CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
MESTRADO PROFISSIONAL EM AUTOMAÇÃO E SISTEMAS

FÁBIO ARAUJO FABRES

FARAONIC: UM FRAMEWORK HÍBRIDO PARA DETECÇÃO DE ANOMALIAS EM TEMPO REAL EM REDES OT COM MODBUS/TCP

LEOPOLDINA

2026

FÁBIO ARAUJO FABRES

FARAONIC: UM FRAMEWORK HÍBRIDO PARA DETECÇÃO DE ANOMALIAS EM TEMPO REAL EM REDES OT COM MODBUS/TCP

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Automação e Sistemas do Centro Federal de Educação Tecnológica de Minas Gerais, Campus Leopoldina, como parte dos requisitos para obtenção do título de Mestre em Automação e Sistemas.

Orientador: Prof. Dr. Fabiano Pereira Bhering

Coorientador: Prof. Dr. Silvio Ereno Quincozes

LEOPOLDINA

2026

Fabres, Fábio Araujo.
F123f Faraonic: um framework híbrido para detecção de anomalias em tempo real em redes OT com modbus/TCP. / Fábio Araujo Fabres. – 2026.
97 f. : il.

Dissertação apresentada ao Programa de Pós-Graduação em Automação e Sistemas (PPGAS), 2026.
Orientador: Fabiano Pereira Bhering.
Coorientador: Silvio Ereno Quincozes.
Bibliografia.
Dissertação (mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais. *Campus* Leopoldina.

1. Segurança cibernética. 2. Computadores – Redes – Medidas de segurança. 3. Aprendizado computacional. 4. Segurança industrial. 5. Protocolo de comunicação. I. Bhering, Fabiano Pereira. II. Quincoze, Silvio Ereno. III. Título.

CDU: 004.72.052.52

Elaboração da ficha catalográfica pela bibliotecária Luzia Adriana Damasceno, CRB 6º 2305 / Cefet/MG c*ampus* Leopoldina.

MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM AUTOMAÇÃO E SISTEMAS - LP

ATA DE DEFESA DE DISSERTAÇÃO Nº 11 / 2025 - PPGAS (11.52.20)

Nº do Protocolo: 23062.065813/2025-82

Leopoldina-MG, 11 de dezembro de 2025.

DISSERTAÇÃO DE MESTRADO

FARAONIC: UM FRAMEWORK HÍBRIDO PARA DETECÇÃO DE ANOMALIAS EM TEMPO REAL EM REDES OT COM MODBUS/TCP

AUTOR: FÁBIO ARAUJO FABRES

ORIENTADOR: Professor Doutor FABIANO PEREIRA BHERING

COORIENTADOR EXTERNO: Professor Dr. SILVIO ERENO QUINCOZES

A Banca Examinadora composta pelos membros abaixo aprovou em 11 de dezembro de 2025 essa Dissertação:

Professor Doutor Fabiano Pereira Bhering (ORIENTADOR)

Centro Federal de Educação Tecnológica de Minas Gerais – CEFET/MG

Professor Dr. Silvio Ereno Quincozes (COORIENTADOR)

UNIPAMPA

Professora Doutora Gabriella Castro Barbosa Costa (MEMBRO INTERNO)

Centro Federal de Educação Tecnológica de Minas Gerais – CEFET/MG

Professor Doutor José Geraldo Ribeiro Júnior (MEMBRO INTERNO)

Centro Federal de Educação Tecnológica de Minas Gerais – CEFET/MG

Professor Doutor Diego Luis Kreutz (MEMBRO EXTERNO)

UNIPAMPA

(Assinado digitalmente em 12/12/2025 12:26)
FABIANO PEREIRA BHERING
PROFESSOR ENS BASICO TECN TECNOLOGICO
DECOMLP (11.61.11)
Matrícula: 2556430

(Assinado digitalmente em 11/12/2025 18:15)
GABRIELLA CASTRO BARBOSA COSTA
PROFESSOR ENS BASICO TECN TECNOLOGICO
DECOMLP (11.61.11)
Matrícula: 2933153

(Assinado digitalmente em 11/12/2025 17:53)
JOSE GERALDO RIBEIRO JUNIOR
DIRETOR - TITULAR
DCLP (11.61)
Matrícula: 1322715

(Assinado digitalmente em 11/12/2025 17:59)
DIEGO LUIS KREUTZ
ASSINANTE EXTERNO
CPF: ###.###.200-##

(Assinado digitalmente em 12/12/2025 11:06)
SILVIO ERENO QUINCOZES
ASSINANTE EXTERNO
CPF: ###.###.260-##

Visualize o documento original em **https://sig.cefetmg.br/public/documentos/index.jsp** informando seu número: **11**, ano: **2025**, tipo: **ATA DE DEFESA DE DISSERTAÇÃO**, data de emissão: **11/12/2025** e o código de verificação: **b7a1fa326b**

Dedico este trabalho, primeiramente, a Deus, fonte inesgotável de sabedoria, força e fé, cujo amor e presença guiaram cada passo desta jornada.

À minha família, minha base segura e inspiração diária: à minha esposa e filhos, por compreenderem minhas ausências, apoiarem meus sonhos e compartilharem comigo cada vitória; à minha mãe e irmão, pelo incentivo constante e por ensinarem o valor do esforço, perseverança e humildade.

AGRADECIMENTOS

Expresso meus sinceros agradecimentos ao Prof. Dr. Fabiano Bhering, meu orientador, pela dedicação, paciência e sábias orientações que tornaram possível a concretização deste trabalho. Seu apoio constante foi fundamental para o sucesso desta pesquisa.

Agradeço igualmente ao Prof. Dr. Silvio E. Quincozes, meu coorientador, pelo incentivo contínuo, pelas contribuições técnicas precisas e pelo compartilhamento generoso de sua vasta experiência profissional e acadêmica.

Estendo minha gratidão aos professores do Programa de Pós-Graduação em Automação e Sistemas do CEFET-MG Campus Leopoldina, pela excelência acadêmica e pelos ensinamentos transmitidos ao longo da minha formação. Aos colegas de mestrado, agradeço pelo convívio enriquecedor, pela troca de experiências e pelo companheirismo nos momentos de estudo, desafios e conquistas.

Por fim, agradeço a todos que, direta ou indiretamente, colaboraram para a realização deste trabalho, contribuindo com sugestões, apoio técnico e motivacional, tornando esta jornada mais leve e frutífera.

“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Romanos 12, 2)

RESUMO

O crescimento da conectividade entre redes de Tecnologia da Informação (IT) e de Tecnologia Operacional (OT) ampliou a superfície de ataque de infraestruturas industriais, especialmente em protocolos como Modbus/TCP, que carecem de autenticação e criptografia. Para enfrentar esse cenário, este trabalho apresenta o FARAONIC (*Framework for Anomaly Recognition and Analysis in Operational Networks for Industrial Cybersecurity*), uma abordagem híbrida de detecção que integra inspeção profunda de pacotes (DPI) e aprendizado de máquina supervisionado para identificar anomalias e intrusões em tempo real sem comprometer a disponibilidade operacional. O objetivo central é detectar comportamentos maliciosos em redes OT baseadas em Modbus/TCP por meio de análise determinística e comportamental combinadas. A metodologia envolveu a construção de um ambiente experimental virtualizado e a geração de um *dataset* rotulado contendo mais de 1,5 milhão de pacotes distribuídos em seis diferentes tipos de ataques. O módulo de DPI apresentou desempenho elevado em ataques com assinatura clara, enquanto o módulo de aprendizado de máquina alcançou acurácia superior a 99%, destacando-se especialmente na detecção de ataques de mascaramento. Os resultados demonstram que a integração entre DPI e aprendizado de máquina forma uma estratégia eficaz, complementar e interpretável para a segurança de redes OT, capaz de operar em tempo real e de cobrir diferentes perfis de ataque. O estudo contribui para o avanço da segurança cibernética industrial ao disponibilizar um *framework* replicável, uma metodologia de detecção interpretável e um novo *dataset* rotulado de tráfego Modbus/TCP, ampliando as possibilidades de pesquisa e aplicação prática em detecção e resposta a incidentes em ambientes industriais.

Palavras-chaves:Segurança Cibernética; Modbus/TCP; ICS; *OT Security*; Detecção de Intrusões; DPI; Aprendizado de Máquina; Segurança Industrial; Dataset Rotulado.

ABSTRACT

The increasing interconnection between Information Technology (IT) and Operational Technology (OT) networks has expanded the attack surface of industrial infrastructures, especially in protocols such as Modbus/TCP, which lack authentication and encryption mechanisms. To address this scenario, this work presents *FARAONIC* (Framework for Anomaly Recognition and Analysis in Operational Networks for Industrial Cybersecurity), a hybrid detection approach that integrates Deep Packet Inspection (DPI) and supervised Machine Learning to identify anomalies and intrusions in real time without compromising operational availability. The main objective is to detect malicious behaviors in OT networks based on Modbus/TCP through combined deterministic and behavioral analyses. The methodology involved the construction of a virtualized experimental environment and the generation of a labeled dataset containing more than 1.5 million packets distributed among six different types of attacks. The DPI module achieved high performance in attacks with clear signatures, while the Machine Learning module reached accuracy above 99%, excelling particularly in the detection of masquerade attacks. The results demonstrate that the integration of DPI and Machine Learning forms an effective, complementary, and interpretable strategy for OT network security, capable of operating in real time and covering multiple attack profiles. This study contributes to the advancement of industrial cybersecurity by providing a replicable framework, an interpretable detection methodology, and a new labeled Modbus/TCP traffic dataset, expanding research and practical applications in intrusion detection and incident response within industrial environments.

Keywords: Cybersecurity; Modbus/TCP; ICS; OT Security; Intrusion Detection; DPI; Machine Learning; Industrial Security; Labeled Dataset.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACK Acknowledge.

DCS Distributed Control System.

DMZ Demilitarized Zone.

DoS Denial of Service.

DPI Deep Packet Inspection.

ICS Industrial Control System.

IDS Intrusion Detection System.

IT Information Technology.

MitM Man-in-the-Middle.

ML Machine Learning.

Modbus/TCP Modbus over Transmission Control Protocol.

OT Operational Technology.

PLC Programmable Logic Controller.

RBAC Role-Based Access Control.

SCADA Supervisory Control and Data Acquisition.

SHAP SHapley Additive exPlanations.

ST Structured Text.

SVM Support Vector Machine.

SYN Synchronize.

SYN-ACK Synchronize-Acknowledge.

TAP Test Access Point.

TCP/IP Transmission Control Protocol/Internet Protocol.

SUMÁRIO

1 INTRODUÇÃO

Sistemas de controle industrial, do inglês, Industrial Control System (ICS), são amplamente utilizados no setor industrial para monitorar e controlar processos críticos, como manufatura, energia, transporte e infraestrutura de água Carneiro and Assunção Júnior [2020]. Esses sistemas, que incluem controladores lógicos programáveis, do inglês Programmable Logic Controller (PLC), sistemas de controle distribuído, do inglês, Distributed Control System (DCS), e sistemas de supervisão e aquisição de dados, do inglês, Supervisory Control and Data Acquisition (SCADA), ocupam um lugar central na indústria moderna ao possibilitar o monitoramento e o controle em tempo real. Por esse motivo, a segurança dos ICS é crucial, pois qualquer comprometimento pode resultar em impactos severos para a operação e a segurança física das instalações. Dentre os diversos protocolos empregados nesses ambientes, o Modbus over Transmission Control Protocol (Modbus/TCP) destaca-se por sua simplicidade e ampla difusão, tendo sido originalmente desenvolvido para a troca de dados entre dispositivos em linhas de produção. Com o advento das redes IP, essa tecnologia evoluiu para o chamado Modbus/TCP, que integra a pilha de protocolos Transmission Control Protocol/Internet Protocol (TCP/IP), facilitando a conexão de equipamentos legados a redes mais amplas e promovendo maior flexibilidade no gerenciamento das operações industriais Carneiro and Assunção Júnior [2019].

Entretanto, os sistemas de controle industrial que utilizam o protocolo Modbus/TCP enfrentam desafios de segurança devido à falta de mecanismos nativos de autenticação, confidencialidade e integridade Morris et al. [2021]. Essas vulnerabilidades tornam os sistemas suscetíveis a diversos tipos de ataques cibernéticos, como Denial of Service (DoS), Man-in-the-Middle (MitM) e injeção de dados não utilizados. Além disso, a crescente convergência entre redes de tecnologia operacional (em inglês Operational Technology (OT)) e redes de tecnologia da informação (em inglês Information Technology (IT)) tem aumentado a exposição dos sistemas críticos a ameaças cada vez mais sofisticadas, cujas consequências podem incluir a paralisação de linhas de produção e riscos à segurança física das instalações.

Diante deste cenário, a falta de visibilidade das redes OT e a ausência de ferramentas eficazes para a detecção de anomalias em tempo real evidenciam a fragilidade das infraestruturas de controle. Uma ampla gama de técnicas de defesa, envolvendo autenticação, criptografia, segmentação de rede e sistemas de detecção de intrusão (em inglês, Intrusion Detection System (IDS)), tem sido estudadas para compreender essas vulnerabilidades [Cheminod et al., 2013]. Embora os trabalhos pioneiros, como o de Cheminod et al. [2013], tenham contribuído significativamente para o mapea-

mento de vulnerabilidades e para a proposição de contramedidas em redes industriais, tais estudos apresentam limitações frente aos avanços tecnológicos atuais. Estudos recentes evidenciam que a segurança específica do protocolo Modbus/TCP continua sendo um desafio crítico. Trabalhos como os de Radoglou-Grammatikis et al. [2020] e Nyasore et al. [2020] demonstram que a ausência de mecanismos robustos para a operação em tempo real e o monitoramento contínuo deixa os sistemas vulneráveis a ataques sofisticados. Adicionalmente, investigações de Alsabbagh et al. [2023] e de Wai and Lee [2024] reforçam a necessidade de *frameworks* de segurança que integrem detecção adaptativa e resposta imediata, atendendo aos rigores dos ambientes industriais contemporâneos.

Diante disso, evidencia-se a carência de soluções que possibilitem o monitoramento contínuo, a detecção proativa de ataques e a implementação de respostas rápidas e automatizadas em ambientes industriais. Essa lacuna nas práticas atuais destaca a importância de desenvolver abordagens inovadoras que assegurem a disponibilidade, confiabilidade e segurança das operações, contribuindo para mitigar os riscos associados às vulnerabilidades do protocolo Modbus/TCP.

1.1 Definição do Problema

Os sistemas de segurança para monitoramento e proteção de redes OT, especialmente quando o protocolo Modbus/TCP é empregado, têm evoluído para lidar com ameaças como Denial of Service (DoS) e injeção de dados não utilizados. Soluções propostas na literatura, como as de Cheminod et al. [2012]; Nyasore et al. [2020]; Radoglou-Grammatikis et al. [2020] e Alsabbagh et al. [2023], incluem a utilização de *firewalls* industriais para segmentar a rede, sistemas Intrusion Detection System (IDS) adaptados a protocolos específicos e o uso de inspeção profunda de pacotes (em inglês, Deep Packet Inspection (DPI)) para analisar o tráfego em tempo real.

Embora tais abordagens contribuam para elevar o nível de segurança das redes OT, elas frequentemente enfrentam desafios significativos, como um alto índice de falsos positivos e limitações na detecção de ameaças desconhecidas. Essas deficiências comprometem a visibilidade e a eficácia do monitoramento das redes OT, podendo resultar em perdas financeiras e operacionais caso um ataque não seja identificado e contido com rapidez.

Este trabalho adota explicitamente o modelo de ameaça do tipo *assumed breach*, no qual se considera que o atacante já possui acesso lógico ao segmento OT. Assim, não são tratados mecanismos de prevenção perimetral, autenticação inicial ou controle de acesso à rede, mas sim a detecção de comportamentos anômalos *dentro* do domínio OT, explorando o uso indevido do protocolo Modbus/TCP e seus efeitos

sobre o processo industrial.

Diante deste contexto, a questão central desta pesquisa é:

Como desenvolver um *framework* híbrido de segurança para redes OT baseadas no protocolo Modbus/TCP que ofereça monitoramento contínuo para detecção de anomalias sem interferir na operação industrial?

1.2 Objetivos

Desenvolver um *framework* híbrido para segurança em redes OT baseadas no protocolo Modbus/TCP, que seja capaz de realizar inspeção passiva, detectar anomalias em tempo real e emitir alertas sem interferir na operação industrial, conciliando precisão e baixo impacto nos sistemas.

1.2.1 Objetivos Específicos

a) Implementar um módulo de captura passiva de tráfego por meio de espelhamento de portas (*Port Mirroring*), garantindo a observação do tráfego Modbus/TCP sem interferir na operação industrial;

b) Construir uma *baseline* composta exclusivamente por comunicações Modbus/TCP legítimas, armazenando-a em um banco de dados de referência para a calibração do módulo de DPI, que utilizará esses padrões normais como parâmetro para a detecção de anomalias;

c) Aplicar inspeção profunda de pacotes (DPI) focada na identificação de anomalias relacionadas a *Function Codes*, endereçamento, valores de registradores e consistência do tráfego Modbus/TCP em relação à *baseline* estabelecida;

d) Integrar um sistema de detecção por assinaturas voltado para ataques conhecidos e caracterizáveis por padrões determinísticos no protocolo Modbus/TCP;

e) Desenvolver alertas automatizados baseados nos eventos de detecção, assegurando que o processo de monitoramento permaneça estritamente passivo e sem impacto no desempenho da rede industrial;

f) Simular ataques reais em um ambiente experimental controlado, produzindo um *dataset* rotulado que contemple diferentes classes de anomalias em Modbus/TCP, tais como enumeração, uso indevido de *Function Codes*, injeção de registradores falsos, mascaramento e ataques de DoS;

g) Treinar e avaliar modelos supervisionados de aprendizado de máquina (Machine Learning (ML)) utilizando o *dataset* gerado, comparando seu desempenho com o método baseado em assinaturas por meio de métricas consolidadas, como precisão, revocação, F1-Score e matriz de confusão.

1.3 Organização

Esta dissertação está organizada em sete capítulos, conforme descrito a seguir.

Capítulo 1 – Introdução: apresenta o contexto e a motivação do trabalho, a definição do problema de pesquisa, os objetivos geral e específicos e a metodologia adotada. Também discute a relevância da segurança em redes OT baseadas em Modbus/TCP e a motivação para o uso de abordagens híbridas de detecção.

Capítulo 2 – Fundamentação Teórica: reúne os conceitos essenciais para o entendimento do problema de segurança abordado. São apresentados os princípios da pilha TCP/IP, o funcionamento do Modbus/TCP, seus tipos de registros e vulnerabilidades, bem como os principais ataques associados ao protocolo. O capítulo também discute DPI, aprendizado de máquina aplicado a detecção em OT e desafios da convergência entre redes OT e Information Technology (IT).

Capítulo 3 – Estado da Arte: apresenta um mapeamento crítico das pesquisas que tratam de segurança em redes industriais, com foco em Modbus/TCP e em técnicas de detecção de anomalias. São discutidas abordagens determinísticas, comportamentais e híbridas presentes na literatura, bem como suas limitações.

Capítulo 4 – Proposta: FARAONIC: descreve o *framework* proposto, detalhando sua arquitetura, componentes e modos de operação. O capítulo apresenta o posicionamento do FARAONIC no Modelo *Purdue*, o funcionamento do modo baseado em regras (DPI), o processo de calibração, o algoritmo de detecção e o mecanismo de alerta. Também descreve o modo comportamental baseado em aprendizado de máquina, incluindo seu fluxo interno e requisitos de *dataset*.

Capítulo 5 – Desenvolvimento do Ambiente Experimental: apresenta a construção da infraestrutura utilizada para validar o FARAONIC. São detalhados o ambiente Proxmox, a topologia de rede, a configuração das máquinas virtuais (OpenPLC, Factory I/O, pfSense, Kali Offensive e Kali Defensive), o processo de *port mirroring* e o *pipeline* de captura. O capítulo também descreve a criação do *dataset*, incluindo conversão para CSV, taxonomia de rótulos e características das amostras.

Capítulo 6 – Resultados e Discussão: apresenta a avaliação experimental do *framework*. Para o modo determinístico (DPI), são discutidos os resultados de detecção para diferentes famílias de ataque. Para o modo comportamental, são apresentados os experimentos com modelos supervisionados, o processo de mitigação de *data*

leakage, a análise por valores SHapley Additive exPlanations (SHAP) e a comparação entre modelos e experimentos. Capítulo 7 – Considerações Finais e Trabalhos Futuros: sintetiza as contribuições do trabalho, avalia os resultados obtidos e discute as limitações do *framework*. Também aponta direções para pesquisas futuras, incluindo expansão para múltiplos protocolos industriais, aprimoramento do subsistema de captura, integração com resposta automática e ampliação do *dataset*.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica e as tecnologias essenciais para o entendimento e o desenvolvimento do *framework* de segurança proposto para redes de OT que utilizam o protocolo Modbus/TCP. O objetivo é estabelecer a base conceitual necessária para compreender as particularidades da comunicação industrial, as vulnerabilidades associadas a esses ambientes e os mecanismos de monitoramento e detecção de intrusões considerados ao longo deste trabalho.

Ao longo do capítulo, são descritos os princípios do protocolo TCP/IP, que sustentam a transmissão confiável de dados em redes industriais, seguidos pelo detalhamento do funcionamento do Modbus/TCP, incluindo seus tipos de registros e funções de leitura e escrita. Em seguida, são discutidos os principais tipos de ataques cibernéticos que exploram fragilidades desse protocolo, bem como as técnicas de captura de tráfego, o uso de DPI, a aplicação de aprendizado de máquina em IDS voltados a OT e os desafios decorrentes da convergência entre redes OT e IT.

2.1 Princípios de Comunicação TCP

A comunicação em redes OT frequentemente utiliza a pilha TCP/IP para estabelecer conexões confiáveis entre dispositivos, o que é fundamental para garantir a integridade da transmissão dos dados. No entanto, essa mesma estrutura de comunicação pode ser explorada por agentes maliciosos, especialmente quando combinada com a ausência de mecanismos de autenticação e criptografia em protocolos de aplicação como o Modbus/TCP [Cheminod et al., 2013].

2.1.1 *Three-Way Handshake* e Troca de Mensagens

O processo de estabelecimento de uma conexão TCP é conhecido como *three-way handshake* (aperto de mão em três vias) e consiste nas seguintes etapas [Kurose and Ross, 2016]:

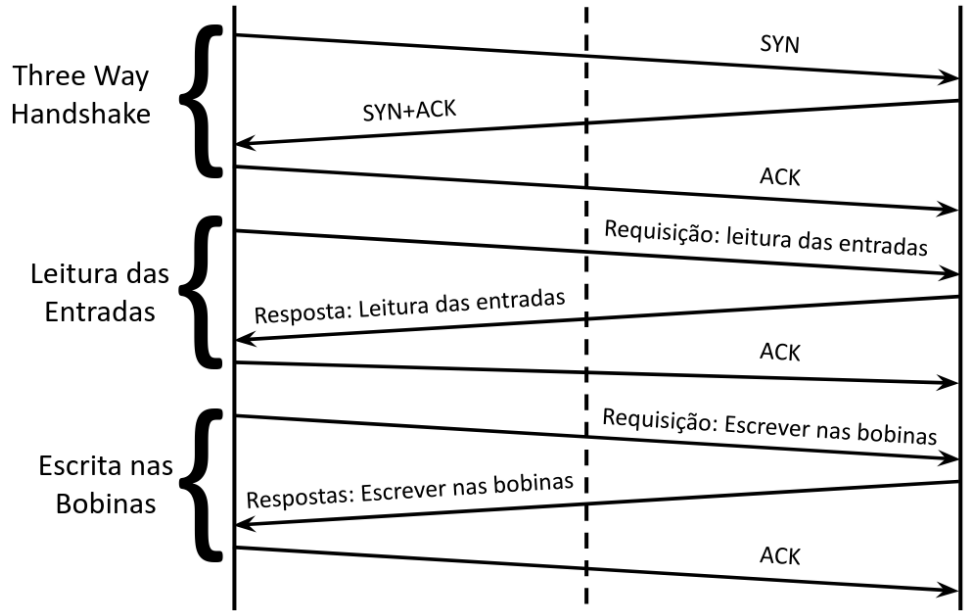
a) Synchronize (SYN): O cliente que deseja iniciar uma comunicação envia um segmento TCP com a flag SYN ativa para o servidor. Este segmento inclui um número de sequência inicial que será usado para rastrear os pacotes.

b) Synchronize-Acknowledge (SYN-ACK): Ao receber o segmento SYN, o servidor responde com um segmento que possui as flags SYN e Acknowledge (ACK) ativadas. O número de sequência do servidor é incluído, bem como a confirmação (ACK) do número de sequência recebido do cliente.

c) Acknowledge (ACK): O cliente envia um último segmento com a flag ACK ativa, confirmando o recebimento do número de sequência do servidor. Neste ponto, a conexão está estabelecida, e a comunicação bidirecional pode ocorrer.

Este processo assegura que tanto o cliente quanto o servidor estejam sincronizados e prontos para trocar dados de maneira confiável. Na Figura 2.1, é ilustrado o processo de *three-way handshake* seguida da requisição de leitura das entradas (*Inputs*) e da escrita das bobinas (*Coils*).

Figura 2.1: Ilustração do processo de *three-way handshake* no protocolo TCP.



Fonte: Elaboração própria, adaptado de [Kurose and Ross, 2016].

