo uma integração fluida ao firmware do roteador. No dispositivo final, as capturas de tráfego ocorrem em tempo real através da biblioteca padrão `libpcap`, injetando os dados de fluxo diretamente no motor nativo. Esta abordagem permitiu reduzir a latência de inferência de forma drástica, saltando de um limite restritivo em Python para operações em níveis sub-microssegundos, indicando a viabilidade técnica da aplicação do IDS em sistemas IoT-Edge.

4. Topologia e Avaliação em Ambiente Emulado

Para validar o funcionamento dinâmico do IoTSHIELD de forma segura e reprodutível, construiu-se uma Prova de Conceito (PoC) utilizando um ambiente virtualizado baseado em contêineres Docker[Elrawy et al. 2018]. Esta abordagem de *sandboxing* lógico é estritamente necessária, uma vez que o experimento lidou com artefatos reais do código-fonte do malware Mirai, exigindo isolamento total para evitar o vazamento da infecção para a rede hospedeira ou para a internet pública[Liu et al. 2025].

4.1. Arquitetura da Rede e Contêineres

O ambiente foi orquestrado sobre uma rede virtual interna do Docker batizada de `botnet`. Para simular uma rede doméstica (LAN) conectada à rede externa (WAN), a topologia utiliza o roteador como núcleo central. O roteador assumiu a responsabilidade pelos serviços de gateway e DHCP, atribuindo dinamicamente os endereços IP para todos os dispositivos da rede.

Ao todo, a topologia foi composta por 5 contêineres atuando de forma simultânea:

1. **Roteador de Borda (IoTSHIELD):** O núcleo defensivo da rede. Para garantir a validade dos testes de desempenho, este contêiner teve sua capacidade de hardware intencionalmente estrangulada, emulando as severas restrições de processamento e memória de um roteador residencial genérico.
2. **Servidor Comando e Controle (C&C):** O nó malicioso responsável por orquestrar a botnet, emitindo as ordens de ataque para as máquinas infectadas.
3. **Nós Infectados (Bots 1 e 2):** Dois contêineres emulando dispositivos IoT vulneráveis dentro da rede local, previamente infectados com o binário do malware Mirai e aguardando comandos do C&C.
4. **Servidor Alvo (Vítima):** Um contêiner executando um servidor web atuando como o alvo final do ataque de Negação de Serviço.

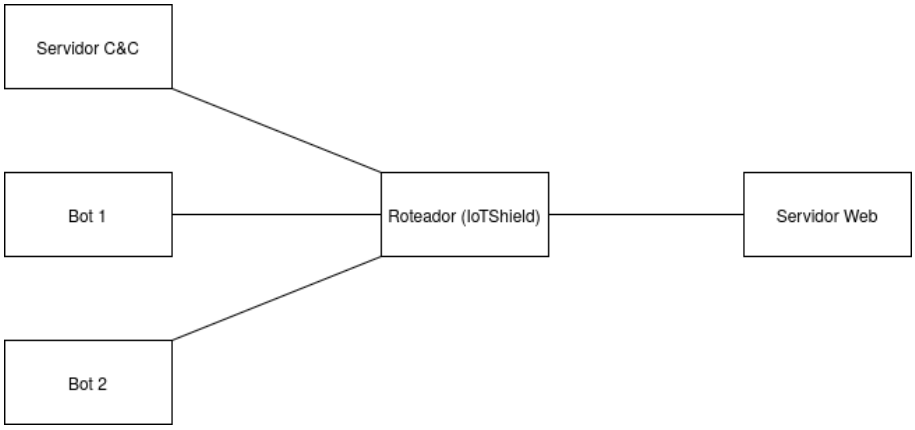

Figura 1. Topologia do ambiente isolado em Docker, evidenciando o roteador central mediando a comunicação entre a botnet e o servidor alvo.

O fluxograma ilustrado na Figura 1 detalha o fluxo vetorial da rede. Todo o tráfego gerado pelos bots em direção ao servidor web obrigatoriamente trafega através do Roteador de Borda. É neste ponto de estrangulamento que o IoTSHIELD opera, capturando as métricas comportamentais dos fluxos em tempo real e as injetando no modelo Random Forest transpilado em C.

4.2. Dinâmica do Ataque e Mitigação em Tempo Real

O experimento simulou o comportamento de uma botnet em que o servidor C&C ordenou a dois bots um ataque SYN Flood contínuo de 60 segundos contra um servidor web [Silva et al. 2013][Antonakakis et al. 2017]. Visando equilibrar agilidade e prevenção de Falsos Positivos, o limiar de sensibilidade (*Threshold*) do IoTSHIELD foi fixado em 0,6.