Após o estabelecimento da conexão TCP, o mestre envia *requisições Modbus* ao escravo utilizando funções específicas:

a) Função 01 (*Read Coils*): Lê o status de uma série de bobinas no dispositivo escravo.

b) Função 02 *(Read Discrete Inputs*): Obtém o status de entradas digitais.

c) Função 03 (*Read Holding Registers*): Lê valores dos registros de manutenção.

d) Função 04 (*Read Input Registers*): Lê valores de registros de entrada analógicos.

e) Função 05 (*Write Single Coil*): Altera o status de uma única bobina.

f) Função 06 (*Write Single Register*): Escreve um valor em um único registro de manutenção.

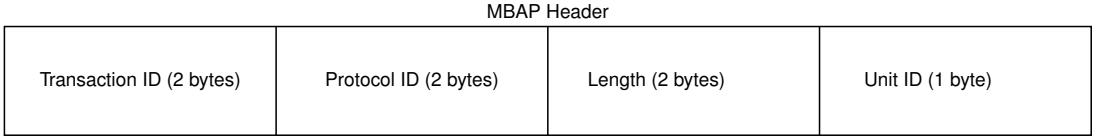
g) Função 15 (*Write Multiple Coils*): Escreve valores em múltiplas bobinas.

h) Função 16 (*Write Multiple Registers*): Escreve valores em múltiplos registros.

Nota: Os códigos de função do protocolo Modbus/TCP não seguem uma numeração contínua para as operações de leitura e escrita. Entre as funções 07 e 15, encontram-se funções destinadas ao diagnóstico e à verificação do estado de exceção dos dispositivos, como a Função 07, que lê o status das exceções, e a Função 08, que abrange operações de diagnóstico. Essas funções são empregadas para monitorar a integridade e a comunicação dos dispositivos, mas não são utilizadas nas operações primárias de controle industrial.

Cada mensagem Modbus inclui um *Modbus Application Protocol Header* (MBAP), que contém informações como identificação da transação, identificação do protocolo, comprimento da mensagem e unidade de identificação. Em seguida, os campos específicos da função são adicionados.

Figura 2.2: Estrutura do MBAP Header (*Modbus Application Protocol Header*).



Fonte: Adaptada de http://www.modbus.org..

2.2 Tipos de Ataques Cibernéticos em Modbus/TCP

Neste trabalho, são considerados cinco tipos principais de comportamentos maliciosos que podem comprometer a integridade e a disponibilidade de sistemas industriais baseados em Modbus/TCP. Esses ataques foram selecionados por representarem tanto ameaças de negação de serviço quanto tentativas de manipulação direta do processo. Embora um dos casos descritos não constitua, por si só, um ataque, ele representa o ponto de partida para a exploração de vulnerabilidades em dispositivos de controle.

2.2.1 Ataque de Negação de Serviço (DoS/DDoS)

Esse tipo de ataque pretende sobrecarregar os dispositivos ou servidores que executam funções de controle, impedindo que requisições legítimas sejam processadas. Em redes industriais, uma inundação de pacotes TCP com a flag SYN ativa ou inúmeras conexões simultâneas na porta 502 pode causar lentidão, falhas de resposta dos controladores ou até a paralisação completa da comunicação entre o sistema supervisório e os Programmable Logic Controller (PLC). O impacto em ambientes operacionais é elevado, pois a indisponibilidade de comunicação pode interromper linhas de produção, causar falhas em válvulas automatizadas e comprometer o monitoramento de variáveis críticas do processo [Cheminod et al., 2012].

2.2.2 Enumeração de *Unit ID* (UNIT_ENUM)

A enumeração não é propriamente um ataque, mas uma etapa de reconhecimento executada antes de ações mais agressivas. Nesse processo, o agente malicioso envia sucessivas requisições Modbus/TCP com diferentes identificadores de unidade (*Unit ID*) a fim de descobrir quais dispositivos estão conectados atrás de um *gateway* ou concentrador industrial [Cheminod et al., 2012, Radoglou-Grammatikis et al., 2020]. O resultado dessa varredura permite ao invasor mapear a topologia da rede e identificar possíveis alvos, como PLC ativos, sensores e atuadores específicos. Esse tipo de sondagem é especialmente perigoso em ambientes que não implementam segmentação de rede ou mecanismos de autenticação entre zonas de controle [Stouffer et al., 2024, Knapp, 2024].

2.2.3 Injeção de Comandos Não Autorizados (*False Command Injection*)

Trata-se da categoria mais ampla de ataques observados em redes industriais e está associada à ausência de autenticação e de controle de integridade no protocolo Modbus/TCP. O invasor envia requisições falsificadas, visando induzir o dispositivo alvo a executar ações não planejadas pelo sistema de controle [Alsabbagh et al., 2023]. Essa categoria foi subdividida em três variantes observadas durante os experimentos realizados neste trabalho:

a) Alteração de Função (FUNC_TAMPER): o invasor modifica apenas o código de função do pacote, trocando, por exemplo, uma operação de leitura (*Read Coils*) por uma de escrita (*Write Multiple Coils*). Dessa forma, comandos aparentemente legítimos passam a ter efeitos diferentes, podendo alterar o estado de atuadores ou modificar parâmetros do processo. [Radoglou-Grammatikis et al., 2020, Nyasore et al., 2020]

b) Alteração de Registradores (FAKE_REG): o atacante envia comandos de

escrita direcionados a registradores não utilizados ou fora das faixas esperadas pelo processo. Essa modificação do *payload* pode resultar na troca de valores críticos, como limites de temperatura, pressão ou *setpoints*, interferindo diretamente na lógica de controle [de Brito and de Sousa Jr, 2022].

c) Mascaramento de Contexto (MASQ): essa variante consiste em reproduzir o contexto de uma comunicação legítima, utilizando o mesmo *Transaction ID*, *Unit ID* e parâmetros válidos, mas com pequenas alterações nos dados de comando. O objetivo é disfarçar o ataque em um fluxo aparentemente normal, dificultando sua identificação. O mascaramento pode ser empregado de forma isolada ou combinado às demais variantes para criar cenários de manipulação gradual de processos industriais [Alsabbagh et al., 2023].

Os cinco tipos de ataques aqui descritos refletem vulnerabilidades inerentes ao protocolo Modbus/TCP e justificam a necessidade de mecanismos contínuos de monitoramento em ambientes de Tecnologia Operacional. Cada uma dessas categorias representa uma forma distinta de comprometimento, seja por sobrecarga, sondagem, falsificação ou mascaramento.

2.3 Técnicas para Capturar o Tráfego

A captura de tráfego em redes OT é essencial para monitorar e proteger infraestruturas críticas contra ataques cibernéticos. Entre as principais técnicas destaca-se o espelhamento de porta (*Port Mirroring*) e os dispositivos TAP (*Test Access Points*).

2.3.1 Espelhamento de Porta (*Port Mirroring*)

O *Port Mirroring* é uma funcionalidade de *switches* que permite copiar o tráfego de uma ou mais portas para uma porta de destino, onde ferramentas de monitoramento podem analisar os dados. Essa técnica é amplamente utilizada devido à sua simplicidade e custo-benefício; porém, apresenta limitações quanto ao impacto no desempenho do *switch* e à possibilidade de perda de pacotes em cenários de tráfego intenso. Um exemplo de aplicação foi destacado em um estudo sobre ataques Man-in-the-Middle (MitM), onde ferramentas como *Ettercap* foram utilizadas para capturar e modificar tráfego em redes Supervisory Control and Data Acquisition (SCADA), explorando vulnerabilidades em protocolos como Modbus/TCP [Radoglou-Grammatikis et al., 2020].

2.3.2 Test Access Point (TAP)

Os dispositivos Test Access Point (TAP) oferecem uma solução mais robusta para a captura de tráfego, permitindo a cópia física dos dados que passam por um

link. Esses dispositivos são independentes de *switches* e garantem uma cópia exata dos pacotes sem interferir no desempenho da rede, sendo ideais para monitoramento em tempo real de sistemas industriais. TAP é particularmente útil em ambientes críticos onde a perda de pacotes é inaceitável, como em plantas industriais que utilizam o protocolo Modbus/TCP para comunicação entre PLC e sistemas SCADA. Um estudo recente destacou o uso de TAP na análise de tráfego Modbus/TCP para detectar ataques de DoS e injeções de comandos falsos, demonstrando a eficácia dessa técnica na identificação de anomalias [de Brito and de Sousa Jr, 2022].

2.4 Deep Packet Inspection (DPI)

A DPI é uma técnica que permite analisar o conteúdo da carga útil dos pacotes de rede, indo além das informações de cabeçalho para identificar comportamentos suspeitos [Kleinman and Barbosa, 2021]. No contexto desta pesquisa, a DPI pode ser utilizada para examinar o tráfego Modbus/TCP e detectar possíveis anomalias ou atividades maliciosas. Trabalhos anteriores mostraram que a DPI é eficaz, embora apresente desafios de latência em ambientes que exigem baixa latência [Goldenberg and Kleinman, 2020].

2.5 Sistemas de Segurança para Monitoramento e Proteção de redes OT

Esta seção será dividida em subseções que tratam dos principais tópicos envolvidos, incluindo Industrial Control System (ICS), protocolos de comunicação utilizados em redes OT, técnicas de detecção de intrusão, inspeção profunda de pacotes e o papel da convergência entre redes OT e IT na segurança cibernética.

2.5.1 Protocolo Modbus/TCP

O protocolo Modbus foi originalmente desenvolvido na década de 1970 pela *Modicon* (atualmente parte da Schneider Electric) para facilitar a comunicação entre dispositivos em ambientes industriais. Com a evolução das redes IP, surgiu o Modbus/TCP, que utiliza a pilha de protocolos TCP/IP para transmitir dados de forma rápida e confiável, permitindo a integração de sistemas legados com tecnologias modernas.[Carneiro and Assunção Júnior, 2019]

Embora não exista um documento oficial que regulamente o funcionamento do Modbus/TCP, há um rascunho que descreve o protocolo na camada de aplicação, intitulado *draft-dube-modbus-applproto-00*[1].

As orientações detalhadas sobre a implementação do Modbus/TCP estão a

[1] Disponível em: https://datatracker.ietf.org/doc/html/draft-dube-modbus-applproto-00

cargo da *Modbus Organization*[2], que disponibiliza um guia[3] detalhado sobre como o Modbus deve ser mapeado sobre TCP/IP.

Para compreender as vulnerabilidades e o funcionamento do protocolo Modbus/TCP, é essencial detalhar a estrutura de dados utilizada na comunicação entre dispositivos. O Modbus/TCP organiza a informação em diferentes tipos de registros, cada um destinado a uma função específica:

a) Discrete Inputs (Entradas Digitais): Entradas de um bit, somente leitura, geralmente conectadas a dispositivos como botões e sensores de estado. Representam sinais discretos que informam o status de condições binárias no sistema.

b) Coils (Bobinas): Saídas de um bit, que podem ser lidas e escritas. Utilizadas para controlar dispositivos como relés, lâmpadas e outros atuadores binários. As bobinas permitem ao mestre ativar ou desativar funções no escravo.

c) Input Registers (Registros de Entrada): Registros de 16 bits, somente leitura, que armazenam valores analógicos provenientes de sensores. Utilizados para monitorar variáveis como temperatura, pressão e fluxo, fornecendo dados detalhados sobre o processo industrial.

d) Holding Registers (Registros de Manutenção): Registros de 16 bits, que podem ser lidos e escritos. Usados para configurar parâmetros operacionais, definir *setpoints* e ajustar valores de controle no dispositivo escravo.

A Tabela 2.1 resume os tipos de registros Modbus e suas características principais.

A comunicação é realizada através de funções predefinidas no protocolo, como leitura de bobinas (*Função 01*), leitura de registros de entrada (*Função 04*), escrita em uma bobina única (*Função 05*) ou escrita em múltiplos registros (*Função 16*).

Entender a estrutura da Tabela 2.1 é crucial para identificar possíveis pontos de ataque. Por exemplo, um invasor pode tentar escrever valores maliciosos em *holding*

[2]A *Modbus Organization* é um grupo independente de usuários e fornecedores de dispositivos de automação que busca promover a adoção do conjunto de protocolos de comunicação Modbus e sua evolução para arquiteturas distribuídas de automação em diversos segmentos de mercado. A organização também fornece a infraestrutura necessária para obter e compartilhar informações sobre os protocolos, suas aplicações e certificações, facilitando a implementação pelos usuários e reduzindo custos. https://www.modbus.org/about_us.php

[3]Disponível em: https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

Tabela 2.1: Tipos de registros Modbus e suas características

Tipo de Registro	Função	Acesso	Tamanho
Discrete Inputs	Entradas digitais	Somente leitura	1 bit
Coils	Saídas digitais	Leitura/Escrita	1 bit
Input Registers	Entradas analógicas	Somente leitura	16 bits
Holding Registers	Parâmetros e configurações	Leitura/Escrita	16 bits

Fonte: Adaptada de http://www.modbus.org..

registers para alterar configurações críticas ou manipular *coils* para controlar diretamente equipamentos. A falta de mecanismos de autenticação no protocolo facilita essas ações se medidas de segurança adicionais não forem implementadas.

2.5.2 Sistemas de Detecção de Intrusão (IDS)

Os IDS desempenham um papel importante na segurança de redes OT, ao monitorar o tráfego de rede em busca de atividades suspeitas. Trabalhos recentes propõem o uso de IDS híbridos, que combinam detecção baseada em assinaturas e detecção comportamental [Barbosa and Caselli, 2022]. A detecção baseada em assinaturas é eficaz para identificar ataques conhecidos, enquanto a detecção comportamental utiliza técnicas de aprendizado de máquina para identificar padrões anômalos e novos tipos de ataques [Morris et al., 2023].

2.6 Aprendizado de Máquina em IDS para OT

Em redes OT, a detecção baseada exclusivamente em assinaturas tende a falhar diante de variações e ataques inéditos. Nesse contexto, técnicas de aprendizado de máquina (ML) têm sido empregadas para reconhecer padrões e desvios no tráfego industrial, complementando as abordagens tradicionais [Pinto et al., 2023]. Em particular, destacam-se três famílias: (i) métodos supervisionados, eficazes para classes de ataque já observadas; (ii) métodos não supervisionados voltados à caracterização do comportamento normal e à detecção de anomalias; e (iii) deep learning, com maior capacidade de modelar dependências complexas, porém usualmente com custo e opacidade maiores [Pinto et al., 2023].

Alinhado a essas diretrizes, este trabalho extrai *features* de duas naturezas: (i) semânticas de Modbus/TCP (código de função, *unit id*, faixas de *reference*/*quantity*, padrões binários de *coils*/*inputs*); e (ii) agregações temporais de curta janela (taxas por função, variações de bitmaps e progressões de *Transaction ID*). Essa combinação privilegia *interpretabilidade* e baixo custo de inferência, requisitos recorrentes reportados para ICS/OT [Pinto et al., 2023], ao mesmo tempo em que possibilita capturar

tanto desvios semânticos (uso indevido de funções/registradores) quanto estruturais (frequências/ordens improváveis no ciclo do processo).

2.6.1 Modelos para o Aprendizado de Máquina

Neste trabalho, optou-se pela árvore de decisão e pela floresta aleatória devido à combinação de *baixo custo*, boa acurácia em cenários supervisionados e interpretabilidade de regras, atributos frequentemente destacados em revisões e estudos comparativos para ICS [Pinto et al., 2023, Shikhaliyev et al., 2023]. A seleção prioriza modelos coerentes com restrições típicas de OT (latência, disponibilidade e auditabilidade).

2.6.1.1 *Decision Tree*

Proporciona regras explícitas (limiares por campo/propriedade), úteis para validação por equipes de operação e análise forense. Em cenários ICS, sua simplicidade favorece implantação e manutenção com dados rotulados limitados [Shikhaliyev et al., 2023].

Além disso, árvores de decisão particionam o espaço de atributos de forma hierárquica, selecionando iterativamente o atributo que maximiza um critério de pureza, como *Gini* ou *Information Gain*, até atingir folhas homogêneas [Pinto et al., 2023]. Essa estrutura permite capturar relações não lineares comuns no tráfego industrial, como faixas específicas de *function codes* ou padrões temporais. Outro benefício destacado em Pinto et al. [2023] é a facilidade de auditoria: cada decisão no fluxo da árvore pode ser rastreada e justificada, característica crítica em ambientes OT sujeitos a requisitos regulatórios e de rastreabilidade.

2.6.1.2 *Random Forest*

O *ensemble* de árvores eleva robustez e estabilidade frente a ruído/variação de tráfego, preservando interpretabilidade global básica (importância de atributos) e mantendo custo de inferência compatível com operação quase em tempo real [Pinto et al., 2023, Shikhaliyev et al., 2023].

A floresta aleatória agrega o resultado de múltiplas árvores construídas com amostragem aleatória de instâncias e subconjuntos de atributos, reduzindo o risco de sobreajuste e aumentando a capacidade de generalização do modelo [Shikhaliyev et al., 2023]. Essa diversidade estrutural permite capturar diferentes aspectos do tráfego Modbus/TCP — desde padrões semânticos de função até variações estruturais de frequência e volume, tornando o modelo menos sensível a anomalias isoladas ou ruídos operacionais. Segundo Pinto et al. [2023], esse equilíbrio entre desempenho e

estabilidade explica a adoção recorrente de *Random Forest* em IDS industriais contemporâneos.

2.6.1.3 Observação sobre alternativas.

SVM e redes neurais podem oferecer ganhos em certas condições, mas tendem a exigir *tuning* mais sensível e apresentam menor transparência do que os modelos baseados em árvore, um ponto crítico em OT, onde explicabilidade e previsibilidade operacional são prioritárias [Pinto et al., 2023, Shikhaliyev et al., 2023].

Os estudos comparativos apresentados em Pinto et al. [2023] mostram que, embora modelos como Support Vector Machine (SVM) e *Multi-Layer Perceptron* alcancem boas taxas de detecção, seu desempenho é altamente dependente de parametrizações específicas (por exemplo, escolha de kernel no SVM ou número de camadas/neurônios nas redes neurais). Além disso, Shikhaliyev et al. [2023] observa que esses modelos carecem frequentemente de mecanismos claros de interpretação, o que dificulta sua adoção em operações industriais que exigem justificativas audíveis, como rastreios de alarmes, investigação pós-incidente e conformidade com normas de segurança.

2.6.1.4 Síntese prática.

A dupla *Decision Tree + Random Forest* atende ao equilíbrio entre desempenho, simplicidade operacional e explicabilidade requeridos neste estudo; além disso, integra-se naturalmente às *features* oriundas de DPI em Modbus/TCP, sem impor dependência de *toolchains* pesadas ou de grandes volumes de dados rotulados [Pinto et al., 2023, Shikhaliyev et al., 2023].

Os resultados reportados em [Pinto et al., 2023] indicam que modelos baseados em árvore apresentam desempenho particularmente consistente quando aplicados a tráfego de protocolos industriais, justamente por conseguirem extrair relações lógicas entre atributos sem necessidade de pré-processamento extensivo. Em paralelo, o uso combinado dessas técnicas favorece ciclos de implantação rápidos, permitindo ajustes incrementais conforme novas variantes de ataque são incorporadas ao *dataset* ou surgem alterações no comportamento normal da planta industrial [Shikhaliyev et al., 2023].

2.7 Convergência entre redes OT e IT

A convergência entre redes de OT e IT trouxe inúmeros benefícios, como a melhoria na eficiência operacional e a integração de dados, mas também introduziu novas vulnerabilidades. Dispositivos OT, tradicionalmente isolados, passaram a estar conec-

tados a redes corporativas e, consequentemente, expostos a ameaças cibernéticas que antes eram exclusivas do domínio de IT [Assunção Júnior, 2018].

Em arquiteturas industriais modernas, essa integração ocorre por meio de controladores, *gateways*, servidores de histórico e sistemas de supervisão conectados por redes IP. Embora essa conectividade viabilize maior automação e monitoramento, ela também rompe a barreira que antes separava o ambiente operacional do ambiente corporativo. Assim, ataques direcionados à infraestrutura de IT podem, direta ou indiretamente, atingir sistemas industriais críticos.

Para reduzir os riscos dessa convergência, recomenda-se uma abordagem de *Defesa em Profundidade* (*Defense in Depth*), que combina medidas técnicas, procedimentais e organizacionais. Essa abordagem, amplamente descrita em normas como a IEC 62443 e a NIST SP 800-82, define múltiplas camadas de proteção e monitoramento entre os níveis corporativo e operacional. As principais práticas incluem:

a) Segmentação de Redes: criar zonas e condutos lógicos ou físicos, separando claramente os domínios OT e IT com o uso de *firewalls* industriais, zonas desmilitarizadas (Demilitarized Zone (DMZ)) e *gateways* seguros;

b) Controle de Acesso: aplicar políticas de autenticação multifator, autorização baseada em função (Role-Based Access Control (RBAC)) e restrições de privilégios mínimos, limitando a interação de usuários e dispositivos apenas ao necessário;

c) Monitoramento Contínuo: empregar sistemas de detecção de intrusão (IDS) especializados em protocolos industriais e técnicas de inspeção profunda de pacotes (DPI) para observar o tráfego em tempo real;

d) Criptografia na Camada de Borda: utilizar canais seguros nas fronteiras entre OT e IT, quando tecnicamente viável, para mitigar interceptações e manipulações de dados;

e) Gestão de Vulnerabilidades e Atualizações: manter inventários atualizados de ativos, realizar varreduras periódicas e aplicar correções (*patches*) de segurança de forma controlada para minimizar a superfície de ataque.

A aplicação conjunta dessas medidas permite reduzir significativamente o risco de propagação de incidentes entre redes corporativas e ambientes de controle, fortalecendo a resiliência das infraestruturas críticas sem comprometer os requisitos de disponibilidade e tempo real característicos das redes OT.

2.8 Conclusão

A revisão realizada neste capítulo permitiu consolidar os fundamentos essenciais para a compreensão do problema de segurança tratado neste trabalho. Inicialmente, foram discutidos os princípios de comunicação em redes industriais, com destaque para a pilha TCP/IP e para a estrutura do protocolo Modbus/TCP, evidenciando vulnerabilidades inerentes à ausência de autenticação e criptografia. Em seguida, foram analisados os principais tipos de ataques que exploram essas fragilidades, bem como as técnicas de captura de tráfego e de inspeção profunda de pacotes empregadas para monitorar a comunicação em ambientes de OT.

Foram também apresentados os conceitos centrais de sistemas de detecção de intrusão voltados a redes industriais, incluindo o uso de DPI e de técnicas de aprendizado de máquina em IDS específicos para OT, além dos desafios decorrentes da convergência entre redes OT e IT. Esses elementos fornecem a base conceitual necessária para o desenvolvimento do *framework* proposto nos capítulos seguintes, que combina inspeção semântica de tráfego Modbus/TCP, captura passiva e mecanismos de detecção orientados às particularidades de redes operacionais.

3 ESTADO DA ARTE

Diversos pesquisadores têm explorado soluções para a segurança em redes OT, especialmente no contexto do protocolo Modbus/TCP. Nesta seção, analisamos os principais trabalhos relacionados ao tema, destacando suas contribuições e limitações.

3.1 Revisão da Literatura

Radoglou-Grammatikis et al. [2020] investigaram ataques cibernéticos no protocolo Modbus/TCP e desenvolveram um sistema de detecção baseado em anomalias. Utilizaram um IDS para monitorar o tráfego e detectar ataques de DoS e outras ameaças. Embora tenham mostrado bons resultados para ataques conhecidos, o método apresenta limitações quanto à detecção de variações inéditas e depende de atualizações frequentes do modelo para manter a efetividade. No contexto deste trabalho, essa limitação é particularmente relevante: ao contrário do sistema proposto por Radoglou-Grammatikis et al. [2020], o *framework* desenvolvido aqui integra inspeção profunda de pacotes de forma passiva, via *port mirroring*, evitando o impacto de latência de soluções *inline*[1], e combina detecção baseada em assinaturas com mecanismos comportamentais calibrados sobre o tráfego legítimo da planta. Esse caráter híbrido permite detectar tanto ataques com padrão conhecido quanto desvios sutis na semântica Modbus, ampliando a cobertura enquanto preserva a disponibilidade do ambiente operacional.

Nyasore et al. [2020] demonstraram que o uso de DPI na análise do tráfego do protocolo Modbus/TCP é eficaz para identificar comportamentos anômalos e mitigar ataques. Entretanto, os autores também enfatizam que a aplicação de DPI de forma *inline*, ou seja, realizando a inspeção detalhada de todo o tráfego dentro do próprio caminho de comunicação, pode introduzir um *overhead* computacional significativo. Esse processamento intensivo resulta em latência e jitter adicionais que podem comprometer a eficácia das operações de sistemas industriais, as quais exigem comunicação quase em tempo real para garantir a segurança e a precisão dos controles.

No entanto, esse problema não se aplica à proposta desenvolvida neste trabalho. O *framework* aqui apresentado executa a DPI de forma **passiva**, por meio de *port mirroring*, analisando uma cópia do tráfego sem interferir diretamente no fluxo crítico entre mestre e escravo. Essa arquitetura elimina o impacto de latência ca-

[1] *Inline* refere-se a mecanismos inseridos diretamente no caminho de comunicação entre mestre e escravo, de modo que cada pacote precisa ser processado antes de seguir adiante. Essa abordagem, comum em firewalls e IDS ativos, pode introduzir *latência* e *jitter* indesejados em redes industriais.

racterístico das soluções baseadas em inspeção *inline* e mantém o requisito de alta disponibilidade essencial para redes OT. Assim, o framework evita o overhead apontado em Nyasore et al. [2020] ao deslocar a carga de processamento para um sensor dedicado, preservando tanto o desempenho quanto a previsibilidade do sistema de controle industrial.

De maneira semelhante, Mahendra et al. [2021] abordaram a implementação de DPI para a identificação de vulnerabilidades em sistemas de controle industrial. Segundo esses autores, embora a análise detalhada do tráfego permita a detecção precisa de padrões maliciosos, ela também impõe um custo extra em termos de latência, especialmente em cenários de tráfego intenso. Essa limitação é crítica em ICS, onde mesmo pequenos atrasos podem impactar a operação e a segurança.

No entanto, essa restrição não se aplica à abordagem adotada neste trabalho. Enquanto Mahendra et al. analisam soluções *inline*, nas quais o pacote precisa ser processado antes de chegar ao destino, o *framework* proposto utiliza *port mirroring* para capturar o tráfego de maneira passiva. Nesse arranjo, a DPI é realizada fora do caminho operacional, em um sensor dedicado, eliminando o impacto de latência sobre a comunicação entre mestre e escravo. Assim, o mecanismo de coleta garante que o sistema permaneça compatível com os requisitos de tempo quase real característicos de redes OT.

de Brito and de Sousa Jr [2022] apresentaram um *testbed* de código aberto para simular ciberataques e analisar a segurança de plantas industriais, incluindo plantas nucleares, utilizando o protocolo Modbus/TCP. O *testbed* permite a reprodução de cenários de ataque, o que é útil para avaliar a eficácia de técnicas de mitigação. Contudo, a falta de integração com técnicas automatizadas de detecção limita a capacidade do *testbed* de responder em tempo real a incidentes de segurança.

Ilgner and Fujdiak [2022] propuseram um *framework* que realiza *fuzzing* nos protocolos ICS, especificamente no protocolo Modbus/TCP, para identificar vulnerabilidades potenciais em dispositivos de automação industrial. A abordagem é eficaz para testar a robustez dos dispositivos e descobrir falhas não documentadas. No entanto, o foco está na identificação de vulnerabilidades, não fornecendo soluções para mitigar os problemas descobertos.

Alsabbagh et al. [2023] demonstraram um ataque furtivo de injeção de dados não utilizados em sistemas SCADA baseados no protocolo Modbus/TCP [Alsabbagh et al., 2023]. O método explora a falta de autenticação e criptografia no protocolo para realizar ataques que passam despercebidos pelo operador. Apesar de destacar uma vulnerabilidade crítica, o trabalho não propõe soluções para prevenir tais ataques.

A Tabela 3.1 apresenta uma comparação das características dos trabalhos rela-

cionados sobre segurança em redes OT. Para facilitar a compreensão, as abreviações utilizadas na tabela são definidas a seguir:

a) Abord. Híbrida: Abordagem Híbrida.

b) Amb. Indus.: Simulação de Ambiente Industrial.

c) Det. Anom.: Foco em Detecção de Anomalias.

d) ML: Uso de *Machine Learning*.

e) Testbed: *Testbed* Aberto e Reprodutível.

f) Temp. Real: Detecção em Tempo Real.

g) Mitigação: Mitigação Ativa de Ameaças.

h) Fuzzing: Capacidade de Fuzzing.

Tabela 3.1: Comparação das características dos trabalhos relacionados sobre segurança em redes OT.

Trabalhos	Abord. Híbrida	Amb. Indus.	Det. Anom.	ML	Testbed	Temp. Real	Mitigação	Fuzzing
Este trabalho	✓	✓	✓	✓	✓	✓	✓	×
[Radoglou-Grammatikis et al., 2020]	×	×	✓	×	×	✓	×	×
[Nyasore et al., 2020]	×	×	✓	×	×	✓	✓	×
[Mahendra et al., 2021]	×	×	✓	×	×	✓	×	×
[de Brito and de Sousa Jr, 2022]	×	✓	×	×	✓	×	×	×
[Ilgner and Fujdiak, 2022]	×	×	×	×	×	×	×	✓
[Alsabbagh et al., 2023]	×	✓	×	×	×	×	×	×

Fonte: Própria (2025)..

3.1.1 Síntese Crítica e Lacunas Identificadas na Literatura

A análise da literatura revela que, embora existam contribuições relevantes para a segurança em redes OT, cada abordagem apresenta limitações importantes que comprometem sua aplicabilidade em cenários industriais reais. Os trabalhos baseados exclusivamente em detecção por anomalias, como o de Radoglou-Grammatikis et al. (2020), dependem de modelos constantemente atualizados e apresentam dificuldade em lidar com variações inéditas de ataque. Já as soluções que utilizam DPI de forma *inline*, como discutido em Nyasore et al. [2020], Mahendra et al. [2021], sofrem com aumento de latência e *jitter*, efeitos incompatíveis com os requisitos de tempo quase real em redes industriais. Outras iniciativas, como testbeds puramente experimentais [de Brito and de Sousa Jr, 2022] ou propostas voltadas à identificação

de vulnerabilidades sem mecanismos de mitigação [Ilgner and Fujdiak, 2022], contribuem para o entendimento dos ataques, mas não oferecem soluções práticas de detecção contínua.

Além disso, observa-se a ausência de abordagens que integrem simultaneamente: (i) inspeção semântica do tráfego Modbus/TCP; (ii) mecanismos comportamentais baseados em padrões reais da planta; e (iii) operação não intrusiva que não interfira no ciclo de controle. Nenhum dos trabalhos revisados combina análise determinística de baixo nível, como validação de *function codes*, faixas de registradores e coerência de *Transaction IDs*, com um mecanismo adaptativo capaz de reconhecer desvios progressivos e ataques encobertos, como *false command injection* furtivos [Alsabbagh et al., 2023]. Persistem também limitações relativas à escassez de *datasets* realistas construídos a partir de tráfego industrial representativo.

Essas lacunas indicam a necessidade de uma solução que concilie robustez, interpretabilidade e operação segura em tempo quase real, características fundamentais em ambientes OT. Os capítulos seguintes apresentam uma proposta que busca responder a essas deficiências de forma integrada.

As principais contribuições deste trabalho podem ser resumidas nos seguintes pontos:

a) *framework* Específico para Modbus/TCP: Desenvolvimento de uma solução de segurança dedicada a redes OT que utilizam o protocolo Modbus/TCP, assegurando a operação em tempo real sem prejuízo do desempenho.

b) Abordagem Híbrida de Detecção: Implementação de dois mecanismos complementares de detecção: (i) uma camada determinística baseada em DPI, operando em tempo real, e (ii) uma camada comportamental baseada em aprendizado de máquina, treinada sobre o *dataset* gerado em ambiente industrial simulado. Embora os módulos funcionem de forma independente (regras ou ML), a arquitetura do *framework* foi concebida para permitir sua integração futura em um *pipeline* unificado, ampliando a capacidade de detectar tanto ataques conhecidos quanto variações inéditas.

c) Validação Experimental em Ambiente Simulado: Implementação e teste do *framework* em um *testbed* virtualizado que reproduz cenários reais de operação industrial, possibilitando a avaliação prática da eficácia da solução.

d) Melhoria na Visibilidade e Resposta a Incidentes: Aumento da visibilidade do tráfego em redes OT e facilitação da resposta rápida e automatizada a potenciais ameaças, contribuindo para a continuidade e a segurança das operações industriais.

3.2 Conclusão

A revisão da literatura evidenciou que, embora existam esforços relevantes voltados à detecção de anomalias, à aplicação de DPI e à construção de *testbeds* experimentais, nenhuma dessas iniciativas oferece uma solução integrada que concilie operação em tempo quase real, análise semântica do tráfego Modbus/TCP e mecanismos comportamentais capazes de ampliar a capacidade de detecção. Também ficou claro que abordagens *inline* introduzem atrasos incompatíveis com o ciclo de controle industrial, ao passo que soluções passivas são mais adequadas para preservar a disponibilidade.

Esse conjunto de análises fundamenta a motivação para o *framework* proposto nos capítulos seguintes, que busca preencher as lacunas identificadas ao combinar inspeção profunda de pacotes de forma passiva, detecção baseada em assinaturas e uma arquitetura preparada para extensões comportamentais futuras.

4 PROPOSTA: FARAONIC

O FARAONIC — *Framework for Anomaly Recognition and Analysis in Operational Networks for Industrial Cybersecurity* (em português: *Framework para Reconhecimento e Análise de Anomalias em Redes Operacionais para Cibersegurança Industrial*), é um sistema desenvolvido para monitorar e detectar comportamentos anômalos em redes de OT que utilizam o protocolo Modbus/TCP. O código-fonte e a implementação completa encontram-se disponíveis em `https://github.com/f4bio89/FARAONIC`.

O objetivo do sistema é proporcionar visibilidade sobre a comunicação entre os PLC, permitindo identificar tentativas de ataque ou alterações indevidas no tráfego de controle industrial.

a) Modo baseado em regras: utiliza DPI e assinaturas específicas do protocolo Modbus/TCP para detectar padrões de tráfego maliciosos ou fora do comportamento esperado;

b) Modo baseado em ML: analisa o comportamento do tráfego a partir de modelos treinados com características extraídas de comunicações legítimas e de cenários de ataque, permitindo identificar anomalias que não são capturadas por regras fixas.

O módulo de aprendizado de máquina não substitui o mecanismo baseado em regras, mas o complementa em cenários onde padrões comportamentais sutis não podem ser descritos de forma determinística.

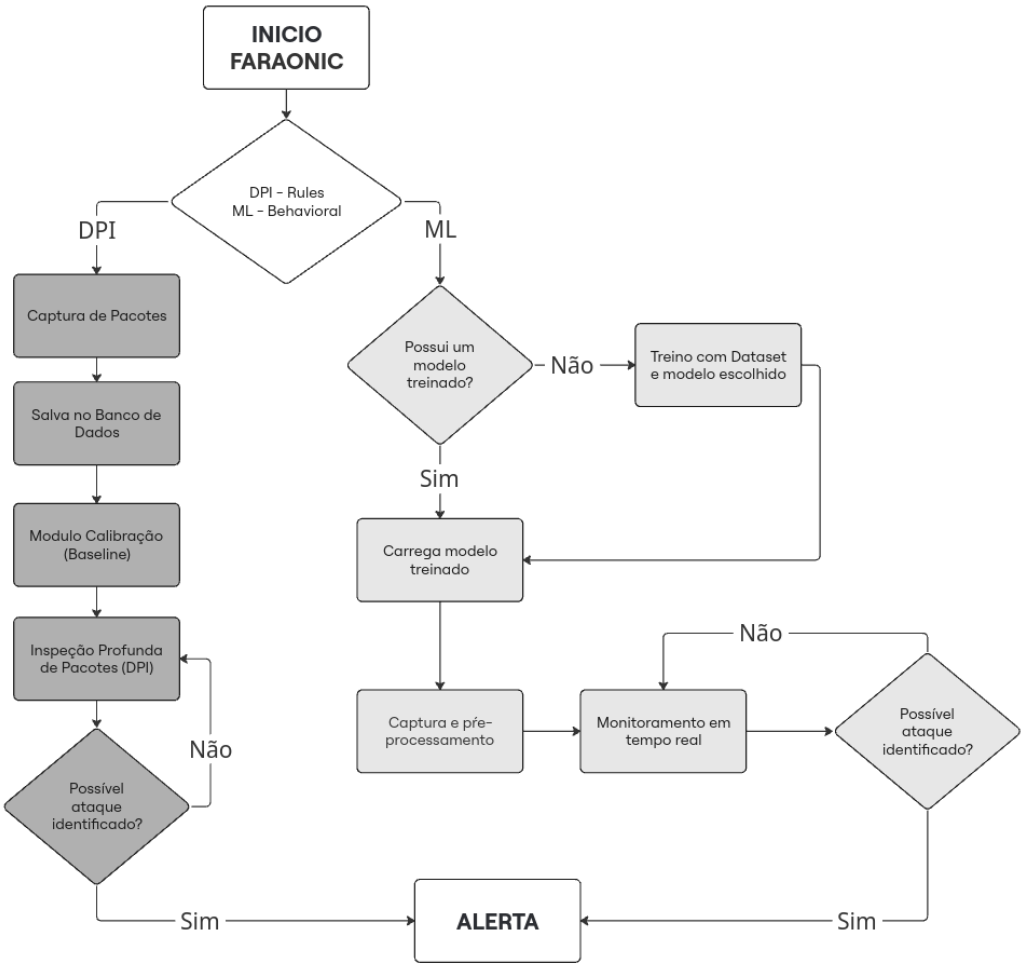
Esses dois modos de operação permitem que o sistema atue tanto de forma determinística, por meio de regras definidas, quanto de forma adaptativa, por meio da análise comportamental, tornando o FARAONIC adequado a diferentes contextos e níveis de maturidade em segurança de redes industriais.

No modo baseado em regras (*RULES*), o processo inicia-se com a captura de pacotes provenientes da rede OT. Esses pacotes são armazenados no banco de dados, onde passam por uma etapa de calibração, responsável por definir a *baseline* de funcionamento normal do sistema. Em seguida, o módulo de DPI extrai campos específicos do protocolo Modbus/TCP, como códigos de função, identificadores de unidade e tipos de registro, comparando-os com os parâmetros calibrados. O módulo de detecção aplica então as regras configuradas para identificar comportamentos fora do padrão, como acessos não autorizados ou alterações indevidas nos registradores. Quando uma anomalia é confirmada, o mecanismo de alerta notifica o operador ou registra o evento para posterior análise.

No modo baseado em aprendizado de máquina (*BEHAVIORAL*), o funcionamento segue uma lógica distinta. É necessário, primeiramente, dispor de um *dataset* rotulado, contendo exemplos de tráfego legítimo e de ataques simulados. A partir desse conjunto de dados, o sistema executa o treinamento de um modelo supervisionado, como uma árvore de decisão ou uma floresta aleatória, capaz de reconhecer padrões de comportamento. Uma vez treinado, o modelo é utilizado para realizar a inferência em tempo real, classificando as amostras capturadas conforme sua similaridade com os padrões aprendidos. Quando uma amostra é classificada como anômala, o resultado é encaminhado ao mesmo mecanismo de alerta, assegurando uniformidade na resposta operacional.

A Figura 4.1 apresenta uma visão geral do fluxo operacional do *framework*, destacando o fluxo de dados compartilhado entre os modos e a bifurcação dos caminhos de análise, por regras ou por aprendizado de máquina.

Figura 4.1: Fluxo operacional do FARAONIC, com análise por regras e por aprendizado de máquina.



Fonte: Própria (2025)..

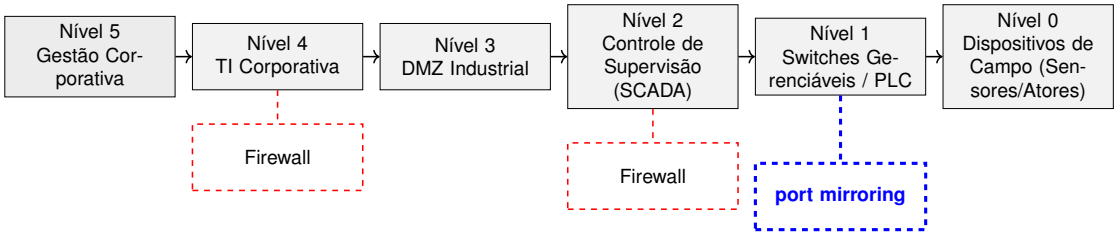
4.1 Posicionamento do FARAONIC

O posicionamento do FARAONIC foi definido com base no Modelo *Purdue*, amplamente utilizado como referência para a segmentação de redes industriais. Esse modelo organiza os ativos em níveis hierárquicos, do nível 0 (dispositivos de campo) ao nível 5 (rede corporativa) [Stouffer et al., 2024], promovendo a separação entre os domínios de IT e OT e a aplicação de políticas de segurança específicas para cada camada.

Como ilustrado na Figura 4.2, a arquitetura de referência recomenda a presença de dois *firewalls* entre a rede corporativa (nível 4) e os sistemas industriais (níveis 0 a 3), geralmente com DMZ entre eles. Essa configuração tem como objetivo reduzir a superfície de ataque dos sistemas de controle, protegendo os ativos industriais contra

acessos indevidos oriundos da rede corporativa ou da Internet [Stouffer et al., 2024].

Figura 4.2: Modelo Purdue, destacando o posicionamento da captura de tráfego entre os níveis 1 e 2.



Fonte: Própria (2025)..

A captura do tráfego utilizada pelo FARAONIC é feita de forma passiva, sem interferir no funcionamento da rede industrial. Para isso, é empregada a técnica de *port mirroring*, configurada diretamente em *switches* gerenciáveis posicionados no Nível 1 do modelo *Purdue*, camada responsável pela interligação entre os PLC e os dispositivos de supervisão do Nível 2.

O *port mirroring* consiste em instruir o *switch* a duplicar o tráfego de determinadas portas ou VLANs para uma porta de monitoramento dedicada, à qual é conectado o sensor de coleta do FARAONIC. Dessa forma, todo o fluxo Modbus/TCP que transita entre os equipamentos de campo e o sistema supervisório é copiado em tempo real, permitindo a análise completa dos pacotes sem afetar o processo produtivo. Essa abordagem é amplamente utilizada em ambientes de teste e simulação, pois não exige alterações físicas de cabeamento nem interrupções de operação, fatores críticos em sistemas industriais.

Apesar de depender da capacidade de processamento do *switch*, a técnica oferece praticidade e baixo custo de implantação, sendo suficiente para cenários controlados ou de volume moderado de tráfego. Em implementações reais de grande escala, onde a integridade da captura precisa ser garantida mesmo sob altas taxas de transmissão, pode-se recorrer ao uso de dispositivos TAP. Esses equipamentos são instalados fisicamente entre dois pontos da rede e replicam o tráfego de maneira totalmente transparente, sem risco de perda de pacotes. No entanto, por exigirem intervenção no cabeamento e custo adicional de hardware, os TAP são mais adequados a ambientes de produção, enquanto o *port mirroring* é particularmente adequado para laboratórios e simulações controladas.

4.2 Modo de Operação por Regras (DPI)

O modo baseado em regras realiza DPI e compara cada pacote com parâmetros e assinaturas calibrados. É indicado para ambientes que exigem decisões previsíveis e explicáveis, identificando desvios com critérios objetivos extraídos do tráfego legítimo.

A seguir, são descritos os principais componentes que integram o modo de operação por regras, bem como a lógica de funcionamento do algoritmo de detecção.

4.2.1 Captura de Pacotes

Este módulo é responsável por interceptar o tráfego de rede e armazenar amostras de pacotes que representam o comportamento normal da comunicação Modbus/TCP. Essas amostras são utilizadas para ajustar parâmetros, refinar regras e, eventualmente, compor o conjunto de dados usado por outros módulos do *framework*. Este módulo é utilizado de forma semelhante nas duas formas de operação do FARAONIC.

4.2.2 Banco de Dados (BD)

O banco de dados atua como repositório central do *framework*, armazenando os pacotes capturados e seus metadados (*timestamp*, endereços IP de origem e destino, códigos de função Modbus/TCP, etc.).

Durante o processo de ingestão, os registros capturados são normalizados e adaptados ao modelo documental adotado, de forma a facilitar consultas e filtragens posteriores.

4.2.3 Módulo de Calibração

O módulo de calibração tem como objetivo estabelecer uma linha de base (*baseline*) de funcionamento normal da rede OT. Durante essa etapa, são analisados os pacotes armazenados no banco de dados para identificar os parâmetros legítimos de comunicação, como endereços IP confiáveis, identificadores de unidade (*Unit ID*), códigos de função e tipos de registradores Modbus/TCP. Essas informações formam o perfil de tráfego esperado, contra o qual futuras comunicações serão comparadas.

A calibração não ocorre continuamente, sendo executada apenas quando há mudanças significativas no ambiente, como:

a) Reprogramação ou atualização da lógica de controle dos PLC;

b) Alterações na topologia ou nas políticas de acesso na rede;

c) Introdução de novos dispositivos ou modificações nos ciclos de processo;

d) Detecção de divergências (*drift*) entre o tráfego atual e o perfil de referência.

Fora desses eventos, a *baseline* permanece válida, evitando reprocessamentos desnecessários e mantendo a eficiência operacional.

4.2.4 Inspeção Profunda de Pacotes (DPI)

A DPI é o componente responsável por realizar a análise detalhada das mensagens Modbus/TCP capturadas pela rede. Seu papel é extrair, decodificar e normalizar os campos de interesse de cada pacote, transformando dados brutos em informações estruturadas que possam ser utilizadas pelos módulos de calibração e detecção.

Durante essa etapa, o conteúdo do pacote é desmembrado, permitindo identificar parâmetros como o código de função (*Function Code*), o identificador de unidade (*Unit ID*), os endereços de registradores e os valores transmitidos. Esses elementos são padronizados e armazenados em formato legível, compondo uma base de eventos coerente para análise posterior.

Embora a adoção de criptografia, como TLS, possa proteger a comunicação, sua implementação em sistemas industriais é limitada devido ao aumento da latência e da carga computacional, o que pode comprometer requisitos de tempo real [Knapp, 2024]. Por esse motivo, o FARAONIC adota a inspeção passiva e transparente, garantindo visibilidade total do tráfego sem interferir na operação da rede OT.

4.2.5 Módulo de Detecção por Regras

O módulo de detecção utiliza as informações produzidas pela DPI para identificar comportamentos anômalos ou maliciosos. A partir dos campos normalizados, o sistema compara cada pacote com os parâmetros definidos na calibração e aplica regras determinísticas baseadas nos parâmetros da *baseline*.

Essa etapa constitui o núcleo da operação por regras: é nela que o tráfego analisado é confrontado com o perfil legítimo da rede. As condições típicas de detecção incluem:

a) IP não autorizado: tentativas de conexão ao servidor Modbus/TCP oriundas de endereços fora da lista de confiança;

b) Unit ID inválido: consultas a identificadores de controladores inexistentes, típicas de varreduras de enumeração;

c) Código de função não permitido: uso de comandos Modbus/TCP não contemplados na operação calibrada, possivelmente indicativo de abuso ou manipulação de registradores;

d) Dados fora do padrão: valores de leitura ou escrita incompatíveis com o estado esperado do processo ou com os limites definidos na calibração.

4.2.5.1 Algoritmo de Detecção

O Algoritmo 1 ilustra, de forma simplificada, uma lógica genérica para o modo de detecção baseado em regras do FARAONIC, que pode ser adaptada conforme o ambiente e o conjunto de políticas adotadas.

Algorithm 1 Lógica de inspeção e detecção de anomalias Modbus/TCP.

```
1: while pacote ← CapturarPacote() do
2:     if ETráfegoModbusTCP(pacote) = verdadeiro then
3:         if IPConfiavel(pacote.ip) = falso then
4:             if IndicadorSobrecargaConexao(pacote) = verdadeiro then
5:                 AtualizarJanelaConexao(pacote.ip)
6:                 if ExcedeLimiarConexaoLocal() ou ExcedeLimiarConexaoGlobal() then
7:                     GerarAlerta("Padrão compatível com DoS/DDoS detectado")
8:                 end if
9:             else
10:                GerarAlerta("Atividade de origem não confiável")
11:            end if
12:            continue
13:        end if
14:        if PossuiCamadaAplicacao(pacote) = verdadeiro then
15:            {funcao, unidade, idTransacao, recurso, parametros} ← ExtrairCamposAplicacao(pacote)
16:            if UnidadeAutorizada(unidade) = falso then
17:                GerarAlerta("Identificador de unidade não autorizado")
18:                continue
19:            end if
20:            if FuncaoAutorizada(funcao) = falso then
21:                GerarAlerta("Function Code fora do padrão esperado")
22:                continue
23:            end if
24:            if OperacaoDeEscrita(funcao) e RecursoAutorizado(recurso, parametros) = falso then
25:                GerarAlerta("Operação de escrita em recurso não autorizado")
26:            end if
27:            if MudancaInconsistente(idTransacao, recurso, parametros) = verdadeiro then
28:                GerarAlerta("Possível mascaramento de transação detectado")
29:            end if
30:        end if
31:    end if
32: end while
```

4.2.6 Mecanismo de Alerta

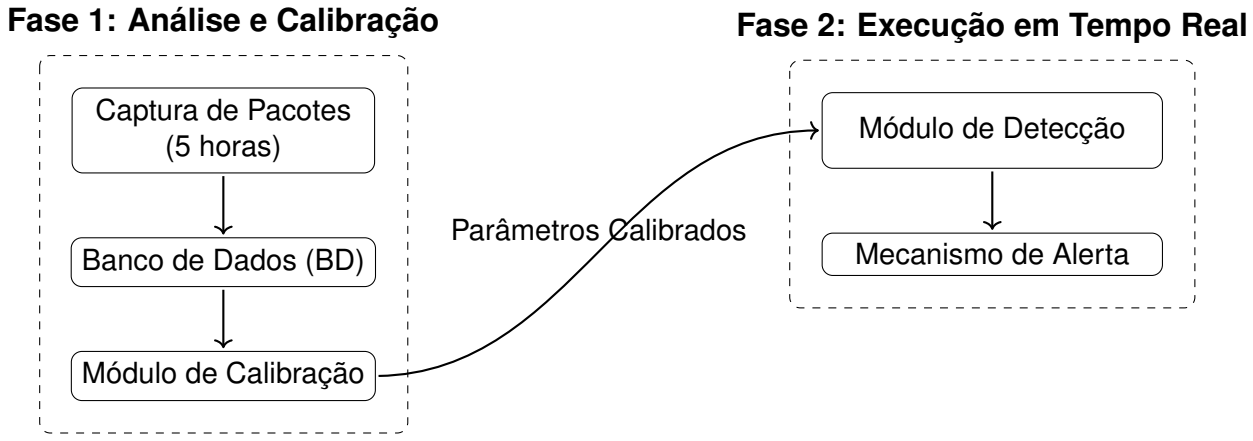
O mecanismo de alerta é o componente responsável por comunicar ao operador os eventos identificados como anômalos pelo módulo de detecção. Ele representa a etapa final do fluxo de monitoramento, convertendo as decisões técnicas do *framework* em informações acionáveis para a resposta operacional.