Resultados da Interceptação: Logo nos primeiros instantes do ataque, o módulo extrator detectou anomalias drásticas na proporção TCP e na taxa de conexões concorrentes. A avaliação da Inteligência Artificial sobre os fluxos de saída superou rapidamente o limiar de 0,6, acionando de imediato as regras de bloqueio no firewall do roteador. Como resultado, a comunicação dos bots infectados foi cortada, mitigando o ataque nos primeiros momentos.

Essa mitigação cirúrgica evidencia que, em um cenário real de sequestro pelo malware Mirai, o IoTSHIELD detecta e contém a anomalia na origem. A intervenção precoce minimiza a degradação do serviço na infraestrutura alvo, transformando o roteador doméstico de um mero vetor de ataque em uma barreira defensiva ativa.

5. Resultados e Discussão

A avaliação do IoTSHIELD foi conduzida com um duplo objetivo: aferir a eficácia do algoritmo na identificação de comportamentos maliciosos operando sob severas restrições de hardware. Os resultados apresentados nesta seção demonstram a viabilidade de transferir a inteligência de detecção de infraestruturas centralizadas para a borda da rede.

5.1. Desempenho de Classificação e Sensibilidade

O modelo Random Forest transpilado foi submetido à validação com uma base de dados contendo mais de 318 mil fluxos bidirecionais isolados. O sistema alcançou médias de Acurácia global e F1-Score de 99,94% (calculados sobre o consolidado das 5 partições da validação cruzada), indicando sua capacidade de separação entre o tráfego limpo e as assinaturas de ataque.

Tabela 2. Matriz de confusão e métricas de desempenho evidenciando a baixa incidência de Falsos Positivos frente a mais de 159 mil fluxos benignos.

Métrica / Categoria	**Valor Observado**
Acurácia Global	99,94%
Precisão	99,94%
F1-Score	99,94%
Verdadeiros Negativos (TN)	158.972
Falsos Positivos (FP)	133
Falsos Negativos (FN)	63
Verdadeiros Positivos (TP)	159.041

A Tabela 2 ilustra que a alta precisão não prejudicou a usabilidade da rede: a taxa de Falsos Positivos permaneceu restrita a apenas 133 fluxos em um universo de 159.105 conexões benignas (0,08%). Essa segurança operacional é resultado direto do balanceamento multidimensional da fase de treinamento.

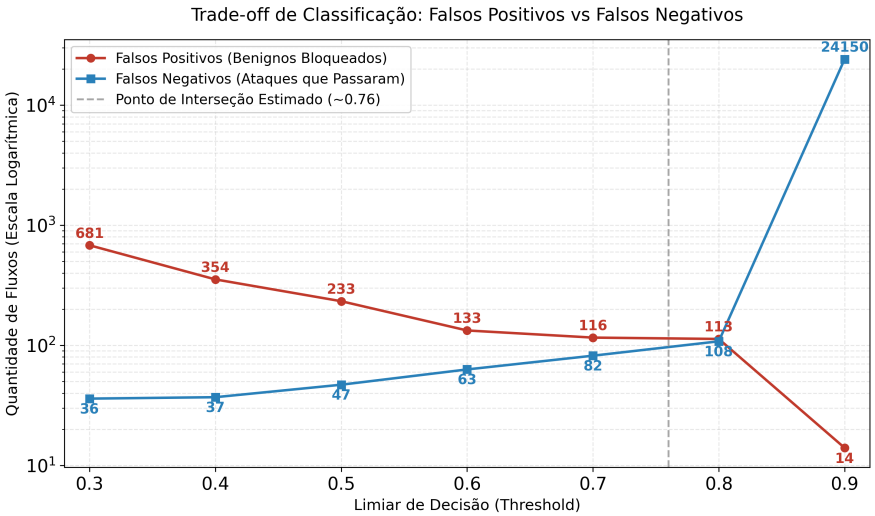

Figura 2. Impacto do limiar de confiança (*Threshold*) na taxa de bloqueio e passagem de tráfego.

Além disso, a sensibilidade do IDS é parametrizável. Conforme demonstrado na Figura 2, o sistema permite ajustar o limiar de decisão (*Threshold*). Na Prova de Conceito (PoC) dinâmica, a configuração do limiar em 0,6 revelou-se ideal: diante de uma inundação SYN Flood orquestrada pela botnet Mirai, o modelo identificou a anomalia e acionou o bloqueio logo após o início do ataque, interrompendo a ameaça.