As notificações podem ser exibidas em tempo real durante a execução do módulo de detecção, permitindo que o analista visualize imediatamente o tipo de anomalia e o endereço de origem associado.

4.2.7 Operação em Duas Fases

O modo de operação baseado em regras do FARAONIC foi estruturado em duas fases complementares: a Fase 1, voltada à análise e calibração do ambiente, e a Fase 2, responsável pela execução contínua da detecção em tempo real. Essa divisão permite que o *framework* se adapte às características específicas de cada rede OT, mantendo a precisão na identificação de anomalias sem comprometer o desempenho operacional. A Figura 4.3 ilustra o encadeamento dessas etapas.

Figura 4.3: Fluxo de Operação do *framework* em Duas Fases



Fonte: Própria (2025)..

4.2.7.1 Fase 1 — Análise e Calibração

Na primeira fase, o objetivo é compreender o comportamento normal da rede e gerar parâmetros de referência (*baseline*) que servirão de base para a detecção posterior. O módulo de captura de pacotes é conectado de forma passiva ao *switch* por meio de um *port mirroring*, durante uma janela de observação suficientemente longa para capturar os ciclos típicos de operação do processo industrial (por exemplo, algumas horas).

Os pacotes capturados são enviados ao banco de dados, onde são armazenados e normalizados para análise. O módulo de calibração, então, processa esses registros e extrai as informações mais relevantes para a caracterização do tráfego legítimo, como endereços IP de origem e destino, identificadores de unidade (*Unit ID*) e tipos de registradores utilizados. A partir dessas informações, o sistema constrói uma lista de IPs confiáveis e define os padrões de comunicação esperados, formando a *baseline* do ambiente. Essa etapa é fundamental para reduzir falsos positivos, pois fornece um retrato fiel do comportamento real da rede antes de iniciar a fase de detecção.

4.2.7.2 Fase 2 — Execução em Tempo Real

Na segunda fase, o *framework* passa a operar de forma contínua, monitorando o tráfego da rede em tempo real com base nos parâmetros definidos na calibração. Os módulos de detecção e alerta entram em funcionamento simultâneo, avaliando cada pacote Modbus/TCP recebido na porta 502 e comparando seus campos com o *baseline* previamente estabelecido.

O processo de análise segue uma sequência determinística: as informações extraídas pela DPI (IP de origem, *Unit ID*, *Function Code*, registradores/valores e metadados Modbus) são comparadas ao *baseline* e a limiares temporais, resultando nas seguintes detecções:

a) IP não confiável: qualquer atividade de conexão ou tráfego Modbus/TCP oriunda de endereço fora da lista de IPs legítimos gera alerta informativo e registro para auditoria.

b) DoS local: detecção de rajadas de pacotes SYN acima de um limiar por IP (dentro de uma janela temporal), ou aumento súbito agregado com múltiplas origens distintas (caracterizando distribuição). Há *debounce* temporal para evitar tempestade de alertas.

c) Unit ID divergente: quando o *Unit ID* observado difere do *Unit ID* legítimo calibrado, indicando enumeração de controladores ou acesso indevido.

d) *Function Code* não permitido: uso de códigos Modbus/TCP fora do conjunto calibrado (por exemplo, tentativa de executar funções não previstas na operação normal).

e) Escrita em registradores (*coil*) fora do padrão: para requisições de escrita de *coil*, se o padrão de *coil* em hexadecimal não pertencer ao conjunto de valores legítimos, sinaliza tentativa de escrita indevida.

f) *Masquerade*: inconsistência entre o identificador de transação observado em sequência e os pares de números de referência e de estados dos registradores; variações anômalas entre requisições correlatas sugerem encobrimento de ação maliciosa.

Sempre que uma dessas condições é verificada, o mecanismo de alerta é acionado, exibindo notificações em tempo real no terminal ou registrando os eventos para análise posterior. Se todas as verificações forem atendidas, o pacote é considerado legítimo e descartado do processo de detecção. Esse fluxo permite que o FARAONIC mantenha vigilância constante sobre a rede OT, reagindo imediatamente a qualquer comportamento suspeito sem interferir na continuidade dos processos industriais.

Em conjunto, as duas fases tornam o *framework* adaptável a diferentes cenários, equilibrando a necessidade de aprendizado inicial com a agilidade necessária para respostas em tempo real.

4.3 Modo de Operação por Aprendizado de Máquina (ML)

No modo ML, o FARAONIC detecta anomalias a partir de padrões aprendidos por um algoritmo de classificação. Esse modo pode ser ativado de duas formas:

a) Com modelo já treinado: quando há um classificador previamente treinado e validado para o processo alvo, o sistema apenas carrega o modelo e executa a inferência em tempo real.

b) Com treinamento local: quando não existe um modelo adequado, é necessário treinar um classificador com dados do próprio ambiente em que o FARAONIC será implantado. Isso garante que o modelo aprenda as características reais daquele processo (equipamentos, ciclos, códigos de função, faixas de valores etc.).

Independentemente da opção, a detecção por ML não trabalha diretamente com os pacotes brutos; ela opera sobre atributos pré-processados (forma tabular) extraídos do tráfego.

4.3.1 Fluxo genérico do modo ML

O modo baseado em aprendizado de máquina (*ML*) do FARAONIC segue um fluxo operacional estruturado para transformar o tráfego da rede OT em instâncias analisáveis, treinar um modelo supervisionado quando necessário e executar a classificação em tempo real durante a operação. Esse processo é composto por quatro etapas

principais: captura e pré-processamento, treinamento, inferência e alerta. Organizadas conforme a Figura 4.4, que evidencia em que momento ocorre o treinamento e como o modelo é utilizado durante a execução do sistema

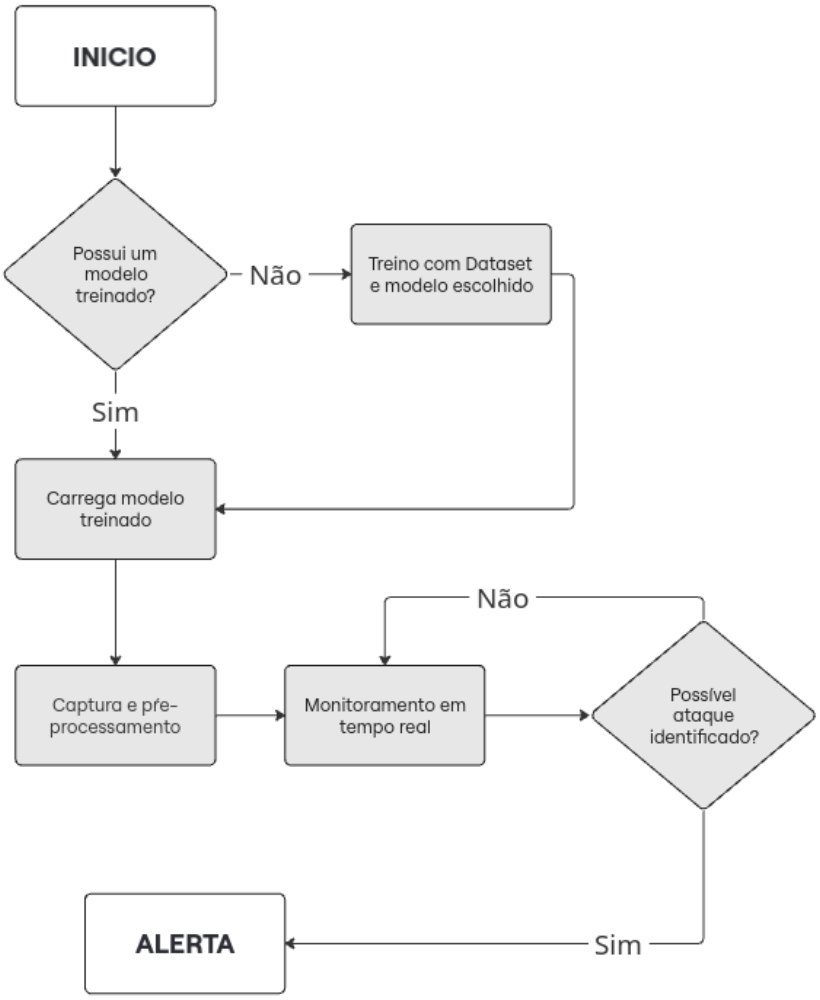
a) Captura e pré-processamento - O tráfego é coletado passivamente (por exemplo, via *port mirroring*). Em seguida, cada mensagem Modbus/TCP é decodificada e convertida em atributos numéricos e categóricos, como endereço IP de origem, *Unit_ID*, *Function_Code*, endereços de registrador, contagens e intervalos de mensagens, comprimentos de payload e padrões de valores. O resultado é uma tabela estruturada em que cada linha representa uma amostra e cada coluna representa um atributo extraído.

b) Treinamento (quando necessário) - Caso não exista um modelo previamente treinado, constrói-se um *dataset* com exemplos do próprio ambiente, incluindo tráfego normal e, quando possível, cenários de ataque simulados ou históricos. Esse conjunto é rotulado e dividido em treino e validação para ajustar um classificador supervisionado (por exemplo, árvore de decisão, floresta aleatória ou *gradient boosting*), priorizando baixo custo de inferência e boa interpretabilidade. Ao final, obtém-se um modelo treinado pronto para uso em produção.

c) Inferência em tempo real - Durante a operação, novas amostras passam pelo mesmo pré-processamento e são classificadas pelo modelo treinado. Para cada amostra, o classificador retorna uma classe prevista, como NORMAL ou um tipo específico de ataque, e, opcionalmente, uma medida de confiança que auxilia na interpretação do resultado.

d) Ação e alerta - Caso a classe prevista indique comportamento anômalo, o FARAONIC aciona o módulo de alerta, registrando o evento e notificando o operador. Esse mecanismo garante uniformidade na resposta operacional, independentemente do tipo de anomalia identificada.

Figura 4.4: Fluxo genérico de operação do modo baseado em aprendizado de máquina no FARAONIC.



Fonte: Própria (2025)..

4.4 Conclusão

Em síntese, o modo baseado em aprendizado de máquina expande a capacidade de detecção do FARAONIC ao capturar relações e padrões de comportamento que não podem ser expressos de forma determinística por regras fixas. A integração entre DPI, calibração e classificação supervisionada cria um mecanismo híbrido capaz de responder tanto a desvios claros e estruturados quanto a variações sutis no tráfego Modbus/TCP. Dessa forma, o *framework* consolida-se como uma proposta flexível, extensível e adequada a diferentes graus de complexidade e maturidade em ambientes industriais.

A combinação dos dois modos, determinístico e comportamental, forma uma arquitetura complementar capaz de operar em cenários estáveis, mas também de se

adaptar a ambientes que sofrem alterações dinâmicas. Essa convergência reforça o papel do FARAONIC como um *framework* coerente, modular e aplicável a diferentes contextos de redes OT.

5 DESENVOLVIMENTO DO AMBIENTE EXPERIMENTAL

Este capítulo apresenta o processo de construção e configuração de um ambiente experimental voltado à implementação e avaliação do framework de detecção de intrusões para redes OT que utilizam o protocolo Modbus/TCP. O objetivo principal é reproduzir, em escala de laboratório, condições semelhantes às encontradas em ambientes industriais, permitindo a análise de vulnerabilidades e a validação das medidas de proteção discutidas nos capítulos anteriores.

Inicialmente, descreve-se a topologia de rede adotada, evidenciando a utilização de máquinas virtuais (VMs) e dispositivos simulados que compõem o cenário de testes. Em seguida, detalham-se as configurações específicas de cada componente – desde o firewall *pfSense* até as VMs com Debian 13, Kali Linux e o Windows rodando o software Factory I/O. Posteriormente, são apresentadas as etapas de comunicação entre o OpenPLC e o Factory I/O, ilustrando como o protocolo Modbus/TCP é empregado para controlar sensores e atuadores simulados.

Além de descrever a arquitetura e a configuração da infraestrutura, o capítulo também aborda o processo de monitoramento e testes de segurança realizados na rede, destacando os desafios práticos enfrentados. Por fim, discute-se como esse ambiente experimental subsidia a avaliação do framework de segurança, bem como a identificação de possíveis limitações e oportunidades de aprimoramento. A Figura 5.1 apresenta a interface de linha de comando (CLI) do FARAONIC, sintetizando os modos de operação utilizados ao longo dos experimentos.

Figura 5.1: Interface de linha de comando (CLI) do FARAONIC, mostrando os modos de operação *Rules* (baseado em assinaturas) e *Behavioral* (baseado em aprendizado de máquina).

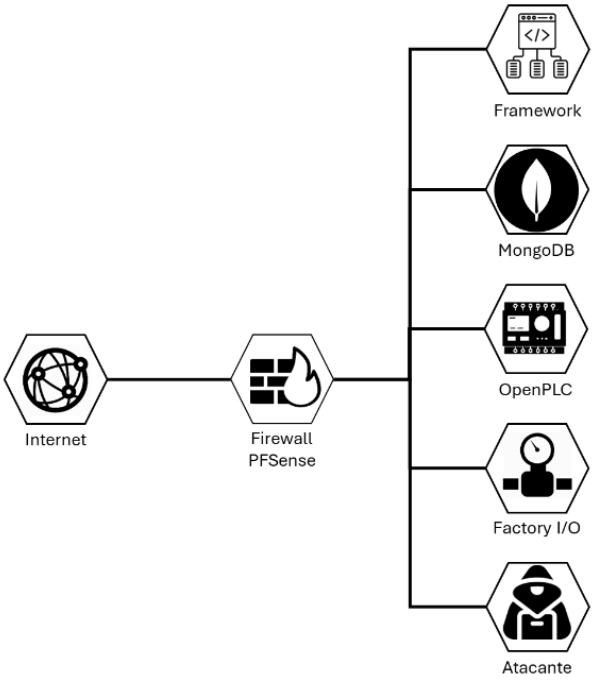
Fonte: Própria (2025)..

5.1 Topologia de Rede e Componentes do Laboratório

A Figura 5.2 mostra a topologia do ambiente experimental. O laboratório é composto por seis máquinas virtuais interconectadas em duas redes (bridges) do Proxmox: `vmbr30` e `vmbr254`, que simulam, respectivamente, os domínios OT e IT. A comunicação principal entre PLC e processo ocorre na sub-rede `192.168.30.0/24` (vmbr30), enquanto a sub-rede `192.168.254.0/24` (vmbr254) é dedicada à telemetria, administração e coleta de dados.

No centro da topologia apresentada encontra-se o *pfSense*, responsável por atuar como firewall e roteador, segmentando e controlando o tráfego entre as redes OT e IT. O OpenPLC, instalado em uma VM *Debian 13*, desempenha o papel de *Modbus Master*, enviando comandos de controle e leitura para o *Factory I/O*, configurado em uma VM *Windows 11* e operando como *Modbus Slave*. Essa comunicação ocorre sobre a rede 192.168.30.0/24 (bridge vmbr30), dedicada exclusivamente ao domínio operacional, onde circulam os pacotes Modbus/TCP.

Figura 5.2: Diagrama da estrutura da rede do laboratório



Fonte: Própria (2025)..

A infraestrutura também inclui duas máquinas *Kali Linux*, com funções distintas. A primeira, denominada Kali Defensive (192.168.30.30), executa o framework FARAONIC para monitoramento e detecção de intrusões em tempo real, analisando o tráfego capturado na rede OT. A segunda, chamada Kali Offensive (192.168.30.40), é utilizada para a simulação de ataques Modbus, incluindo tentativas de *Denial of Service* (DoS), *fake registers*, *masquerade*, enumeração de *Unit IDs*, entre outras. Essa configuração permite, em um mesmo ambiente virtualizado, tanto a condução de experimentos defensivos quanto ofensivos, de forma controlada e reprodutível.

Ressalta-se que, no modelo experimental adotado, o nó *Kali Offensive* representa um atacante já presente no segmento OT (rede `192.168.30.0/24`), com conectividade direta aos ativos industriais. Dessa forma, os experimentos não abrangem a etapa de intrusão inicial, mas sim ações pós-comprometimento, como abuso do protocolo Modbus/TCP, enumeração e interferência na operação do processo.

O ambiente ainda conta com uma VM adicional *Debian 13* dedicada ao MongoDB (192.168.30.15), responsável pelo armazenamento dos dados capturados e estruturados pelo framework FARAONIC. Dessa forma, os resultados das coletas, calibrações e execuções de detecção podem ser registrados e analisados posteriormente, sem interferência na rede de controle. Todo o ambiente foi projetado no hipervisor Proxmox, o que possibilita a criação de redes isoladas, gerenciamento centralizado,

realização de *port mirroring* e *snapshots* para restauração de estados de teste, mantendo a integridade dos experimentos.

A Tabela 5.1 apresenta a relação completa dos dispositivos utilizados, suas funções e respectivos endereços IP nas duas redes (`vmbr30` e `vmbr254`). O objetivo dessa estrutura é agrupar os principais elementos de um sistema industrial real, incluindo o controle lógico programável (OpenPLC), o processo simulado (Factory I/O), os mecanismos de proteção (*pfSense* e Kali Defensive) e os vetores de ataque (Kali Offensive).

Tabela 5.1: Configurações de rede dos dispositivos no ambiente Proxmox

Sistema Operacional	**Função**	**OT / vmbr30**	**IT / vmbr254**
pfSense (FreeBSD)	Firewall / Roteador	192.168.30.1	–
Debian 13	Banco *MongoDB*	192.168.30.15	192.168.254.15
Debian 13	OpenPLC Runtime (*Master*)	192.168.30.20	192.168.254.20
Windows 11	Factory I/O (*Slave*)	192.168.30.25	192.168.254.25
Kali Linux	*Defensive* (FARAONIC)	192.168.30.30	192.168.254.30
Kali Linux	*Offensive* (Red Team)	192.168.30.40	192.168.254.40

Fonte: Própria (2025)..

Adicionalmente, foram adotadas práticas de segurança recomendadas na literatura para ambientes industriais, incluindo segmentação de rede, uso de firewalls dedicados e aplicação de regras de controle de acesso estritas. Tais medidas visam reduzir a superfície de ataque e preservar a integridade e a disponibilidade dos sistemas críticos [Knapp, 2024]. Esse conjunto de boas práticas torna o ambiente laboratorial uma representação fiel e segura de uma rede OT real, apta para experimentação de mecanismos de detecção e resposta a intrusões.

5.2 OpenPLC como Master Modbus/TCP

O OpenPLC é uma plataforma de controle lógico programável de código aberto, compatível com a norma IEC 61131-3, que possibilita a criação de aplicações em diferentes linguagens (como *Ladder Diagram, Structured Text*, entre outras). Neste laboratório, o OpenPLC assume o papel de *Modbus Master*, enviando comandos para o *Factory I/O* (*Slave*) e recebendo informações dos sensores simulados.

5.2.1 Instalação e Acesso

A instalação do OpenPLC Runtime foi realizada em uma máquina virtual (IP 192.168.30.20), seguindo as instruções oficiais[1]. Uma vez instalado, o serviço do

[1] `https://autonomylogic.com`

OpenPLC é executado em `/etc/init.d/`, permitindo inicialização automática junto ao sistema.

Para configurar e monitorar o comportamento do PLC virtual, utiliza-se a *interface web* acessível via navegador na porta 8080, digitando o endereço `http://192.168.30.20:8080`. Por meio dessa interface, é possível:

a) Importar e Gerenciar Programas: Carregar arquivos de lógica de controle.

b) Configurar drivers: Ajustar parâmetros de comunicação Modbus ou outros protocolos suportados.

c) Monitorar Variáveis: Visualizar, em tempo real, valores de entradas, saídas e registradores intermediários.

5.2.2 Desenvolvimento do Programa de Controle

A lógica do PLC foi reimplementada em Structured Text (ST), uma das linguagens padronizadas pela norma IEC 61131-3. O ST é uma linguagem textual semelhante a linguagens de programação convencionais, como Pascal ou C, e permite descrever o comportamento do controlador de forma sequencial e estruturada. Essa abordagem facilita a criação de lógicas mais complexas, o uso de estruturas de decisão, temporizadores, variáveis auxiliares e intertravamentos, mantendo a clareza e a modularidade do código.

No programa desenvolvido, o controle é baseado em um sequenciador de estados representado pela variável Step_1, responsável por coordenar o funcionamento da célula automatizada composta por uma esteira transportadora, um atuador de garra e um mecanismo de descarte de caixas. Cada estado do sequenciador representa uma etapa do processo, transporte, descida, fechamento, subida, rotação e retorno, sendo a transição entre eles determinada pelas condições lógicas e pelos sensores do sistema. O princípio de operação segue a seguinte ordem:

a) Condição de execução: `RUN := StartButton AND SafetyDoor`.

b) Estados principais (`CASE Step_1 OF`): 0 (Transporte), 1 (Descida), 2 (Fechamento/Validação), 3 (Subida), 4 (Rotação CCW/Descarte), 5 (Abertura/Retorno) e 10 (Parada de Segurança).

c) Temporizadores TON: `T_Subida` (1,5 s na validação; 0,7 s na rotação; 0,5 s no retorno) e `T2` (2 s para retentativas quando `valid = FALSE`).

d) Segurança e retomada: intertravamentos impedem rotação simultânea (CW/CCW) e `Step_2` guarda o último estado para retomada segura após `RUN = FALSE`.

Algoritmo 1 - Lógica de controle da esteira com garra em Structured Text

Listing 5.1: Trecho do código Structured Text (ST) do programa do PLC basicstyle

```
RUN := StartButton AND SafetyDoor;
CASE Step_1 OF
  0: Conveyor_Move := RUN AND (NOT Picker_Holding_Box);
     IF RUN AND Box_Waiting AND (NOT Picker_Holding_Box) THEN
        Conveyor_Move := FALSE; Step_1 := 1; END_IF;
  1: Picker_Move_Down := TRUE; Picker_Grab := TRUE;
     IF NOT Picker_Moving_Z THEN Step_1 := 2; END_IF;
  2: T_Subida(IN := TRUE, PT := T#1500ms);
     IF T_Subida.Q THEN
        IF Picker_Holding_Box THEN Step_1 := 3;
        ELSE valid := FALSE; Step_1 := 5; END_IF;
     END_IF;
  3: IF NOT Picker_Moving_Z THEN Step_1 := 4; END_IF;
  4: Picker_Rotate_CCW := TRUE;
     T_Subida(IN := TRUE, PT := T#700ms);
     IF T_Subida.Q THEN Picker_Rotate_CCW := FALSE; Step_1 := 5;
     END_IF;
  5: IF valid THEN
          Picker_Grab := FALSE; Picker_Rotate_CW := TRUE;
          T_Subida(IN := TRUE, PT := T#500ms);
          IF T_Subida.Q THEN valid := FALSE; Step_1 := 0; END_IF;
     ELSE
          T2(IN := TRUE, PT := T#2000ms);
          IF T2.Q THEN valid := TRUE; END_IF;
     END_IF;
 10: IF StartButton AND SafetyDoor THEN Step_1 := Step_2; END_IF;
END_CASE;
IF (NOT RUN) AND (Step_1 <> 10) THEN
  Step_2 := Step_1; Step_1 := 10;
END_IF;
IF Picker_Rotate_CW THEN Picker_Rotate_CCW := FALSE; END_IF;
IF Picker_Rotate_CCW THEN Picker_Rotate_CW := FALSE; END_IF;
```

Durante a execução cíclica realizada pelo OpenPLC, o código em ST avalia continuamente as variáveis de entrada recebidas do Factory I/O via Modbus/TCP, processa as instruções de controle e atualiza as saídas correspondentes. Essa rotina ga-

rante a sincronização entre a lógica de comando e o comportamento físico simulado da célula automatizada, assegurando o funcionamento correto da esteira, da garra e da porta de segurança.

Como resultado, qualquer mudança detectada pelos sensores, como a chegada de uma caixa ao ponto de coleta ou a abertura inesperada da porta de segurança, é imediatamente refletida nas variáveis internas do PLC, permitindo que o sistema responda de forma rápida e previsível, interrompendo operações ou ajustando o ciclo conforme a lógica implementada.

5.2.3 Configuração do Cliente Modbus

Para que o OpenPLC possa comunicar-se com o *Factory I/O*, é necessário configurar um dispositivo do tipo *Modbus TCP* na interface web do OpenPLC. A Figura 5.3 ilustra essa tela de configuração, destacando os campos principais:

a) Device Name: Rótulo utilizado para identificar o alvo Modbus, aqui nomeado como `Factory I/O`.

b) Device Type: Selecionado como `Generic Modbus TCP Device`, indicando que se trata de um dispositivo escravo Modbus acessível via TCP.

c) Slave ID: Número de identificação do escravo, configurado como 5 de acordo com o *Factory I/O*.

d) IP Address: Definido em `192.168.30.25`, o endereço estático atribuído ao *Factory I/O*.

e) IP Port: `502`, porta padrão para o protocolo Modbus/TCP.

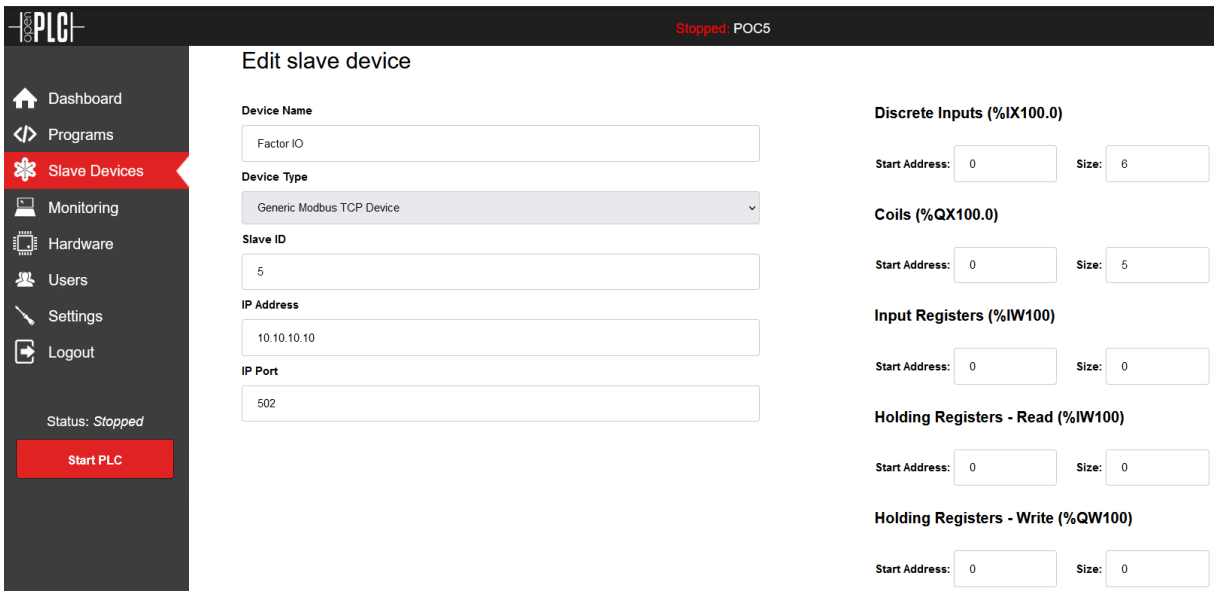
f) Discrete Inputs (%IX100.0): `Start Address = 0`, `Size = 6`, indicando que as primeiras 6 entradas digitais (*Inputs*) iniciam no endereço 0.

g) Coils (%QX100.0): `Start Address = 0`, `Size = 5`, definindo 5 saídas digitais (*Coils*) a partir do endereço 0.

h) Input Registers (%IW100): `Start Address = 0`, `Size = 0`, não utilizados no cenário atual.

i) Holding Registers – Read (%IW100) e Write (%QW100): Também iniciados em 0, mas com tamanho 0 para esse projeto (ajustáveis caso sejam necessárias variáveis analógicas ou parâmetros adicionais).

Figura 5.3: Configuração do dispositivo Factory I/O como Generic Modbus TCP Device no OpenPLC



Fonte: Própria (2025)..

Uma vez concluída essa definição, basta clicar em *Start PLC* para que o OpenPLC inicie o ciclo de varredura (scan cycle) e passe a ler/escrever nos registradores correspondentes. Nesse processo, o programa de controle executa periodicamente a lógica, verificando o estado dos *Discrete Inputs* e acionando as *Coils* conforme a necessidade do sistema.

5.2.4 Operação e Monitoração

Após o carregamento da lógica ST e a configuração do dispositivo Modbus (conforme descrito nas seções anteriores), o OpenPLC inicia o ciclo de varredura (*scan cycle*). Nesse processo, o PLC lê periodicamente as variáveis de entrada do *Factory I/O* (ex.: `SafetyDoor`, `StartButton`, `Picker_Holding_Box`), processa as instruções do diagrama de controle e atualiza as saídas correspondentes (por exemplo, `Conveyor_Move`, `Picker_Rotate_CW`).

a) Leitura de Entradas: Em cada varredura, o OpenPLC obtém o estado das *Discrete Inputs* e *Coils* configuradas no *Factory I/O*, garantindo a sincronização do processo simulado com a lógica de controle.

b) Execução do Programa: A lógica verifica se a porta de segurança (`SafetyDoor`) está fechada, se há caixa na esteira (`Box_Not_Waiting`) e se o braço me-

cânico (`Picker`) está segurando ou não um objeto, definindo os próximos passos (ligar a esteira, acionar o agarramento, rotacionar etc.).

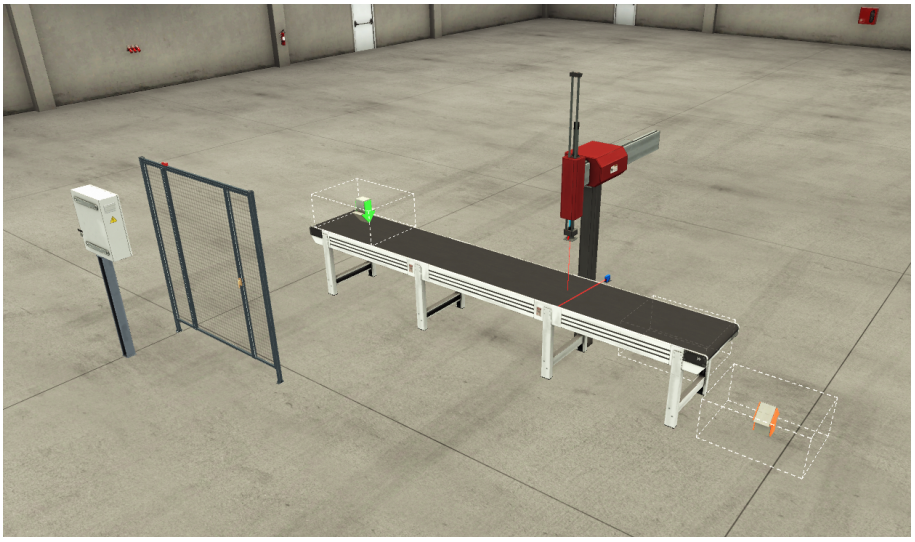
c) Escrita de Saídas: As decisões do programa (por exemplo, `Picker_Move_Down = 1`) são enviadas ao *Factory I/O* na forma de comandos Modbus, alterando o estado dos atuadores virtuais. Dessa maneira, o usuário pode observar em tempo real o movimento da esteira ou do braço mecânico no cenário simulado.

A interface *web* do OpenPLC disponibiliza a aba de *Monitoring*, na qual é possível acompanhar as variáveis declaradas no programa, como `StartButton`, `SafetyDoor`, `Picker_Moving_Z`, entre outras. Cada mudança de valor reflete imediatamente a troca de dados pela rede Modbus/TCP, permitindo ao operador diagnosticar eventuais problemas de comunicação, atrasos de resposta ou falhas na lógica de controle.

5.3 Factory I/O como Slave Modbus/TCP

O Factory I/O é um software de simulação industrial que permite recriar, em ambiente virtual, diversos elementos típicos de uma planta fabril, como esteiras, portas de segurança, sensores e braços mecânicos. Neste trabalho, ele funciona como *Modbus/TCP Slave*, recebendo comandos do OpenPLC (*Master*) e enviando de volta leituras dos dispositivos simulados. A Figura 5.4 mostra o cenário principal desenvolvido para este laboratório.

Figura 5.4: Cenário principal no Factory I/O, com porta de segurança, esteira transportadora e braço mecânico.



Fonte: Própria (2025)..

5.3.1 Elementos Simulados

O ambiente virtual criado no Factory I/O traz componentes essenciais para testes de controle e segurança:

a) Painel de Controle (Botoeira Start/Stop): Simula o acionamento de um botão físico para iniciar ou parar o processo.

b) Porta de Segurança (Safety Door): Deve permanecer fechada para liberar o funcionamento da linha, reproduzindo a lógica de *interlocks* comuns em plantas reais.

c) Esteira Transportadora (6 m): Desloca materiais de um ponto a outro, possibilitando testar sequências de produção e detecção de objetos.

d) Braço Mecânico (Two-Axis Pick & Place): Atua na movimentação e manipulação de itens, com ações de *grab*, rotação e movimentações verticais.

Cada dispositivo é mapeado internamente para *Coils* (bobinas) ou *Discrete Inputs* do protocolo Modbus, permitindo que o OpenPLC leia estados (ex.: porta aberta/fechada) e emita comandos (ex.: ligar a esteira).

5.3.2 Configuração do Servidor Modbus

Para habilitar a comunicação com o OpenPLC, o Factory I/O foi configurado como Servidor *Modbus/TCP*, conforme ilustrado na Figura 5.5:

a) Host (IP): 192.168.30.25 (estático, conforme a Tabela 5.1).

b) Port (502): Padrão para Modbus/TCP.

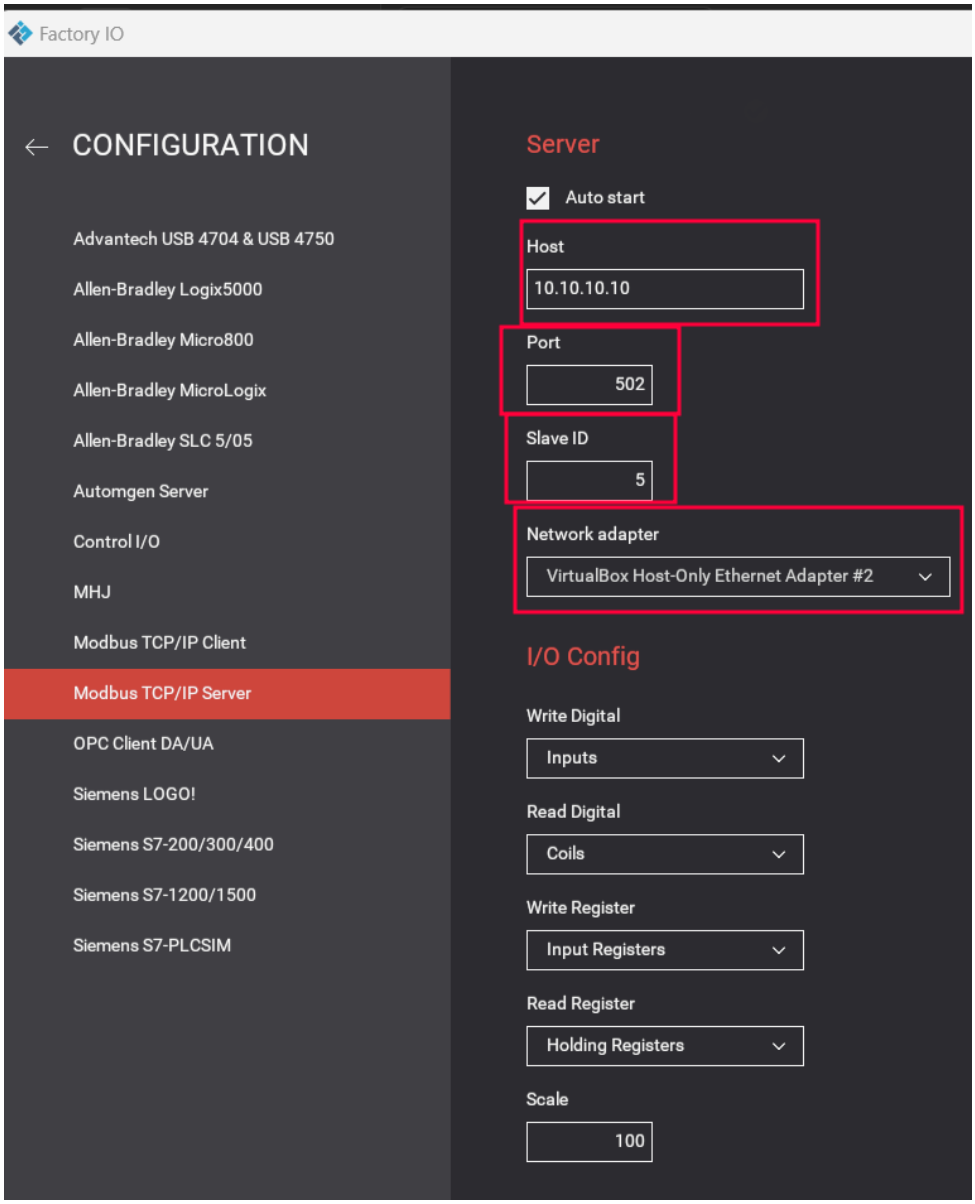
c) Slave ID (5): Identificador do dispositivo *slave*.

d) *Network Adapter*: interface da VM conectada a *bridge vmbr30* no Proxmox, para assegurar comunicação com as VMs isoladas.

e) I/O Config: Associa cada tipo de registro (*Write Digital*, *Read Digital*, *Write Register*, *Read Register*) aos pontos de E/S do cenário.

A opção *Auto start* faz com que o servidor Modbus seja iniciado automaticamente ao carregar o cenário, possibilitando a troca de dados imediata com o OpenPLC.

Figura 5.5: Tela de configuração do Modbus TCP/IP Server no Factory I/O



Fonte: Própria (2025)..

5.3.3 Mapeamento de Inputs e Coils

Após definir as configurações principais, o Factory I/O gera uma lista que relaciona cada dispositivo físico (botoeira, sensor, atuador) aos respectivos registradores Modbus. A Figura 5.6 exibe o mapeamento adotado, em que:

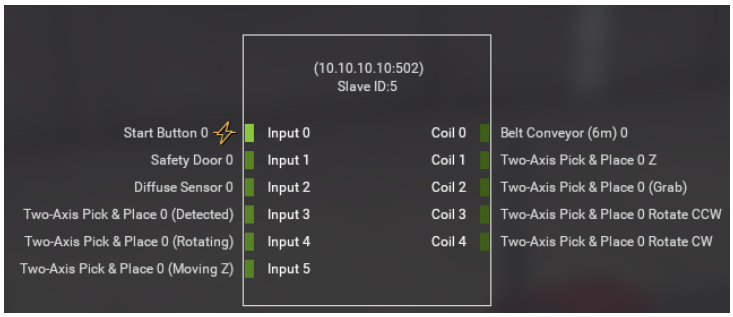
a) Start Button 0 e Safety Door 0 aparecem como *Inputs* 0 e 1, permitindo que o PLC identifique se o botão foi pressionado ou se a porta está aberta/fechada.

b) Diffuse Sensor 0 e Two-Axis Pick & Place 0 (Detected) também são lidos como *Inputs*, fornecendo ao PLC informação sobre detecção de objetos ou

posicionamento do braço.

c) Belt Conveyor (6m) 0 e Two-Axis Pick & Place (Grab, Rotate, Move) aparecem como *Coils*, sendo acionados pelo OpenPLC para ligar a esteira, girar o braço ou efetuar movimentações específicas.

Figura 5.6: Mapeamento dos *Inputs* e *Coils* no Factory I/O (*Slave ID: 5*)



Fonte: Própria (2025)..

Esse mapeamento assegura que cada elemento do cenário virtual seja acessível via registro Modbus, tornando o processo de leitura/escrita pelo PLC imediato e transparente.

5.3.4 Operação no Cenário Simulado

Com o servidor Modbus ativado, o Factory I/O e o OpenPLC operam em conjunto:

a) Leitura de Sensores: O OpenPLC consulta periodicamente as *Inputs* (ex.: Start Button, Safety Door) para verificar as condições de operação.

b) Decisão de Controle: Consoante à lógica ST ou outra linguagem IEC 61131-3, o PLC determina se a esteira ou o braço mecânico devem ser acionados.

c) Escrita em Atuadores: Os comandos são direcionados às *Coils* correspondentes (por exemplo, `Coil 2` para iniciar a esteira), resultando na execução das ações no cenário virtual.

Assim, a dinâmica de um ambiente industrial é simulada em tempo real, possibilitando a verificação de sequências de produção, condições de segurança (porta fechada) e eventos como detecção de objetos na esteira.

5.3.5 Benefícios para a Simulação

O uso do Factory I/O no ambiente experimental proporciona uma série de benefícios significativos. Primeiramente, o ambiente permite a simulação de situações industriais complexas com elevado realismo, possibilitando a reprodução de condições adversas, como falhas de sensores ou desvios operacionais, sem expor equipamentos reais ou colocar operadores em risco. Essa abordagem facilita a realização de testes e ajustes na lógica de controle de forma rápida e segura. Ademais, a integração simples com PLCs, por meio do suporte nativo a Modbus/TCP, simplifica a configuração e o mapeamento das variáveis, tornando o ambiente acessível para avaliações contínuas. Por fim, a criação de um ambiente isolado para testes de segurança permite a análise detalhada das vulnerabilidades e a validação das contramedidas propostas, garantindo que a implementação do framework não comprometa a operação dos sistemas industriais.

Dessa forma, o Factory I/O atende aos requisitos de realismo, flexibilidade e segurança, reforçando a adequação do ambiente experimental para a análise de vulnerabilidades e a validação do framework proposto.

5.4 Comunicação Modbus/TCP: Ciclo de Mensagens e Funções

Nesta seção, descreve-se como ocorre a troca de dados entre o OpenPLC (atuando como *Modbus Master*) e o Factory I/O (operando como *Modbus Slave*). Apesar de já terem sido apresentadas as configurações individuais desses dispositivos, aqui o foco recai sobre o ciclo de mensagens e as funções Modbus empregadas, evidenciando o fluxo contínuo de leitura/escrita entre ambos.

5.4.1 Protocolo Modbus/TCP e Funções Utilizadas

O Modbus/TCP segue uma arquitetura de requisição–resposta, na qual o *Master* (OpenPLC) envia comandos e o *Slave* (Factory I/O) retorna dados ou confirmações de escrita. Entre as funções mais relevantes no cenário do laboratório, destacam-se:

a) Função 1 (Read Coils): Permite ler o estado de saídas digitais (ex.: `Coils` 0–4) do Slave.

b) Função 2 (Read Discrete Inputs): Fornece o estado de entradas digitais (ex.: `Inputs` 0–5), representando sensores e botoeiras.

c) Função 15 (*Write Multiple Coils*): Altera os valores de saídas digitais, possibilitando o acionamento de atuadores como a esteira (`Conveyor_Move`) ou o braço mecânico (`Picker_Rotate_CW`).

Essas funções garantem que o PLC possa monitorar em tempo real as variáveis do Factory I/O e enviar comandos de forma pontual, mantendo a coerência com a lógica implementada no OpenPLC.

5.4.2 Fluxo de Mensagens e Ciclo de Varredura

Uma vez estabelecida a conexão (porta 502 no endereço 192.168.30.25), o OpenPLC inicia o *scan cycle* (ciclo de varredura), no qual:

a) Leitura: O *Master* envia requisições de leitura (`Read Coils` ou `Read Discrete Inputs`) para obter o estado atual dos sensores e atuadores. O *Slave* responde com os valores dos bits associados.

b) Processamento da Lógica: O OpenPLC interpreta esses valores conforme a lógica (por exemplo, verificando se a porta de segurança está fechada) e decide as próximas ações.

c) Escrita: Caso seja necessário acionar um atuador, o *Master* envia uma instrução *Write Multiple Coils* (Função 15), modificando o estado de saídas digitais no *Slave* (ex.: ligando a esteira).

d) Resposta e Atualização: O *Slave* confirma a escrita ou retorna possíveis erros. Em seguida, o ciclo se reinicia, tipicamente a cada 100 ms (ou conforme configurado).

Esse fluxo é ininterrupto enquanto o sistema se encontra em operação, permitindo ao OpenPLC reagir imediatamente a eventos no Factory I/O, como a chegada de uma caixa ao final da esteira ou a abertura inesperada da porta de segurança.

5.4.3 Exemplo de Troca de Mensagens

Para ilustrar a comunicação, a Tabela 5.2 apresenta um exemplo simplificado de requisições e respostas Modbus, capturadas durante a execução do teste:

Tabela 5.2: Exemplo de troca de mensagens Modbus entre OpenPLC (Master) e Factory I/O (Slave).

Momento	Operação	Função	Descrição
1	Master $\rightarrow$ Slave	0x01 (Read Coils)	Leitura do estado das saídas (0–4)
2	Slave $\rightarrow$ Master	0x01 (resposta)	Retorno do mapa de bits (ex.: 0b00110)
3	Master $\rightarrow$ Slave	0x15 (Write Multiple Coils)	Ativação da saída `Conveyor_Move` (Coil 2)
4	Slave $\rightarrow$ Master	0x15 (resposta)	Confirmação da escrita bem-sucedida

Fonte: Adaptada de http://www.modbus.org..

Nessa sequência, o OpenPLC primeiro consulta o estado das *Coils* do Factory I/O (passo 1–2) e, ao constatar a necessidade de ligar a esteira, envia um comando de escrita (`Write Multiple Coils`) para a Coil 2 (passo 3). O Factory I/O atualiza o atuador e retorna uma resposta confirmando a operação (passo 4). Esse padrão se repete durante todo o ciclo de controle, gerando a dinâmica desejada no ambiente simulado.

5.5 Construção do Conjunto de Dados

Esta seção descreve como o conjunto de dados de tráfego Modbus/TCP foi criado a partir das capturas no ambiente experimental. O objetivo foi transformar arquivos já processados em JSON (lista ou JSON Lines) em um CSV estruturado próprio para experimentos de aprendizado de máquina.

Cada linha do CSV representa um pacote de rede. O script de conversão "achata" somente as camadas Ether, IP e TCP, além de quaisquer campos cujo nome comece com Modbus*. Assim, o *dataset* reúne informações de enlace, rede, transporte e, quando presente, detalhes do protocolo Modbus (requisições e respostas). O arquivo final utiliza o separador ; e termina com uma coluna chamada *Classification*, onde ficam os rótulos (por exemplo, `NORMAL`, `DDOS`, `UNIT_ENUM`). Totalizando 49 colunas.

5.5.1 Origem dos dados e captura

As comunicações foram geradas no cenário OpenPLC e Factory I/O descrito anteriormente. A coleta foi passiva, por meio de *port mirroring*, evitando interferência no funcionamento da célula simulada. Os arquivos `.pcap` resultantes foram convertidos para JSON e, em seguida, processados pelo script Python que consolida e padroniza os campos antes de gravar o CSV. Os cenários de ataque foram gerados a partir de um host já conectado ao segmento OT, permitindo rotulação supervisionada baseada no tipo de atividade maliciosa executada durante as janelas de captura.

5.5.2 Taxonomia de rótulos

Os rótulos adotados seguem a seguinte taxonomia:

a) `NORMAL`: tráfego legítimo de operação.

b) `FAKE_REG`: *Fake Register Injection* (inserção de registros falsos).

c) `FUNC_TAMPER`: *Invalid Function Code Injection* (função Modbus inválida).

d) `UNIT_ENUM`: `unit_id` não utilizado na comunicação dos PLC.

e) `DDOS`: negação de serviço Modbus (inundação).

f) `MASQ`: *masquerade* criação de pacotes semelhantes aos legítimos com *payload* divergente. Para rotulagem do cenário MASQ, adotou-se, por convenção experimental deste trabalho, reference_number = 2 nas requisições Modbus como critério de marcação dos pacotes de masquerade. Essa regra foi utilizada exclusivamente para este conjunto de dados e pode ser ajustada em cenários mais gerais.

5.5.3 Limpeza, normalização e validação supervisionada dos rótulos

Após a etapa de rotulação por cenário e a consolidação dos arquivos em um único *dataset*, foi realizada uma etapa final de limpeza e normalização com o objetivo de reduzir ruídos de captura e garantir consistência entre os rótulos e o comportamento esperado do tráfego Modbus/TCP. Essa etapa foi supervisionada e baseada no conhecimento do cenário experimental (topologia, papéis dos nós e janelas de execução dos ataques), combinando inspeção manual e validação automatizada.

Um ponto crítico observado foi a presença de pacotes associados ao estabelecimento de conexões TCP (*three-way handshake*), especialmente segmentos com flags SYN, SYN/ACK e ACK, que não carregam conteúdo do protocolo Modbus/TCP e não representam, por si só, a ação maliciosa definida na taxonomia de ataques deste trabalho. Como tais pacotes podem ocorrer tanto durante tráfego legítimo quanto durante a preparação de ferramentas de ataque, optou-se por normalizá-los com o rótulo `NORMAL`, evitando que a etapa de aprendizado incorporasse sinais não relacionados à semântica do protocolo industrial.

A normalização foi implementada por um script *Python* executado após a união do conjunto rotulado, aplicando regras determinísticas para: (i) identificar pacotes sem campos *Modbus** e com flags TCP compatíveis com estabelecimento de sessão; (ii) manter somente os campos relevantes para caracterização do tráfego Modbus/TCP (Ether/IP/TCP e campos *Modbus**); e (iii) garantir consistência do rótulo final com base no comportamento observado e na janela de execução do cenário. Essa abordagem reduz a influência de artefatos do transporte (TCP) na caracterização das classes de ataque.

Para assegurar a veracidade e a rastreabilidade das decisões de limpeza/-normalização, foram utilizados mecanismos complementares de validação. Primeiramente, filtros no Wireshark foram empregados para confirmar a presença e a distribuição de padrões específicos (por exemplo, flags TCP do *handshake* e presença/ausência de campos Modbus/TCP) nas janelas temporais correspondentes aos ataques. Em paralelo, consultas e verificações no *dataset* final (CSV) confirmaram que a aplicação das regras não alterou a proporção esperada dos cenários e preservou os pacotes que efetivamente continham operações Modbus associadas às classes `FAKE_REG`,

FUNC_TAMPER, UNIT_ENUM, DDOS e MASQ.

Como resultado, a etapa final de limpeza e normalização aumentou a consistência dos dados rotulados e reforçou a validade do conjunto para os experimentos subsequentes, mantendo o foco do aprendizado nos elementos semânticos do protocolo Modbus/TCP, em vez de artefatos de sessão e estabelecimento de conexão.

5.5.4 Resumo quantitativo (conjuntos de treino e validação)

O material coletado foi dividido em dois conjuntos distintos: um conjunto de treino destinado ao ajuste dos modelos e outro de validação reservado para a avaliação independente do desempenho. A separação foi realizada de modo a manter a representatividade dos cenários experimentais em ambos os conjuntos. Além disso, no conjunto de validação, os ataques foram gerados a partir de um endereço IP diferente do utilizado no conjunto de treino, o que amplia a capacidade de generalização e reduz o risco de sobreajuste aos padrões específicos de origem do tráfego.