5.2. Desempenho Computacional e Consumo de Recursos

Para atestar a adequação do IoTSHIELD a equipamentos domésticos, realizou-se um benchmark End-to-End (integrando a captura de pacotes nativa em C com o motor de inferência) simulando as especificações de um roteador commodity. O ambiente foi limitado, via contenção de sistema, a 128 MB de memória RAM total e a um processador monocore emulado operando a 1,15 GHz.

Tabela 3. Consumo de recursos computacionais do módulo de interceptação e inteligência artificial no ambiente restrito.

Ambiente	CPU Utilizada	Memória RAM	Latência/Pacote
Python (Scapy + ML)	109%	94,3 MB	10,64 μs
Firmware Nativo C	99%	36,6 MB	0,40 μs
C Emulado (1.15GHz/128MB)	25%	36,8 MB	1,73 μs

De igual importância, a Tabela 3 destaca o consumo de memória RAM. O IoTSHIELD operou de forma estável utilizando apenas 36,8 MB, um valor amplamente compatível com roteadores legados, que comumente possuem entre 64 MB e 128 MB de memória instalada, deixando os recursos restantes livres para as funções naturais de roteamento (NAT, DHCP, Wi-Fi). Os 109% de CPU do Python refletem sua paralelização multicore, contrastando com a eficiência do código C, restrito a 25% de um único núcleo.

5.3. Discussão: A Descentralização da Defesa na Borda

Historicamente, o mercado de cibersegurança tem focado em proteger as vítimas dos ataques DDoS (bancos, provedores e datacenters) utilizando Sistemas de Detecção de Intrusão extremamente pesados, de alto custo e centralizados. No entanto, esses ataques massivos são gerados por exércitos de dispositivos IoT residenciais infectados, operando em redes sem qualquer tipo de defesa nativa.

Os resultados obtidos pelo IoTSHIELD demonstram a viabilidade técnica de inverter esse paradigma: é possível embarcar Inteligência Artificial de alta precisão diretamente nos roteadores domésticos. Bloquear o ataque na fonte traz vantagens arquiteturais profundas em comparação aos modelos tradicionais centralizados:

- **Eliminação do Ponto Único de Falha (SPOF):** A descentralização da defesa em redes IoT é defendida como uma estratégia para evitar pontos únicos de falha e a dependência de gateways centrais, que podem se tornar gargalos ou alvos de ataque[Ioannou et al. 2021]. O IoTSHIELD distribui o esforço de detecção. O fardo computacional não recai sobre um único servidor na nuvem, mas é dividido horizontalmente entre milhões de roteadores independentes na borda.
- **Mitigação Precoce:** Estudos indicam que a implementação de agentes de segurança dedicados em posições estratégicas permite a detecção e o isolamento de comportamentos maliciosos na camada de roteamento, antes que a propagação do ataque comprometa a integridade de toda a infraestrutura[Ioannou et al. 2021].
- **Imunidade Sistêmica:** A segregação de funções de segurança, onde a lógica de detecção é independente da aplicação principal, confere robustez ao sistema. Conforme proposto por Ioannou et al. (2021), essa abordagem garante que o monitoramento permaneça operacional mesmo que os nós da aplicação sejam comprometidos, promovendo uma defesa resiliente e distribuída[Ioannou et al. 2021].

Portanto, a arquitetura viabiliza um IDS ágil e preciso sem *upgrades* físicos nos roteadores domésticos, fomentando uma defesa distribuída e atuando precocemente sobre as fontes das botnets.

5.4. Limitações e Escolhas de Projeto

A transposição do modelo para a rede real revelou desafios de generalização da Inteligência Artificial. Nos testes em Docker, o bloqueio do ataque ocorreu com uma certeza algorítmica inferior aos 99,94% obtidos na validação estática. Embora a confiança tenha sido suficiente para superar o limiar seguro de 0,6 e mitigar a anomalia, essa discrepância ocorreu porque a variante do Mirai compilada para a Prova de Conceito apresentava assinaturas ligeiramente distintas da base de treinamento [Stratosphere Laboratory][Lee]. Isso indica que a eficácia preditiva contra variantes inéditas (*zero-day*) pode ser aprimorada diversificando o conjunto de dados com novos comportamentos e sub-balanceamentos por tipos de ataque.

Apesar disso, a estratégia de balanceamento adotada provou-se acertada. O nivelamento considerou não apenas as classes (benigno e malicioso), mas a representatividade dos diferentes tipos de tráfego. Essa decisão evitou vieses (*bias*) em relação a comportamentos majoritários, forçando o IDS a aprender anomalias de forma agnóstica [Barkah et al. 2023].