A Tabela 5.3 apresenta a distribuição absoluta e relativa de rótulos nos dois conjuntos.

Tabela 5.3: Contagem e percentual de amostras por rótulo nos conjuntos de treino e validação

Rótulo	Treino		Validação	
	Qtd.	% do treino	Qtd.	% da validação
DDOS	194 677	5,96%	195 628	7,42%
FAKE_REG	61 545	1,88%	31 339	1,19%
FUNC_TAMPER	422 845	12,95%	436 205	16,54%
MASQ	26 582	0,81%	153 899	5,84%
NORMAL	2 558 636	78,34%	1 816 372	68,87%
UNIT_ENUM	1 621	0,05%	3 812	0,14%
Total	3 265 906	100%	2 637 255	100%

Fonte: Própria (2025)..

A distribuição de classes evidencia a predominância do tráfego normal em ambos os conjuntos, o que reflete a natureza real de ambientes industriais, onde os eventos de ataque representam uma fração reduzida do total de comunicações. A variação entre os percentuais dos dois conjuntos decorre da diferença nas amostras e do uso de fluxos distintos para geração das capturas.

O resultado final são dois arquivos, contendo tráfego legítimo e malicioso gerado sob controle experimental. Esse material serve tanto para calibração do framework FARAONIC quanto para o treino e a avaliação de modelos preditivos, ga-

rantindo reprodutibilidade e fidelidade às condições observadas em redes OT reais. Esses *datasets* encontram-se disponíveis em: `https://www.kaggle.com/datasets/fabioaraujofabres/modbustcp-cybersecurity-dataset`

5.6 Conclusão

Este capítulo detalhou a construção do ambiente experimental e as escolhas de projeto que viabilizam a implementação e a avaliação do framework proposto em um cenário próximo ao de redes OT reais. A topologia contemplou componentes essenciais de uma célula industrial: o OpenPLC como mestre Modbus/TCP, o Factory I/O como processo simulado, o *pfSense* na segmentação e filtragem, além de nós distintos para atividades defensivas e ofensivas.

No controlador, a adoção de *Structured Text* trouxe maior estabilidade e clareza à lógica de controle, com um sequenciador de estados responsável por coordenar a esteira, a garra e as etapas de descarte. Foram descritas as configurações necessárias para a comunicação com o Factory I/O, incluindo parâmetros de rede, mapeamento de entradas e saídas e funções Modbus utilizadas, bem como o ciclo de varredura que sustenta a operação contínua do sistema.

Também foi apresentada a criação do conjunto de dados, desde a captura passiva do tráfego até a conversão padronizada para CSV. Foram criados dois conjuntos distintos: treino e validação. Essa divisão preserva a representatividade das classes e reduz o risco de sobreajuste, inclusive pela utilização de um endereço IP de atacante diferente no conjunto de validação. O pipeline de processamento achatou apenas as camadas e campos relevantes (Ether, IP, TCP e Modbus*), estabeleceu uma taxonomia de rótulos e manteve rastreabilidade e consistência por meio de cabeçalhos e reclassificação controlada.

Como resultado, o laboratório passou a oferecer um espaço controlado para experimentar estratégias de defesa e de ataque, avaliar impactos e coletar métricas com confiabilidade. Entre as limitações inerentes, destacam-se a carga de tráfego restrita ao escopo do cenário, a dependência de um conjunto específico de dispositivos e a ausência de variabilidade típica de ambientes industriais heterogêneos; tais pontos orientam trabalhos futuros, como ampliar perfis de tráfego, incorporar novos protocolos industriais e distribuir a detecção entre múltiplos segmentos da rede.

Em síntese, o ambiente descrito estabelece a base técnica para a avaliação do framework nos capítulos seguintes, fornecendo infraestrutura, lógica de controle e dados rotulados suficientes para investigar desempenho, limites e oportunidades de melhoria das abordagens de monitoramento e detecção em redes OT baseadas em Modbus/TCP.

6 RESULTADOS E DISCUSSÃO

Este capítulo apresenta a avaliação experimental e os resultados obtidos com a implementação do FARAONIC, considerando suas duas trilhas independentes de detecção: uma baseada em regras e inspeção profunda de pacotes (DPI) e outra fundamentada em aprendizado de máquina (ML). Cada abordagem foi testada de forma isolada, permitindo a análise comparativa dos resultados de detecção e desempenho em tempo real. As seções a seguir detalham os procedimentos de teste, as métricas de desempenho empregadas, os resultados obtidos para cada modo de operação e a análise crítica dos achados experimentais.

6.1 Avaliação do *framework*

A fase de avaliação teve como propósito mensurar a eficácia, confiabilidade e desempenho operacional do FARAONIC na detecção de diferentes categorias de ataque em redes Modbus/TCP. Para isso, foram definidos cenários de teste representativos, executados sobre o ambiente experimental, abrangendo situações de tráfego legítimo e anômalo.

Os experimentos buscaram responder a três questões centrais:

a) Qual a capacidade do FARAONIC em identificar ataques específicos do domínio OT, considerando os modos DPI e ML separadamente?

b) Como as abordagens de DPI e de ML se comparam em relação à redução de falsos negativos e falsos positivos?

c) Cada trilha é capaz de operar em tempo real sem comprometer a estabilidade da comunicação Modbus?

As subseções seguintes apresentam os procedimentos adotados, as métricas utilizadas e os resultados experimentais obtidos para cada modo de detecção.

6.1.1 Procedimentos de Teste

A validação experimental considerou cinco famílias de ataque representativas em redes OT baseadas em Modbus/TCP. Cada cenário foi projetado para avaliar aspectos distintos da deteção (ex.: assinatura, comportamento temporal, saturação de recursos e camuflagem de identidade). Os ataques utilizados foram:

a) FAKE_REG — *Fake Register Injection*: injeção de registros inexistentes (registros falsos) visando provocar leituras/escritas inesperadas nos PLC.

b) FUNC_TAMPER — *Invalid Function Code Injection*: envio de códigos de função Modbus inválidos ou raramente usados para explorar tratamento irregular pelo servidor.

c) UNIT_ENUM — *Unit ID Enumeration*: enumeração sequencial de *unit_id* para descoberta de dispositivos ativos.

d) DDOS — *Modbus DoS*: inundação de requisições Modbus para exaurir recursos do alvo.

e) MASQ — *Masquerade/Impersonation*: tentativa de mascaramento de origem/identidade (no pipeline de pós-processamento utiliza-se *func_code==2* como gatilho para heurísticas de mascaramento).

Para garantir reprodutibilidade e estabilidade dos resultados, cada cenário foi executado seguindo um protocolo experimental padronizado. Foram realizados cinco ataques distintos, todos seguindo a mesma lógica experimental: orquestração do atacante, registro de *ground-truth*, captura em *pcap* e análise pós-processamento, conforme descrito a seguir:

a) Ataque 1 — FAKE_REG (*Fake Register Injection*): geração de comandos Modbus contendo endereços ou registradores inválidos ou fora do escopo esperado, orquestrada por scripts Python (Scapy) parametrizáveis quanto a carga útil, taxa e *unit_id*. O objetivo foi avaliar a detecção de injeção de registradores inexistentes.

b) Ataque 2 — FUNC_TAMPER (*Invalid Function Code Injection*): envio de códigos de função Modbus inválidos ou raramente utilizados para testar o tratamento irregular pelo servidor. Implementado com scripts Python que controlavam a sequência e o tempo de envio.

c) Ataque 3 — UNIT_ENUM (*Unit ID Enumeration*): varredura sequencial de *unit_id* para descoberta de dispositivos ativos, conduzida via *msfconsole* (módulos de varredura) para simular técnicas de reconhecimento típicas em redes OT.

d) Ataque 4 — DDOS (*Modbus DoS*): inundação de requisições Modbus utilizando *hping3*, que permite controlar precisamente o intervalo entre pacotes por meio do parâmetro *-i u*. Este experimento avaliou a robustez do detector sob diferentes taxas de transmissão.

e) Ataque 5 — MASQ (*Masquerade / Impersonation*): tentativas de mascaramento de origem e identidade, gerando tráfego com variações em campos

de origem e padrões de função para acionar heurísticas de mascaramento após o processamento.

Os seguintes padrões experimentais foram aplicados a todos os ataques:

a) Execuções: cada ataque foi repetido cinco vezes de forma independente, com reinicialização dos contadores e limpeza de logs entre execuções.

b) Duração: janelas mínimas de 120 s para os cenários funcionais (injeção de registros falsos, injeção de funções falsas, enumeração de Unit ID e mascaramento de pacotes). Para o DDOS, a duração foi ajustada conforme a taxa de envio configurada.

c) Taxas e variabilidade: os geradores de tráfego controlaram taxa e variabilidade (pacotes por segundo configuráveis). As variantes de DoS testaram intervalos entre envios conforme detalhado adiante.

d) Ferramentas de geração: uso combinado de scripts Python (Scapy), *msfconsole* para varredura e *hping3* para inundações controladas. Todos os scripts registraram parâmetros como taxa, origem, destino e carga útil.

e) Captura e sincronização: o tráfego foi capturado em formato *pcap* pelo sensor de referência, com sincronização de relógios via NTP. O *ground-truth* foi gravado em arquivos CSV/JSON contendo carimbos de tempo de alta resolução.

No experimento de DoS foi utilizado o parâmetro *-i u* do *hping3* para controlar o intervalo entre pacotes em microsegundos (*µs*). A relação aproximada entre o intervalo e a taxa de envio é dada por:

$$\text{pps (pacotes/s)} \approx \frac{10^6}{u\ (\mu s)}$$

Por exemplo, um valor de $u = 400\ \mu s$ corresponde a aproximadamente $2{,}500$ pacotes por segundo. Cada execução manteve uma janela de duração fixa, de modo que o número total de pacotes transmitidos por execução dependeu diretamente da taxa configurada. Observou-se que a porcentagem de detecção aumenta conforme o valor de u cresce, ou seja, quando o intervalo entre pacotes é maior e a taxa de envio diminui. Em taxas muito altas (valores pequenos de u), parte dos pacotes deixa de ser contabilizada pelo detector, possivelmente em razão de limitações de processamento ou de entrada e saída. Nos valores $u = 900\ \mu s$ e $u = 1000\ \mu s$ foram obtidas análises de

100% em relação aos pacotes transmitidos reportados pelo *hping3*. A Tabela 6.1 apresenta os resultados completos obtidos nas varreduras realizadas com o parâmetro *-i u* variando de 400 a 1000, em incrementos de 100 microsegundos.

Tabela 6.1: Detecção do FARAONIC em função do intervalo *-i u* do *hping3*.

Execução	u (*µs*)	Pacotes transmitidos	Pacotes analisados	Detecção (%)
01	400	46 761	28 661	61,29
02	500	37 939	28 314	74,63
03	600	31 828	27 717	87,08
04	700	27 472	26 654	97,02
05	800	24 083	23 852	99,04
06	900	21 414	21 414	100,00
07	1000	19 309	19 309	100,00

Fonte: Própria (2025)..

Nas execuções em que o número de pacotes detectados permaneceu entre 27 000 e 28 000, o subsistema de captura e inspeção demonstrou estabilidade, indicando que o FARAONIC é capaz de processar aproximadamente 28 000 pacotes por janela de teste sem perda observável de eventos relevantes.

6.1.2 Métricas de Desempenho

A análise quantitativa adotou as métricas clássicas aplicadas a sistemas de detecção de intrusões:

$$\text{Precisão} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad \text{F1-score} = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}}$$

onde:

a) TP — ataques corretamente detectados;

b) FP — alarmes incorretos (falsos positivos);

c) FN — ataques não detectados (falsos negativos).

Essas métricas permitem quantificar a confiabilidade do mecanismo de detecção (precisão), sua capacidade de cobertura (revocação) e o equilíbrio entre ambas (F1-score).

6.2 Análise dos Resultados de DPI

Os resultados apresentados na Tabela 6.2 evidenciam que o FARAONIC apresentou desempenho consistente e elevado nas detecções baseadas em inspeção profunda de pacotes. O mecanismo mostrou-se capaz de identificar, em tempo real, diferentes padrões de ataque com elevada precisão, mantendo estabilidade mesmo sob condições de tráfego intensas.

Tabela 6.2: Resultados dos testes de detecção baseados em DPI, por ataque (abreviado).

Ataque	Precisão (%)	Revocação (%)	F1-score (%)
Atq1 (FAKE_REG)	95,6	100,0	97,7
Atq2 (FUNC_TAMPER)	92,0	92,0	92,0
Atq3 (UNIT_ENUM)	100,0	100,0	100,0
Atq4 (DDOS)	100,0	100,0	100,0
Atq5 (MASQ)	100,00	63,02	77,38

Fonte: Própria (2025)..

Nos cenários de *Fake Register Injection* (Atq1) e *Unit ID Enumeration* (Atq3), o sistema alcançou desempenho praticamente ideal, com *F1-score* de 97,7% e 100%, respectivamente. Esses resultados confirmam a robustez das regras implementadas para reconhecer anomalias estruturais no protocolo Modbus/TCP, especialmente nos campos relacionados a registradores e identificadores de unidade. A resposta imediata do detector a modificações nos campos Modbus demonstra a eficácia da abordagem determinística em capturar comportamentos fora do padrão de comunicação esperado.

O ataque *Invalid Function Code Injection* (Atq2) apresentou desempenho ligeiramente inferior (92%), refletindo a natureza ambígua do tráfego gerado por comandos raros, porém válidos, que podem se assemelhar a atividades maliciosas. Esse resultado indica a necessidade de ajustes finos nas regras de inspeção, a fim de reduzir falsos positivos em contextos industriais onde há uso legítimo de funções pouco comuns.

Nos experimentos de *Denial of Service* (Atq4), observou-se que o FARAONIC manteve 100% de precisão e revocação para os pacotes efetivamente processados, demonstrando que o mecanismo classifica corretamente todos os eventos capturados. Contudo, o sistema não conseguiu processar a totalidade dos pacotes transmitidos em taxas muito elevadas. Assim, o FARAONIC apresenta classificação excelente sobre o tráfego que consegue analisar, mas sua capacidade de captura prática estabiliza em torno de 27 000 a 28 000 pacotes por janela de observação. Esse comportamento é

típico de sistemas que operam em tempo real e evidencia a importância de ajustes de taxa e paralelização de sensores em cenários de saturação de rede.

O ataque de *Masquerade* (Atq5) apresentou taxa de revocação de 63,02% e *F1-score* de 77,38%, considerando ausência de falsos positivos (FP = 0). Embora o desempenho seja inferior aos demais, o resultado é coerente com a natureza do ataque, que busca disfarçar a origem dos pacotes e se confundir com tráfego legítimo. Esse tipo de comportamento demonstra a necessidade de mecanismos complementares, como análise temporal ou correlação de eventos, capazes de identificar desvios sutis no padrão de comunicação.

De modo geral, o conjunto de experimentos confirma que o módulo de DPI do FARAONIC é altamente eficaz na detecção de anomalias estruturais e comportamentais de baixa complexidade, mantendo baixa taxa de falsos positivos e alta precisão sobre o tráfego efetivamente processado. Entretanto, também evidencia que o volume de pacotes capturados limita o desempenho sob condições extremas de carga, e que ataques distribuídos ou mascarados ainda representam desafios relevantes para abordagens puramente determinísticas.

6.3 Treinamento e Avaliação com Aprendizado de Máquina

O módulo de aprendizado de máquina foi projetado para complementar o DPI ao prover detecção comportamental e adaptativa. Para isso, utilizaram-se amostras rotuladas nas classes NORMAL, FAKE_REG, FUNC_TAMPER, UNIT_ENUM, DDOS e MASQ.

6.3.1 Configuração do treinamento e validação

Com o objetivo de avaliar a capacidade de generalização do modelo em condições realistas, adotou-se um protocolo com conjuntos distintos para as fases de treinamento e validação. O treinamento foi conduzido sobre um dataset dedicado, enquanto a validação ocorreu em um segundo conjunto, coletado e rotulado de forma independente. Essa abordagem reduz viés otimista e impede que padrões específicos do conjunto de treino influenciem artificialmente o desempenho reportado.

Os modelos empregados foram *DecisionTreeClassifier* e *RandomForestClassifier*, tendo como variável-alvo *Classification*. Para evitar sobreajuste a identificadores de origem, removeram-se variáveis diretamente utilizadas no processo de rotulagem ou sem relevância para o comportamento Modbus/TCP.

6.3.2 Preparo dos dados e mitigação de *data leakage*

O dataset inicial, resultante da conversão das capturas de tráfego em formato JSON, continha 49 variáveis distribuídas entre as camadas Ethernet, IP, TCP e Mod-

bus/TCP, conforme mostrado na Tabela 6.3. Essa estrutura serviu de base para a seleção e exclusão de atributos, com foco em eliminar redundâncias, constantes e campos suscetíveis a *data leakage*, isto é, atributos que revelam informações externas ao comportamento real do protocolo e que poderiam induzir o modelo a aprender correlações artificiais.

Tabela 6.3: Estrutura original do dataset de tráfego Modbus/TCP antes do pré-processamento.

#	Campo	Descrição resumida
1	timestamp	Tempo de captura do pacote
2–4	Ether_*	Camada de enlace (endereços MAC, tipo EtherType)
5–17	IP_*	Cabeçalho IPv4 (TOS, TTL, endereços, checksum, opções)
18–28	TCP_*	Cabeçalho TCP (portas, sequência, ACK, flags, janela, checksum)
29–33	ModbusTCPRequest_*	Requisição Modbus (transação, unidade, função)
34–38	ModbusTCPResponse_*	Resposta Modbus
39–42	ModbusReadDiscreteInputs_*	Operações de leitura de entradas discretas
43–48	ModbusWriteMultipleCoils_*	Operações de escrita múltipla em coils
49	Classification	Rótulo da classe (NORMAL ou tipo de ataque)

Fonte: Própria (2025)..

Durante o pré-processamento, foram identificadas variáveis que poderiam comprometer a imparcialidade do aprendizado. Endereços de rede e identificadores, como *IP_src*, *IP_dst* e os *reference numbers* Modbus, foram utilizados na rotulagem dos ataques e, por isso, excluídos do treinamento. Também foram removidos campos estruturais ou de integridade, como *IP_version*, *IP_proto*, *TCP_reserved*, *TCP_urgptr*, *IP_chksum* e *TCP_chksum*, que não contribuem para a caracterização comportamental do tráfego. Da mesma forma, variáveis voláteis de sessão, como *TCP_seq* e *TCP_ack*, foram descartadas por representarem apenas a dinâmica interna das conexões TCP.

Essas exclusões abrangeram quatro categorias principais: (i) endereços de enlace e rede; (ii) identificadores e campos invariáveis ou de controle de integridade; (iii) *reference numbers* utilizados na rotulagem; e (iv) metadados de sessão com alta volatilidade. A Tabela 6.4 apresenta de forma consolidada os campos removidos e a justificativa correspondente, evidenciando o critério de filtragem aplicado para evitar *data leakage* e assegurar que o modelo aprendesse padrões genuínos de comportamento Modbus/TCP.

Tabela 6.4: Campos excluídos do dataset na etapa de mitigação de *data leakage*.

Campo	Justificativa para exclusão
Ether_src, Ether_dst, Ether_type	Identificadores físicos fixos e invariáveis
IP_src, IP_dst	Utilizados na rotulagem dos ataques (origem/destino)
IP_version, IP_proto, IP_chksum	Constantes ou campos de integridade sem valor comportamental
TCP_sport, TCP_dport	Portas estáticas (502) ou relacionadas à origem do ataque
TCP_seq, TCP_ack	Variáveis voláteis de sessão TCP
TCP_dataofs, TCP_reserved, TCP_chksum, TCP_urgptr, TCP_options	Campos estruturais ou de controle de integridade
ModbusReadDiscreteInputsRequest_reference_number	Valor usado na rotulagem de ataques FAKE_REG e UNIT_ENUM
ModbusWriteMultipleCoilsRequest_reference_number	Valor usado na rotulagem de ataques de escrita
ModbusWriteMultipleCoilsResponse_reference_number	Campo de correlação da resposta, sem informação comportamental

Fonte: Própria (2025)..

Essa filtragem assegurou a integridade experimental e evitou que o modelo aprendesse correlações espúrias associadas à origem dos pacotes, em vez de padrões genuínos de comportamento Modbus/TCP.

Portanto, o alvo foi *Classification*, os modelos utilizados foram *DecisionTreeClassifier* e *RandomForestClassifier*, o treinamento e a validação foram realizados em datasets distintos, e a exclusão de variáveis seguiu critérios de mitigação de *data leakage* e relevância informacional. Essa configuração permitiu uma avaliação de forma mais fidedigna a capacidade de generalização dos modelos e reforçou o papel do aprendizado de máquina como complemento ao DPI na detecção de anomalias comportamentais em redes industriais.

6.3.3 Resultados dos Modelos

A avaliação dos modelos de aprendizado de máquina foi conduzida de forma incremental, por meio de cinco experimentos sucessivos. Cada experimento utilizou valores SHAP, técnica de interpretabilidade baseada nos valores de Shapley, para identificar as variáveis mais influentes no resultado do modelo, permitindo reconhecer atributos que introduziam viés ou dependência indevida. A partir dessa análise, decidiu-se pela remoção gradual de determinados campos antes da próxima rodada de treinamento. Essa estratégia possibilitou isolar o conjunto mais adequado de variáveis, preservando somente aquelas que representavam o comportamento lógico do tráfego Modbus/TCP.

6.3.3.1 Experimento 1 — Conjunto completo após o pré-processamento

O primeiro experimento utilizou o conjunto de variáveis remanescentes após a etapa inicial de filtragem, em que foram removidos endereços físicos e lógicos, identificadores de sessão, campos constantes e *reference numbers* empregados na rotulagem (Seção 6.3). Assim, o modelo foi treinado apenas com atributos capazes de expressar o comportamento funcional do tráfego Modbus/TCP e com risco menor de *data leakage*.

Os modelos *DecisionTreeClassifier* e *RandomForestClassifier* apresentaram desempenho elevado no F1-score conforme as Tabelas 6.5 e 6.6.

Tabela 6.5: Resultados do *DecisionTreeClassifier* no Experimento 1.

Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	0.00	0.00	0.00	31 339
FUNC_TAMPER	93.57	99.50	96.44	436 205
MASQ	100.00	99.11	99.55	153 899
NORMAL	99.55	97.74	98.63	1 816 372
UNIT_ENUM	1.82	19.99	3.34	3 812
Acurácia global		97.00%		
F1-weighted		97.12%		

Fonte: Própria (2025)..

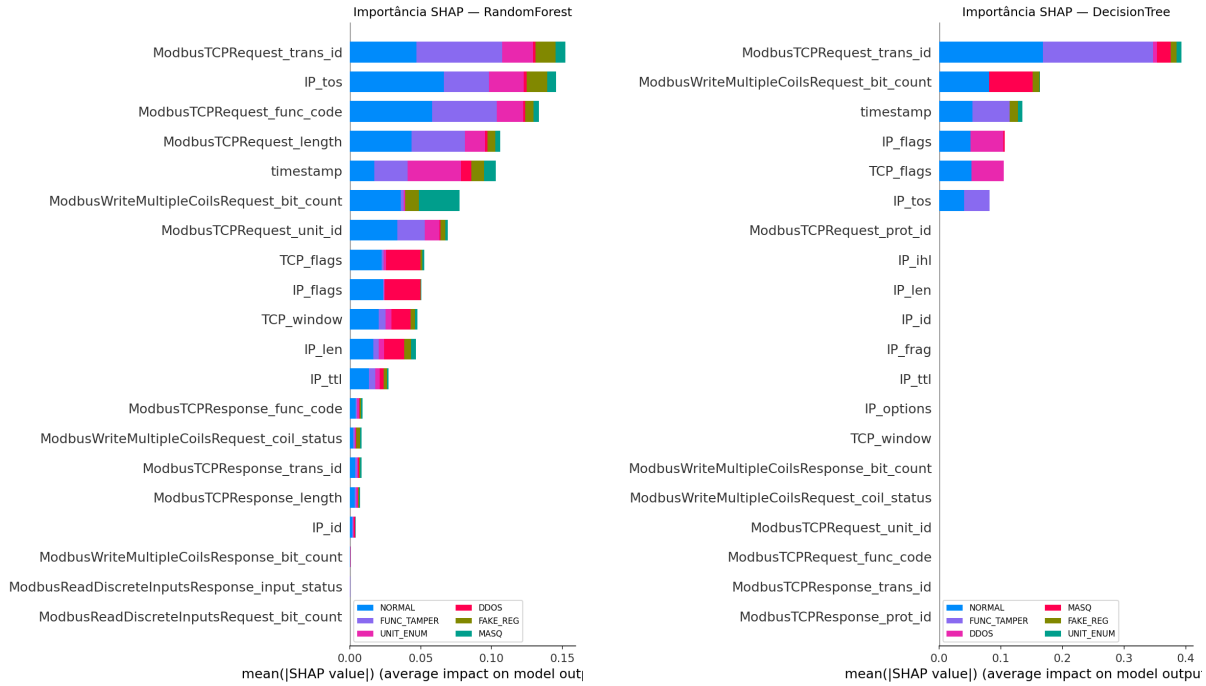
Tabela 6.6: Resultados do *RandomForestClassifier* no Experimento 1.

Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	100.00	95.24	97.56	31 339
FUNC_TAMPER	100.00	99.50	99.75	436 205
MASQ	100.00	99.93	99.97	153 899
NORMAL	99.63	100.00	99.81	1 816 372
UNIT_ENUM	100.00	19.99	33.32	3 812
Acurácia global		99.74%		
F1-weighted		99.70%		

Fonte: Própria (2025)..

Embora o desempenho global tenha sido o melhor entre todos os experimentos, a análise de interpretabilidade com SHAP revelou que o atributo *ModbusTCPRequest_trans_id* exerceu influência dominante no processo de decisão, seguido de variáveis de ordenação temporal como *timestamp*, *IP_tos* e *ModbusTCPRequest_length*. Tais variáveis representam sobretudo a sequência dos pacotes, e não o comportamento semântico do protocolo Modbus/TCP.

Figura 6.1: Importância média das variáveis no Experimento 1 segundo valores SHAP.



(a) Importância SHAP — *RandomForest*. (b) Importância SHAP — *DecisionTree*.

Fonte: Própria (2025)..

A predominância de *ModbusTCPRequest_trans_id* indica que o modelo estava aprendendo correlações de sequência, um padrão artificial decorrente da ordem de chegada dos pacotes. Para impedir que o modelo baseasse suas previsões em artefatos temporais, optou-se por removê-lo no Experimento 2, assegurando uma configuração mais imparcial e robusta.

6.3.3.2 Experimento 2 — Remoção de *ModbusTCPRequest_trans_id*

O Experimento 2 avaliou o impacto da remoção do atributo *ModbusTCPRequest_trans_id*, identificado no Experimento 1 como a variável de maior impacto nos valores SHAP e, portanto, a principal responsável pela indução de padrões artificiais relacionados à ordem de chegada dos pacotes. A exclusão desse campo buscou reduzir vieses de sequência e incentivar os modelos a se apoiarem em características realmente associadas ao comportamento Modbus/TCP.

As Tabelas 6.7 e 6.8 apresentam o desempenho dos modelos após essa modificação.

Tabela 6.7: Resultados do *DecisionTreeClassifier* no Experimento 2.

Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	0.00	0.00	0.00	31 339
FUNC_TAMPER	0.00	0.00	0.00	436 205
MASQ	83.76	100.00	91.16	153 899
NORMAL	99.63	100.00	99.82	1 816 372
UNIT_ENUM	0.18	19.99	0.35	3 812
Acurácia global		82.16%		
F1-weighted		81.48%		

Fonte: Própria (2025)..

Tabela 6.8: Resultados do *RandomForestClassifier* no Experimento 2.

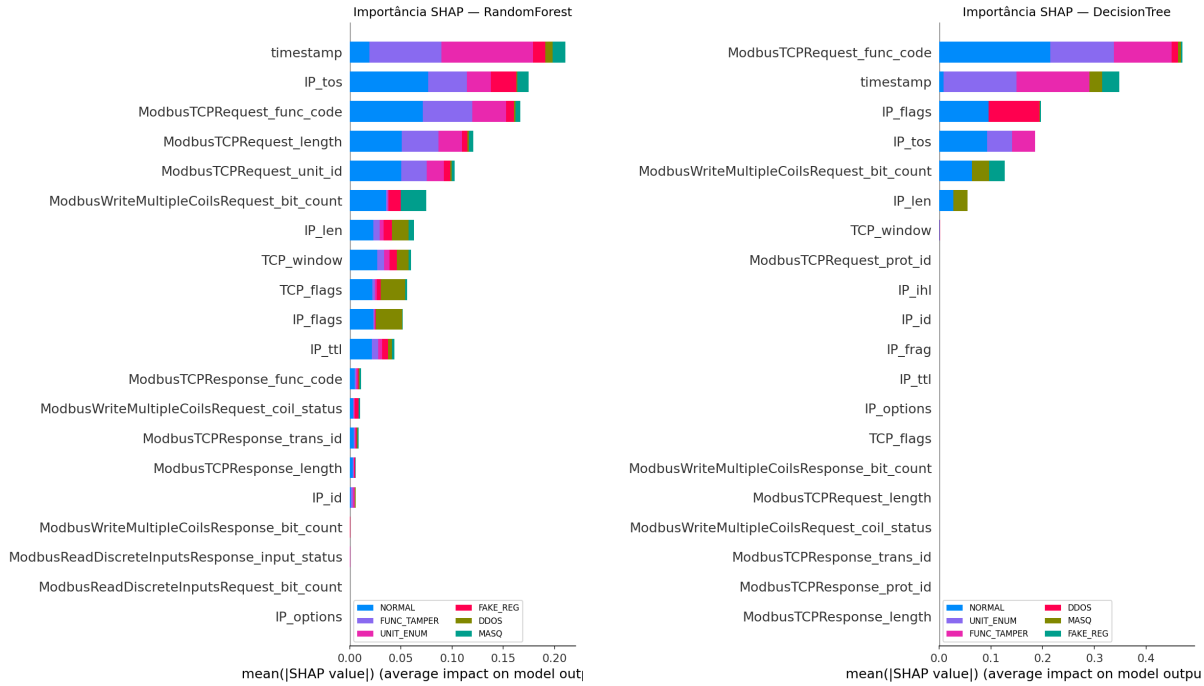
Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	100.00	95.24	97.56	31 339
FUNC_TAMPER	100.00	19.85	33.12	436 205
MASQ	100.00	100.00	100.00	153 899
NORMAL	99.63	100.00	99.82	1 816 372
UNIT_ENUM	0.22	19.99	0.43	3 812
Acurácia global		86.57%		
F1-weighted		88.64%		

Fonte: Própria (2025)..

A remoção de *ModbusTCPRequest_trans_id* produziu queda expressiva no desempenho do *DecisionTreeClassifier*, sobretudo nas classes FAKE_REG e FUNC TAMPER, que passaram a apresentar F1-score praticamente nulo. O *RandomForestClassifier*, por outro lado, manteve desempenho satisfatório em quatro classes, mas continuou apresentando dificuldades significativas nas classes FUNC_TAMPER e UNIT_ENUM.

Os gráficos SHAP das Figuras 6.2 mostram que, após a remoção do identificador de transação, outras variáveis estruturais e temporais, especialmente *timestamp*, *IP_tos* e *IP_flags*, passaram a assumir papel dominante na decisão dos modelos. Esse resultado indica que, mesmo sem o campo de transação Modbus, o modelo seguia explorando correlações relacionadas ao tempo e à dinâmica interna da captura, o que justificou a continuidade do processo de refinamento no Experimento 3.

Figura 6.2: Importância média das variáveis no Experimento 2 segundo valores SHAP.



(a) Importância SHAP — *RandomForest*. (b) Importância SHAP — *DecisionTree*.

Fonte: Própria (2025)..

A predominância de atributos temporais após a remoção de *ModbusTCPRequest_trans_id* sugeriu que ainda havia influência indevida da ordenação dos pacotes na tomada de decisão. Dessa forma, decidiu-se remover também o campo *timestamp* no Experimento 3, buscando reduzir a dependência do modelo em sinais de tempo e aproximar mais o aprendizado das características funcionais do protocolo Modbus/TCP.

6.3.3.3 Experimento 3 — Remoção de *timestamp*

O Experimento 3 avaliou os efeitos da remoção simultânea de dois atributos: *ModbusTCPRequest_trans_id* e *timestamp*. A exclusão deste último visou mitigar de forma mais profunda a dependência do modelo em sinais de tempo e correlações associadas à ordenação dos pacotes, que continuaram presentes no Experimento 2. O objetivo foi aproximar o aprendizado das características funcionais do protocolo Modbus/TCP, reduzindo interferências externas do processo de captura.

As Tabelas 6.9 e 6.10 apresentam os resultados obtidos no F1-score.

Tabela 6.9: Resultados do *DecisionTreeClassifier* no Experimento 3.

Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	100.00	95.24	97.56	31 339
FUNC_TAMPER	100.00	99.50	99.75	436 205
MASQ	100.00	100.00	100.00	153 899
NORMAL	99.63	100.00	99.82	1 816 372
UNIT_ENUM	99.61	19.91	33.19	3 812
Acurácia global		99.75%		
F1-weighted		99.71%		

Fonte: Própria (2025)..

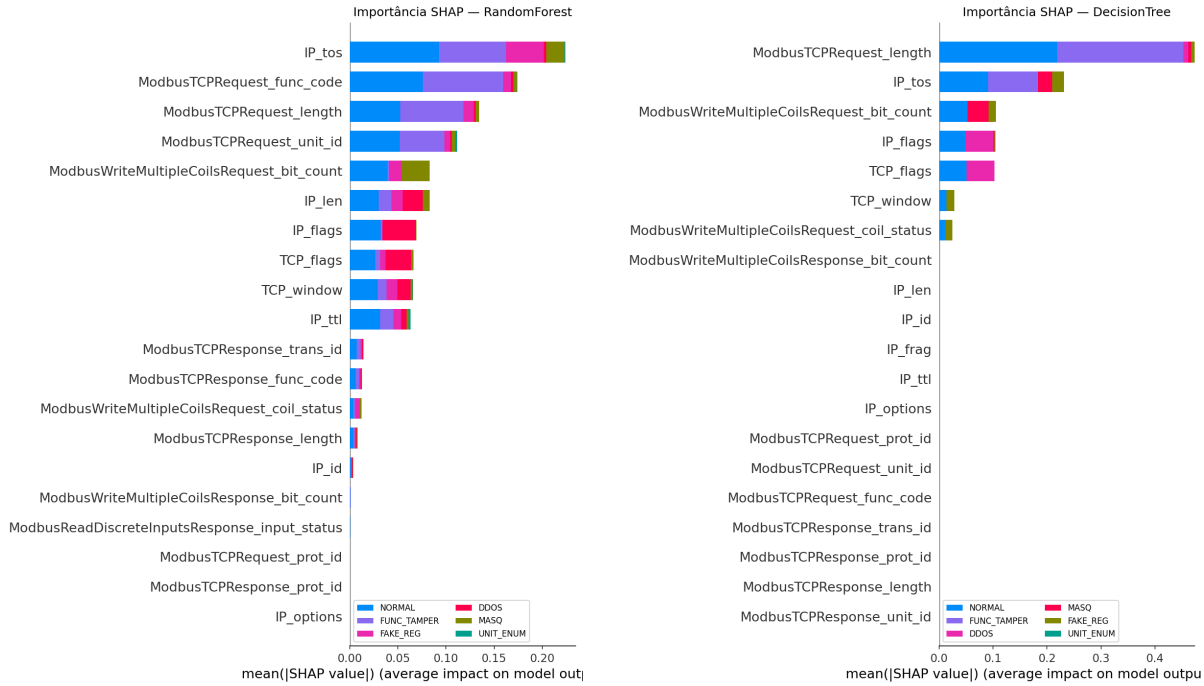
Tabela 6.10: Resultados do *RandomForestClassifier* no Experimento 3.

Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	100.00	95.24	97.56	31 339
FUNC_TAMPER	100.00	99.50	99.75	436 205
MASQ	100.00	100.00	100.00	153 899
NORMAL	99.63	100.00	99.82	1 816 372
UNIT_ENUM	99.61	19.91	33.19	3 812
Acurácia global		99.75%		
F1-weighted		99.71%		

Fonte: Própria (2025)..

A remoção de *timestamp* estabilizou o aprendizado e eliminou as distorções observadas no Experimento 2. O desempenho dos dois modelos voltou a níveis equivalentes aos do Experimento 1, porém sem dependência de variáveis temporais ou de sequência. As Figuras 6.3 mostram que, com a retirada de atributos relacionados ao tempo, a importância das variáveis voltou a se concentrar em parâmetros funcionais do protocolo, especialmente *ModbusTCPRequest_length*, *ModbusTCPRequest_func_code* e *ModbusWriteMultipleCoilsRequest_bit_count*. Trata-se de um perfil de decisão mais condizente com a lógica operacional do Modbus/TCP.

Figura 6.3: Importância média das variáveis no Experimento 3 segundo valores SHAP.



(a) Importância SHAP — *RandomForest*. (b) Importância SHAP — *DecisionTree*.

Fonte: Própria (2025)..

Com a retirada de *ModbusTCPRequest_trans_id* e *timestamp*, o modelo passou a basear suas decisões em atributos diretamente associados ao comportamento semântico do protocolo. A redução dos sinais artificiais de tempo e sequência tornou o Experimento 3 o mais equilibrado e informativamente consistente entre os realizados até esta etapa.

A análise SHAP também mostrou que, na ausência de variáveis temporais, o campo *IP_tos* passou a assumir posição de destaque mesmo não sendo diretamente relacionado à lógica do Modbus/TCP. Esse comportamento sugeriu a presença de um padrão residual dependente da forma como o sistema operacional estruturou o tráfego durante a captura.

Diante dessa observação, decidiu-se remover *IP_tos* no Experimento 4, de modo a avaliar se sua influência elevava artificialmente a capacidade preditiva do modelo e para verificar se a decisão poderia ser ainda mais alinhada às características funcionais do protocolo.

6.3.3.4 Experimento 4 — Remoção de *IP_tos*

O Experimento 4 avaliou o impacto da remoção de *IP_tos*, atributo que no Experimento 3 apresentou forte influência nos valores SHAP, apesar de não expressar diretamente comportamento funcional do protocolo Modbus/TCP. Embora *IP_tos* possa refletir características de priorização ou diferenciação de tráfego, seu uso no processo decisório do modelo sugeria novamente uma dependência indireta da estrutura de captura, e não das operações Modbus propriamente ditas. Assim, sua remoção buscou refinar ainda mais a imparcialidade do processo de classificação.

Os resultados obtidos após a retirada desse campo encontram-se nas Tabelas 6.11 e 6.12.

Tabela 6.11: Resultados do *DecisionTreeClassifier* no Experimento 4.

Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	28.38	90.55	43.22	31 339
FUNC_TAMPER	100.00	99.50	99.75	436 205
MASQ	98.26	53.45	69.24	153 899
NORMAL	99.63	100.00	99.81	1 816 372
UNIT_ENUM	99.61	19.91	33.19	3 812
Acurácia global		96.97%		
F1-weighted		97.26%		

Fonte: Própria (2025)..

Tabela 6.12: Resultados do *RandomForestClassifier* no Experimento 4.

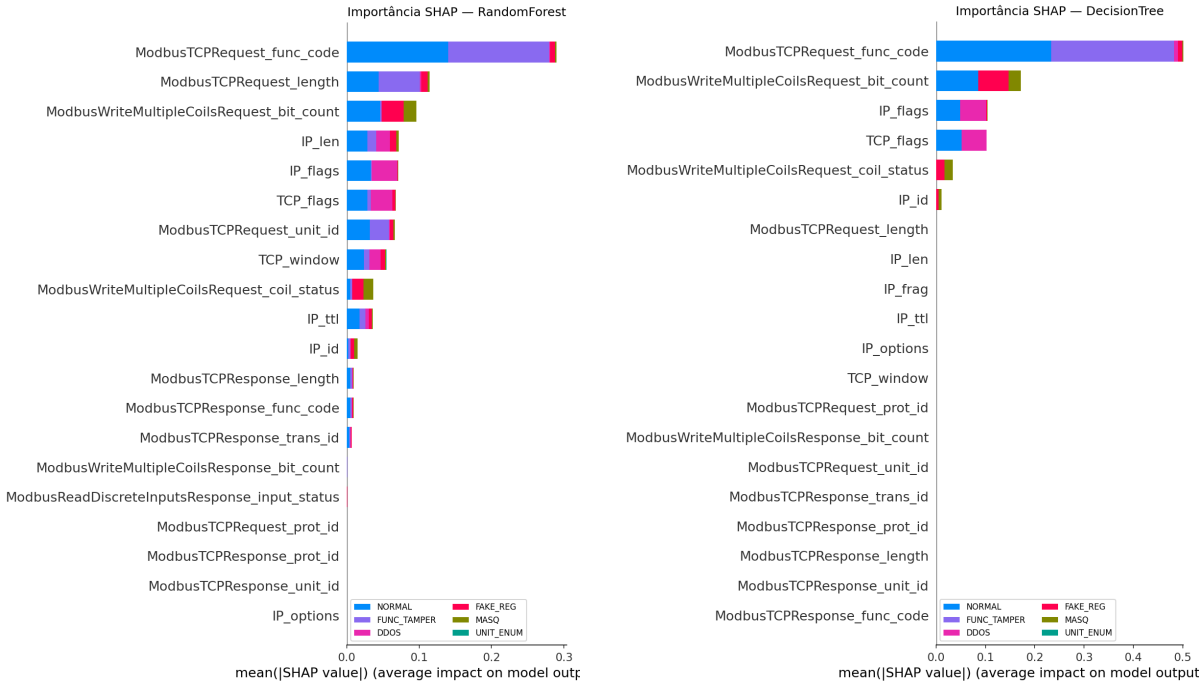
Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	28.38	90.57	43.22	31 339
FUNC_TAMPER	100.00	99.50	99.75	436 205
MASQ	98.25	53.47	69.25	153 899
NORMAL	99.63	100.00	99.82	1 816 372
UNIT_ENUM	99.61	19.91	33.19	3 812
Acurácia global		96.97%		
F1-weighted		97.27%		

Fonte: Própria (2025)..

Os valores SHAP do Experimento 4 (Figuras 6.4) mostram que, após remover *IP_tos*, os modelos passaram a se apoiar fortemente nos campos *ModbusTCPRequest func code*, *ModbusTCPRequest length* e *ModbusWriteMultipleCoilsRequest bit count*, todos diretamente relacionados à operação Modbus.

Apesar disso, observou-se que certas classes continuaram apresentando degradação, especialmente MASQ e FAKE_REG, cujas revocações oscilaram significativamente. Isso sugere que ainda havia campos estruturais com potencial para introduzir ruído, particularmente *IP_len*, que emergiu com importância elevada nos valores SHAP.

Figura 6.4: Importância média das variáveis no Experimento 4 segundo valores SHAP.



(a) Importância SHAP — *RandomForest*. (b) Importância SHAP — *DecisionTree*.

Fonte: Própria (2025)..

A predominância de *IP_len* como uma das principais variáveis após a remoção de *IP_tos* indica que campos estruturais do cabeçalho IP ainda exerciam influência considerável no processo de classificação. Por esse motivo, decidiu-se prosseguir com o Experimento 5 avaliando a remoção de *IP_len* e outros campos estruturais adicionais, buscando reduzir de forma ainda mais consistente a dependência do modelo em aspectos alheios ao comportamento Modbus/TCP.

6.3.3.5 Experimento 5 — Remoção de campos estruturais adicionais

O Experimento 5 deu continuidade ao processo de depuração iniciado nos experimentos anteriores, removendo um conjunto ampliado de atributos estruturais: *IP_len*, *TCP_flags*, *IP_flags*, *ModbusTCPRequest_length* e *timestamp*. Esses campos haviam emergido com elevada importância nos valores SHAP nos experimentos anteriores, mas sem relação direta com a semântica do protocolo Modbus/TCP. A

decisão de removê-los buscou testar se o modelo seria capaz de manter bom desempenho apoiando-se apenas em atributos funcionais do protocolo, eliminando qualquer dependência residual de aspectos estruturais do cabeçalho IP/TCP.

As Tabelas 6.13 e 6.14 apresentam o desempenho dos modelos após essa nova filtragem.

Tabela 6.13: Resultados do *DecisionTreeClassifier* no Experimento 5.

Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	28.38	90.58	43.22	31 339
FUNC_TAMPER	100.00	99.50	99.75	436 205
MASQ	98.26	53.46	69.24	153 899
NORMAL	99.63	100.00	99.82	1 816 372
UNIT_ENUM	99.61	19.91	33.19	3 812
Acurácia global		96.97%		
F1-weighted		97.27%		

Fonte: Própria (2025)..

Tabela 6.14: Resultados do *RandomForestClassifier* no Experimento 5.

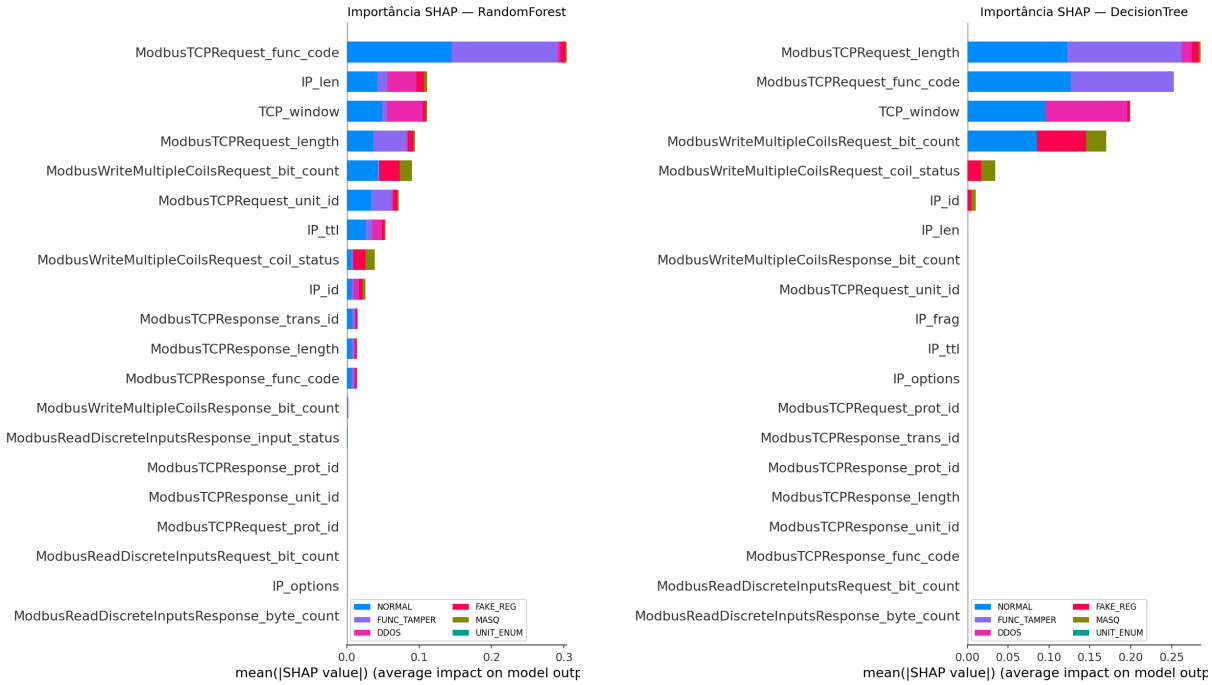
Classe	Precisão (%)	Revocação (%)	F1-score (%)	Suporte
DDOS	100.00	100.00	100.00	195 628
FAKE_REG	28.38	90.57	43.22	31 339
FUNC_TAMPER	100.00	99.50	99.75	436 205
MASQ	98.25	53.46	69.24	153 899
NORMAL	99.63	100.00	99.82	1 816 372
UNIT_ENUM	99.61	19.91	33.19	3 812
Acurácia global		96.97%		
F1-weighted		97.27%		

Fonte: Própria (2025)..

Os resultados mostram que, após a remoção dos campos estruturais adicionais, ambos os modelos mantiveram desempenho praticamente idêntico ao do Experimento 4, sem perda geral de acurácia ou de F1-score ponderado. No entanto, classes específicas, em especial *MASQ* e *FAKE_REG*, continuaram apresentando queda significativa de revocação, indicando que parte das características úteis para distinguir esses ataques se concentrava justamente nos campos estruturais removidos.

As Figuras 6.6 ilustram a distribuição das importâncias SHAP após a nova filtragem.

Figura 6.5: Importância média das variáveis no Experimento 5 segundo valores SHAP.



(a) Importância SHAP — *RandomForest*. (b) Importância SHAP — *DecisionTree*.

Figura 6.6: Importância média das variáveis no Experimento 5 segundo valores SHAP.

Fonte: Própria (2025)..

Após a remoção dos atributos identificados como estruturais, o modelo passou a se apoiar predominantemente em características puramente Modbus/TCP, como *ModbusTCPRequest_func_code*, *ModbusTCPRequest_length*, *TCP_window* e *ModbusWriteMultipleCoilsRequest_bit_count*. Esse comportamento indica que o processo de depuração cumpriu seu objetivo: reduzir a influência de atributos alheios ao protocolo e concentrar a tomada de decisão em campos que expressam o conteúdo lógico das requisições Modbus.

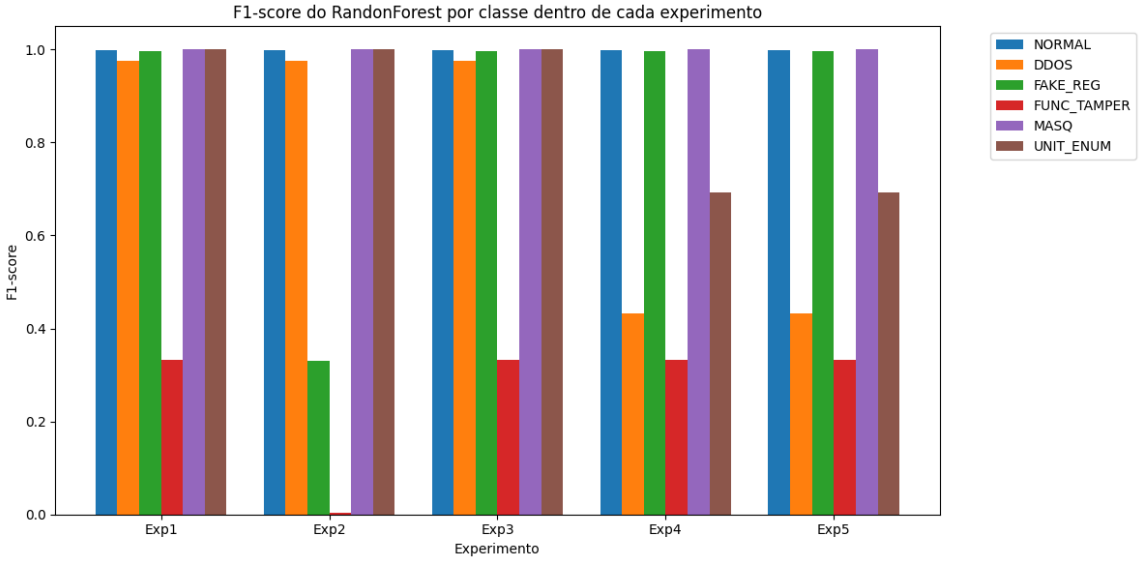
Ainda assim, a forte oscilação nas classes *MASQ* e *FAKE_REG* evidencia que a simples remoção de atributos pode eliminar não apenas vieses estruturais, mas também sinais úteis para a diferenciação de ataques mais sutis. Isso sugere que, embora metodologicamente coerente, o Experimento 5 não superou o desempenho dos experimentos anteriores, que apresentaram o melhor equilíbrio entre acurácia, estabilidade e relevância semântica das variáveis.