Quanto à seleção algorítmica, o projeto focou exclusivamente no Random Forest. Embora não tenha havido comparação prática com modelos como Support Vector Machines ou Gradient Boosting neste escopo, a escolha fundamentou-se na literatura recente. O Random Forest oferece o melhor compromisso entre alta acurácia para dados tabulares de rede e uma estrutura de nós simples, característica vital para viabilizar sua transpilação para código C nativo sem exigir bibliotecas matemáticas pesadas [Rahman et al. 2025].

6. Conclusão e Trabalhos Futuros

A proliferação de dispositivos vulneráveis exige transferir a contenção para a borda. Este trabalho apresentou uma prova de conceito viabilizando IA em roteadores de recursos restritos. Ao extrair apenas características espaço-temporais dos fluxos, o modelo obteve 99,94% de acurácia estática. A transpilação do Random Forest para C nativo dispensou bibliotecas de alto nível, permitindo analisar mais de 578 mil pacotes por segundo consumindo apenas 36,8 MB de RAM. Em testes via Docker, a detecção rápida conteve as inundações da botnet Mirai antes que degradassem o alvo, indicando que o bloqueio ativo na fonte é crucial para quebrar a assimetria dos ataques DDoS.

Como desdobramentos para pesquisas futuras, focados na robustez e escalabilidade do IDS, destacam-se duas frentes principais:

- **Expansão do Espectro de Ameaças:** Estender o treinamento para outras famílias de malwares (como Mozi e Gafgyt), visando a construção de um modelo unificado de contenção.
- **Diversificação de Tipos de Tráfego:** Incluir e balancear um espectro mais amplo de fluxos legítimos, garantindo que o IDS mantenha sua precisão preditiva diante da constante evolução e heterogeneidade das redes domésticas.

Agradecimentos

Este trabalho foi realizado com recursos do CEFET/RJ.

Referências

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., and Zhou, Y. (2017). Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, Vancouver, BC. USENIX Association.

Awajan, A. (2023). A novel deep learning-based intrusion detection system for iot networks. *Computers*, 12(2).

Barkah, A. S., Selamat, S. R., Abidin, Z. Z., and Wahyudi, R. (2023). Impact of data balancing and feature selection on machine learning-based network intrusion detection. *International Journal on Informatics Visualization*.

Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29.

Bhavsar, Y. and Jhaveri, R. H. (2023). Anomaly-based intrusion detection system for iot application. *Discover Internet of Things*.

Bouke, M. A., Zaid, S. A., Abdullah, A., et al. (2024). Implications of data leakage in machine learning preprocessing: A multi-domain investigation. Preprint (Version 1) available at Research Square.

Canadian Institute for Cybersecurity (2017). Intrusion detection evaluation dataset (cicids2017).

Cloudflare (2025). Ddos threat report 2025 q3.

Elrawy, M. F., Awad, A. I., and Hamed, H. F. A. (2018). Intrusion detection systems for iot-based smart environments: a survey. *Journal of Cloud Computing*.

Gamblin, J. Mirai-source-code: Repositório de código fonte do mirai.

Guimarães, M. R. (2026). Iotshield: Sistema de detecção de intrusão contra botnets.

Ioannou, C., Charalambus, A., and Vassiliou, V. (2021). Decentralized dedicated intrusion detection security agents for iot networks. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 414–419.

Kononenko, I. and Hong, S. J. (1997). Attribute selection for modelling. *Future Generation Computer Systems*, 13(2):181–195. Data Mining.

Kumar, V., Sinha, D., Das, A. K., Pandey, S. C., and Goswami, R. T. (2019). Uids: a unified intrusion detection system for iot environment. *Evolutionary Intelligence*.

Lee, J. Implementação mirai.

Liu, M., Yi, B., Huang, L., Yan, X., Wang, J., Yan, R., and Wang, X. (2025). Enhancing network simulation with container: A scalable and real-time supported platform. In *2025 IEEE 25th International Conference on Communication Technology (ICCT)*, pages 379–383.

Livadas, C., Walsh, R., Lapsley, D., and Strayer, W. T. (2006). Usilng machine learning technliques to identify botnet traffic. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 967–974.

Rahman, M. M., Shakil, S. A., and Mustakim, M. R. (2025). A survey on intrusion detection system in iot networks. *Cyber Security and Applications*, 3:100082.

Silva, S. S., Silva, R. M., Pinto, R. C., and Salles, R. M. (2013). Botnets: A survey. *Computer Networks*, 57(2):378–403. Botnet Activity: Analysis, Detection and Shutdown.

Stratosphere Laboratory. A labeled dataset with malicious and benign iot network traffic (iot-23).

Yuan, X., Liu, S., Feng, W., and Dauphin, G. (2023). Feature importance ranking of random forest-based end-to-end learning algorithm. *Remote Sensing*, 15(21).