Em síntese, o Experimento 5 demonstrou que é possível eliminar completamente os campos temporais e estruturais sem comprometer a acurácia global.

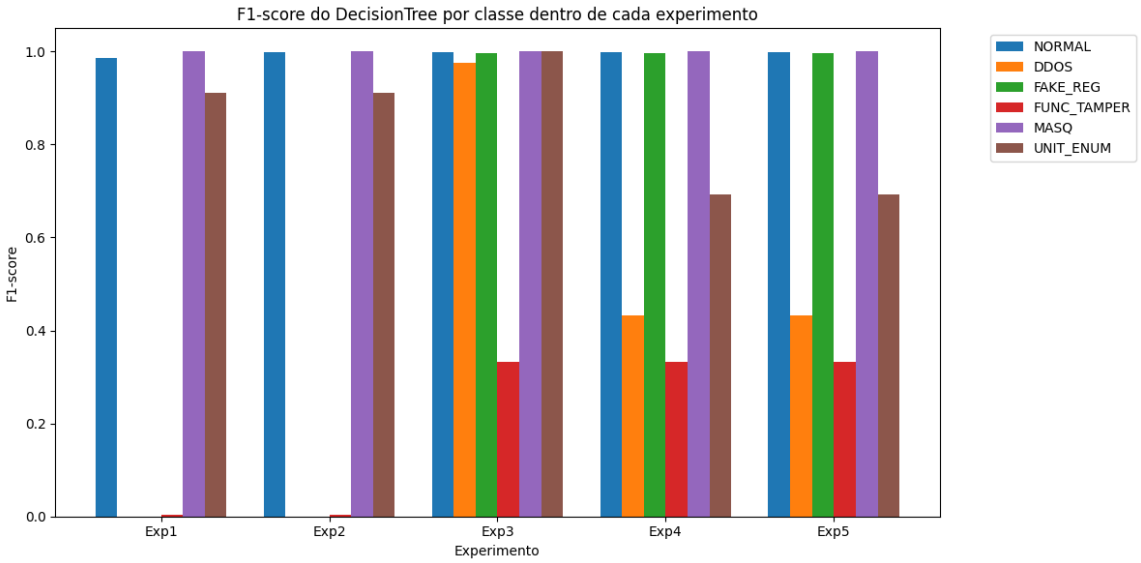
6.3.3.6 Conclusão dos Experimentos

A avaliação sequencial dos cinco experimentos permitiu compreender, de forma progressiva, como diferentes grupos de atributos influenciaram a capacidade dos modelos em discriminar comportamentos legítimos e maliciosos no tráfego Modbus/TCP. A combinação dos resultados quantitativos (Tabelas de desempenho) e qualitativos (valores SHAP), complementada pelos gráficos de F1-score por classe apresentados nas Figuras 6.7a e 6.7b, permitiu observar tendências claras na sensibilidade dos modelos a atributos estruturais, temporais e semânticos.

Figura 6.7: Comparação do F1-score por classe ao longo dos cinco experimentos para os modelos de árvore de decisão.



(a) F1-score do *RandomForestClassifier* por classe em cada experimento.



(b) F1-score do *DecisionTreeClassifier* por classe em cada experimento.

Fonte: Própria (2025)..

Os resultados do Experimento 1 confirmaram que o uso do conjunto completo de atributos remanescentes após o pré-processamento inicial levou a um desempenho global elevado, mas dependente de correlações artificiais. As Figuras de F1-score mostram que, embora todas as classes apresentassem métricas elevadas no RandomForest, variáveis como ModbusTCPRequest_trans_id e timestamp dominavam a decisão do modelo nos valores SHAP, evidenciando viés temporal e dependência da

ordenação dos pacotes.

A remoção isolada de trans_id, avaliada no Experimento 2, resultou em queda abrupta no desempenho de classes minoritárias, especialmente FAKE_REG e FUNC_TAMPER, visível nos gráficos de F1-score. Isso demonstrou que esse atributo, embora indesejado do ponto de vista semântico, havia se tornado um atalho decisório para o modelo. O RandomForest apresentou resiliência parcial, mas não recuperou a estabilidade observada no Experimento 1.

O Experimento 3 representou o ponto de inflexão no processo de depuração. Com a remoção simultânea de trans_id e timestamp, os modelos passaram a se apoiar majoritariamente em atributos funcionais do protocolo. As Figuras de F1-score mostram que todas as classes, incluindo FAKE_REG e FUNC_TAMPER, retornaram a um patamar de desempenho elevado e estável. Os valores SHAP indicaram um deslocamento consistente para variáveis modbus, como func_code, unit_id e bit_count. Esse equilíbrio entre desempenho, estabilidade entre classes e coerência semântica justificou a adoção desse experimento como referência principal.

Nos Experimentos 4 e 5, avaliou-se a remoção adicional de atributos estruturais como IP_tos, IP_len, TCP_flags e campos derivados. Embora os modelos tenham mantido alta acurácia e F1-score global, os gráficos evidenciam degradação expressiva nas classes MASQ e FAKE_REG, cujas revocações caíram significativamente. A análise SHAP revelou que parte das diferenças comportamentais entre esses ataques estava justamente nos campos estruturais removidos, indicando que a depuração excessiva eliminou sinais úteis e reduziu a capacidade discriminativa do modelo.

A análise conjunta das métricas evidencia que o Experimento 3 apresenta o melhor compromisso entre desempenho global, estabilidade entre classes, ausência de viés temporal e interpretabilidade. Os gráficos de F1-score reforçam este cenário mantêm em todas as classes desempenho elevado e consistente, sem dependência de atributos artificiais, o que torna essa configuração a mais adequada para uso em ambientes reais.

6.4 Discussão dos Resultados

A análise conjunta dos resultados de DPI e de ML permite compreender de forma mais clara o papel de cada trilha dentro do FARAONIC. Enquanto o módulo baseado em regras opera diretamente sobre campos específicos do protocolo Modbus/TCP, o módulo de aprendizado de máquina explora padrões multidimensionais do tráfego, oferecendo maior flexibilidade e capacidade de generalização.

Do ponto de vista quantitativo, a Tabela 6.2 mostra que o módulo de DPI apresentou desempenho excelente nos ataques Atq1–Atq4 (FAKE_REG, FUNC_TAMPER,

UNIT_ENUM e DDOS), com *F1-score* igual ou superior a 92%. Nesses cenários, as anomalias são fortemente caracterizadas por violações explícitas da estrutura Modbus (registros inexistentes, códigos de função inválidos ou enumeração evidente de *unit_id*), o que favorece a aplicação de regras determinísticas simples. Os modelos de ML, especialmente na configuração do Experimento 3, também obtiveram métricas elevadas para essas classes, conforme as Figuras 6.7a e 6.7b. Entretanto, o ganho adicional em relação ao DPI foi marginal.

Sob a perspectiva de engenharia de sistemas, para os ataques Atq1–Atq4, técnicas determinísticas de menor complexidade já entregam desempenho próximo ao ótimo, com menor custo computacional, menor necessidade de manutenção de modelos e menor complexidade de implantação. Em outras palavras, para esse conjunto de ameaças, não há benefício claro em substituir ou priorizar ML em detrimento de regras baseadas em DPI. Portanto, a opção mais eficiente é explorar o módulo determinístico como primeira linha de defesa.

O cenário se modifica no ataque Atq5 (MASQ). Na Tabela 6.2, o DPI atinge *F1-score* de 77,38%, com revocação de 63,02%, refletindo a dificuldade em capturar um comportamento projetado justamente para se confundir com tráfego legítimo. Já nos modelos de ML, a classe MASQ apresenta desempenho substancialmente superior, com *F1-score* próximo de 100% em ambos os classificadores. Os gráficos de F1-score por classe (Figuras 6.7a e 6.7b) evidenciam essa diferença de forma clara: enquanto o desempenho do DPI em MASQ permanece limitado, os modelos de ML conseguem explorar combinações sutis de atributos Modbus/TCP e IP para separar comunicações legítimas de tentativas de mascaramento.

Essa diferença de comportamento corrobora a motivação original do FARAONIC como *framework* híbrido. Em ataques com assinatura estrutural bem definida (FAKE_REG, FUNC_TAMPER, UNIT_ENUM e DDOS), o módulo de DPI é suficiente e mais eficiente em termos de recursos, reservando o uso de ML para situações em que padrões mais complexos e distribuídos precisam ser inferidos, como no caso de MASQ. Dessa forma, o ML não substitui as regras determinísticas, mas as complementa, cobrindo regiões do espaço de ameaças em que a lógica baseada exclusivamente em regras se mostra insuficiente.

Em síntese, os resultados indicam que:

a) para os ataques Atq1–Atq4, o DPI oferece desempenho excelente com menor custo computacional e menor complexidade de implementação, tornando desnecessário o uso de ML como mecanismo primário de detecção;

b) para o ataque Atq5 (MASQ), os modelos de ML, especialmente na configuração do Experimento 3, superam claramente o desempenho do DPI, justifi-

cando seu emprego como camada adicional de detecção;

c) o uso combinado de ambas as trilhas é o que melhor aproveita os recursos disponíveis: regras simples e eficientes para ataques de assinatura clara e modelos de ML para comportamentos mais sutis ou camuflados.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este capítulo apresenta as conclusões gerais do trabalho, sintetizando as contribuições obtidas com o desenvolvimento e a validação do FARAONIC, bem como suas limitações e os principais caminhos identificados para evolução futura. As considerações aqui expostas resultam da integração entre a fundamentação teórica, a proposta arquitetural, o ambiente experimental e os resultados de detecção obtidos com as trilhas de inspeção profunda de pacotes (DPI) e aprendizado de máquina (ML).

7.1 Conclusões

As redes OT baseadas em Modbus/TCP apresentam fragilidades inerentes, decorrentes da falta de autenticação, da simplicidade estrutural do protocolo e da previsibilidade das operações. Essas características ampliam a superfície de ataque e favorecem ações que vão desde abusos explícitos dos campos do protocolo até comportamentos mais sutis que se misturam ao tráfego legítimo. Embora a literatura ofereça abordagens baseadas em inspeção profunda de pacotes e em aprendizado de máquina, ainda existe uma lacuna entre transparência, interpretabilidade e capacidade de adaptação em tempo real. Diante desse cenário, o problema central que motivou esta dissertação foi desenvolver um framework capaz de combinar a clareza e a precisão determinística da DPI com a capacidade adaptativa dos modelos supervisionados, sem comprometer a disponibilidade do ambiente industrial.

O objetivo central desta dissertação foi propor e avaliar um *framework* híbrido de segurança para redes OT baseadas em Modbus/TCP, capaz de operar em tempo real. Para alcançar esse propósito, integrou-se captura passiva de tráfego, inspeção profunda do protocolo e modelos de aprendizado de máquina voltados à classificação comportamental. Os objetivos específicos definidos no Capítulo 1 foram atendidos ao longo do trabalho. Inicialmente, foi implementado um módulo de captura passiva por *port mirroring*, tornando possível observar o tráfego Modbus/TCP sem interferir no processo de controle. A partir dessa captura, estabeleceu-se uma *baseline* de comunicações legítimas, empregada tanto na calibração das regras de DPI quanto na identificação dos padrões normais de operação. Em seguida, foram definidas e aplicadas regras de inspeção profunda de pacotes orientadas à semântica do protocolo, com enfoque em códigos de função, endereçamento e conteúdo de dados. Essas regras não têm por finalidade detectar ataques *zero-day*, mas sim captar violações explícitas da estrutura e do comportamento esperado das mensagens Modbus/TCP.

Paralelamente, um ambiente experimental completo foi desenvolvido, integrando OpenPLC, Factory I/O, pfSense e estações destinadas a atividades defensivas e ofen-

sivas, o que permitiu gerar de forma controlada tráfego legítimo e malicioso. A partir do tráfego coletado, foram gerados dois *datasets* rotulados e distintos, elaborados de forma independente mas com a mesma metodologia experimental. Cada *dataset* contém tanto tráfego legítimo quanto a reprodução dos cinco ataques considerados no estudo, executados em instantes diferentes e por máquinas atacantes distintas, de modo a representar cenários variados de operação. Esses conjuntos permitiram o desenvolvimento, a validação e a comparação dos modelos supervisionados. Ao final, foi possível analisar de forma integrada os papéis e limites da detecção baseada em regras e da detecção comportamental.

A partir desses elementos, o FARAONIC se consolidou como um *framework* híbrido em duas trilhas independentes e complementares: uma trilha determinística, baseada em DPI e regras Modbus/TCP, voltada à identificação de ataques com assinatura clara e violações explícitas da semântica do protocolo; uma trilha adaptativa, construída sobre modelos de árvore de decisão e floresta aleatória, ajustados a partir de um processo de depuração progressiva de atributos, guiado por valores *SHAP* e por critérios de coerência semântica com o protocolo.

Os resultados experimentais mostraram que a trilha de DPI é suficiente e altamente eficiente para ataques cuja manifestação altera de forma evidente a estrutura das mensagens (por exemplo, registradores inexistentes, *function codes* indevidos, enumeração explícita de *unit_id* ou padrões claros de *DoS*). Nesses cenários, o custo computacional é reduzido, a lógica de detecção é transparente e a operação em tempo real é mantida com folga.

Por outro lado, ataques projetados para se confundir com o tráfego legítimo, como as tentativas de mascaramento de origem, evidenciaram a necessidade de um mecanismo mais flexível. Nesse contexto, a trilha de ML mostrou-se capaz de explorar combinações sutis de atributos Modbus/TCP e IP para separar comunicações legítimas de padrões camuflados. O processo de experimentação em cinco etapas permitiu identificar uma configuração de atributos que elimina dependências artificiais de ordem e tempo (como *trans_id* e *timestamp*), preservando apenas variáveis diretamente ligadas à semântica do protocolo.

O conjunto dos resultados obtidos confirma a viabilidade do FARAONIC como um *framework* híbrido para detecção de intrusões em redes OT. A arquitetura integra visibilidade passiva, detecção determinística e detecção adaptativa, mantendo aderência às restrições de latência e disponibilidade que caracterizam ambientes industriais. A complementaridade entre as trilhas amplia a cobertura diante dos diferentes perfis de ataque e fornece um caminho coerente para soluções interpretáveis, reprodutíveis e ajustadas ao contexto de sistemas de controle.

7.2 Limitações

Apesar dos resultados promissores, algumas limitações relevantes foram identificadas ao longo do desenvolvimento do trabalho. A primeira está relacionada à escalabilidade do sistema de captura em situações de saturação extrema. Observou-se, no ambiente experimental, uma limitação prática em torno de 28,000 pacotes por janela de observação, o que reduz a eficácia do módulo de DPI em cenários de ataques de negação de serviço com taxas muito elevadas de tráfego. Acima desse volume, parte dos pacotes deixa de ser analisada, indicando a necessidade de paralelização do processamento ou de otimizações específicas na camada de captura para lidar com picos de tráfego mais intensos.

Outra limitação está associada ao escopo protocolar adotado. Este estudo concentrou-se exclusivamente no protocolo Modbus/TCP, enquanto ambientes industriais reais operam frequentemente com múltiplos protocolos, como DNP3, IEC 104, S7Comm e IEC 61850. A interação entre esses protocolos influencia não apenas o perfil de normalidade, mas também o espaço de ataque, de modo que a avaliação restrita a um único protocolo não captura toda a complexidade presente em redes OT heterogêneas.

Também se destaca a ausência de mecanismos nativos de resposta automática. Embora o *framework* ofereça detecção em tempo real e geração de alertas, a etapa de reação ainda depende de intervenção manual ou de ferramentas externas. Não foram implementadas funcionalidades de contenção, isolamento de tráfego suspeito ou reconfiguração automática de elementos de rede, o que limita o uso do sistema como ferramenta completa de detecção e resposta.

Por fim, o ambiente experimental utilizado, embora representativo e adequado para validação controlada, ainda apresenta variabilidade limitada quando comparado à diversidade das plantas industriais reais. Aspectos como a presença de diferentes fabricantes, múltiplos segmentos de rede, caminhos redundantes, dispositivos legados e eventos operacionais prolongados não foram contemplados integralmente, o que sugere cautela ao generalizar diretamente os resultados obtidos.

Essas limitações não comprometem a validade do *framework* proposto, mas indicam pontos essenciais a serem considerados em sua eventual transposição para ambientes produtivos e orientam de maneira direta as oportunidades de continuidade deste trabalho.

7.3 Considerações Finais

O trabalho desenvolvido demonstra que a integração entre inspeção profunda de pacotes e aprendizado de máquina constitui uma abordagem sólida para enfrentar

a crescente complexidade das ameaças em redes OT. A opção por captura passiva via *port mirroring*, a ênfase em atributos semanticamente coerentes com o protocolo Modbus/TCP e o uso combinado de trilhas determinísticas e adaptativas resultaram em um *framework* capaz de operar em tempo real, com elevada capacidade de detecção e sem interferir na continuidade do processo industrial.

Além da arquitetura proposta, a construção do ambiente experimental e do *dataset* rotulado representa uma contribuição prática relevante: ambos fornecem uma base reprodutível para a avaliação de estratégias de defesa e para o treinamento de modelos de detecção em redes OT. Ao articular teoria, implementação e validação experimental, o FARAONIC aproxima o estado da arte em segurança de ICS das demandas concretas de infraestruturas críticas.

Espera-se que as soluções e reflexões apresentadas nesta dissertação sirvam de apoio para a evolução de mecanismos de detecção em redes industriais, contribuindo para a proteção de sistemas que sustentam serviços essenciais à sociedade. A consolidação de arquiteturas híbridas, interpretáveis e alinhadas às restrições de operação em tempo real é um passo importante na direção de ambientes OT mais resilientes frente a ameaças cibernéticas cada vez mais sofisticadas.

7.4 Trabalhos Futuros

As limitações identificadas ao longo deste trabalho e as potencialidades abertas pela arquitetura do FARAONIC apontam para diversas direções naturais de continuidade. Um primeiro caminho consiste no aprimoramento do subsistema de captura, explorando técnicas de paralelização, balanceamento de carga entre sensores e o uso de drivers e bibliotecas de alto desempenho, de forma a ampliar a capacidade de análise sob taxas elevadas de tráfego sem perda de eventos relevantes. Outro eixo importante envolve a extensão do *framework* para múltiplos protocolos industriais, incorporando gradualmente DNP3, IEC 104, S7Comm e IEC 61850, permitindo que o sistema reflita a heterogeneidade presente em ambientes reais e opere em cenários mais complexos.

Também se destaca a necessidade de integração com mecanismos de resposta automática, possibilitando ações em tempo real, como isolamento temporário de endereços suspeitos, reconfiguração dinâmica de regras de *firewall* ou geração de eventos estruturados para plataformas *SIEM* e *SOAR*. No campo do aprendizado de máquina, há espaço para investigar modelos mais avançados, incluindo métodos de *boosting*, como *Gradient Boosting*, *XGBoost* e *LightGBM*, além de redes neurais leves capazes de manter interpretabilidade e tempos de inferência compatíveis com as restrições de operação em sistemas OT.

Outra vertente promissora consiste na exploração de estratégias para detecção de ataques sem rótulo prévio, por meio de abordagens não supervisionadas ou semi-supervisionadas, como *autoencoders*, modelos *one-class* e técnicas baseadas em densidade, adequadas a cenários em que não é possível rotular exaustivamente todos os tipos de ataque. Ademais, a validação do *framework* em ambientes industriais reais constitui um passo relevante, permitindo avaliar seu comportamento sob diferentes perfis de tráfego, cargas reais, múltiplos controladores e eventos operacionais prolongados em plantas piloto ou redes ICS de parceiros industriais.

Por fim, o *dataset* construído pode ser ampliado com novos cenários, ataques e padrões de tráfego, favorecendo comparações entre diferentes técnicas de detecção e contribuindo para a consolidação de *benchmarks* específicos para Modbus/TCP. Essas linhas de continuidade podem ser exploradas de forma incremental, aproveitando a infraestrutura e os *scripts* já desenvolvidos, de modo a ampliar progressivamente o alcance, a robustez e a aplicabilidade do FARAONIC.

REFERÊNCIAS

Wael Alsabbagh, Samuel Amogbonjaye, Diego Urrego, and Peter Langendörfer. **A Stealthy False Command Injection Attack on Modbus based SCADA Systems**. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, pages 1–9, 2023. doi: 10.1109/CCNC51644.2023.10059804.

Juarês Barbosa Assunção Júnior. **Convergência entre Redes OT e TI**: Benefícios e desafios. *Operational Technology Journal*, 2018.

V. Barbosa and M. Caselli. **Sistemas de Detecção de Intrusão Híbridos para Redes OT**. *Revista de Segurança Cibernética*, 2022.

Ana Karoline Carneiro and Juarês Barbosa Assunção Júnior. **Segurança do Protocolo Modbus/TCP em Redes OT**. *Journal of Industrial Security*, 2019.

Ana Karoline Carneiro and Juarês Barbosa Assunção Júnior. **Sistemas de Controle Industrial**: Uma revisão. *Revista de Automação Industrial*, 2020.

Manuel Cheminod, Luca Durante, and Adriano Valenzano. **Review of security issues in industrial networks**. *IEEE transactions on industrial informatics*, 9(1):277–293, 2012.

Manuel Cheminod, Luca Durante, and Adriano Valenzano. **Review of Security Issues in Industrial Networks**. *IEEE Transactions on Industrial Informatics*, 9(1):277–293, 2013.

Israel Barbosa de Brito and Rafael T de Sousa Jr. **Development of an open-source testbed based on the modbus protocol for cybersecurity analysis of nuclear power plants**. *Applied Sciences*, 12(15):7942, 2022.

D. Goldenberg and Y. Kleinman. **Impacto da Latência na Inspeção Profunda de Pacotes**. *Latency Studies*, 2020.

Petr Ilgner and Radek Fujdiak. **Fuzzing ICS Protocols**: Modbus fuzzer framework. In *2022 IEEE International Carnahan Conference on Security Technology (ICCST)*, pages 1–6. IEEE, 2022.

Y. Kleinman and V. Barbosa. **Inspeção Profunda de Pacotes em Ambientes Industriais**. *Industrial Networks Review*, 2021.

E.D. Knapp. ***Industrial Network Security**: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2024. ISBN 9780443137389. URL https://books.google.com.br/books?id=tE7VEAAAQBAJ.

James F. Kurose and Keith W. Ross. ***Computer Networking**: A Top-Down Approach*. Pearson, 7th edition, 2016.

Lagineni Mahendra, RK Senthil Kumar, Reddi Hareesh, BS Bindhumadhava, and Rajesh Kalluri. **Deep Security Scanner for Industrial Control Systems**. In *TENCON 2021-2021 IEEE Region 10 Conference (TENCON)*, pages 447–452. IEEE, 2021.

T. Morris, D. Goldenberg, and Y. Kleinman. **Vulnerabilidades de Segurança no Modbus/TCP**. *Cybersecurity Journal*, 2021.

T. Morris, D. Goldenberg, and V. Barbosa. **Uso de Aprendizado de Máquina em IDS para Redes OT**. *Journal of Machine Learning in Security*, 2023.

Osborn N. Nyasore, Pavol Zavarsky, Bobby Swar, Raphael Naiyeju, and Shubham Dabra. **Deep Packet Inspection in Industrial Automation Control System to Mitigate Attacks Exploiting Modbus/TCP Vulnerabilities**. In *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity)*, pages 241–245, 2020.

Andrea Pinto, Luis-Carlos Herrera, Yezid Donoso, and Jairo A Gutierrez. **Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure**. *Sensors*, 23(5):2415, 2023.

Panagiotis Radoglou-Grammatikis, Ilias Siniosoglou, Thanasis Liatifis, Anastasios Kourouniadis, Konstantinos Rompolos, and Panagiotis Sarigiannidis. **Implementation and Detection of Modbus Cyberattacks**. In *2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, 2020. doi: 10.1109/MOCAST49295.2020.9200287.

Ramiz H Shikhaliyev et al. **Using machine learning methods for industrial control systems intrusion detection**. *Problems of Information Technology*, pages 37–48, 2023. doi: 10.25045/jpit.v14.i2.05.

Keith A. Stouffer, Victoria Y. Pillitteri, Shaun Lightman, et al. **Guide to Operational Technology (OT) Security** – initial public draft. Technical Report NIST SP 800-82 Revision 3, National Institute of Standards and Technology (NIST), 2024. Available at: https://doi.org/10.6028/NIST.SP.800-82r3.

Eric Wai and C. K. M. Lee. **Depth in Defense**: A multi-layered approach to cybersecurity for scada systems in industry 4.0. *Science and Technology Recent Updates and Future Prospects*, 2, 2024. doi: 10.9734/bpi/strufp/v2/12542F.