UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

VITOR SIMON DE SOUZA - 00325102

PROTÓTIPO DE UM SISTEMA DE DETECÇÃO DE ANOMALIAS E INTRUSÃO NÃO-INVASIVO PARA REDES DE AUTOMAÇÃO BASEADAS EM MODBUS TCP

Porto Alegre
2026

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

VITOR SIMON DE SOUZA - 00325102

PROTÓTIPO DE UM SISTEMA DE DETECÇÃO DE ANOMALIAS E INTRUSÃO NÃO-INVASIVO PARA REDES DE AUTOMAÇÃO BASEADAS EM MODBUS TCP

Trabalho de Conclusão de Curso submetido à COMGRAD/CCA da UFRGS como parte dos requisitos para a obtenção do título de *Bacharel em Engenharia de Controle e Automação.*

Orientador:
Prof. Dr. Marcelo Goetz

Porto Alegre
2026

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

VITOR SIMON DE SOUZA - 00325102

PROTÓTIPO DE UM SISTEMA DE DETECÇÃO DE ANOMALIAS E INTRUSÃO NÃO-INVASIVO PARA REDES DE AUTOMAÇÃO BASEADAS EM MODBUS TCP

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Atividade de *Trabalho de Conclusão de Curso CCA - II* e aprovado em sua forma final pelo Orientador e Banca Examinadora abaixo.

Orientador:
Prof. Dr. Marcelo Goetz, UFRGS
Doutor pela Universität Paderborn – Paderborn, Alemanha

Banca Examinadora:

Prof. Dr. Marcelo Goetz, UFRGS
Doutor pela Universität Paderborn – Paderborn, Alemanha

Prof. Dr. João Netto, UFRGS
Doutor pela Université Catholique de Louvain – Mechelen, Bélgica

Prof. Dr. Ivan Muller, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

---

Fabiano Disconzi Wildner
Coordenador de Curso
Eng. de Controle e Automação

Porto Alegre
Junho - 2026

AGRADECIMENTOS

À minha família, por sempre apoiar e prover o necessário para a continuação dos meus estudos.

À minha namorada Karoline, pelo apoio, carinho e conforto durante os inúmeros momentos difíceis da graduação.

Aos meus cachorros, Tazmania e Didivéia, que em meio a lambidas, brincadeiras, sonecas e carinhos, acompanharam plenamente o desenvolvimento do presente trabalho e também aprenderam sobre Modbus TCP.

Ao meu orientador Marcelo, pela disponibilidade e atenção na orientação do trabalho de conclusão, além dos inúmeros ensinamentos ao longo do curso.

Aos meus amigos e colegas da faculdade, que, ao partilhar dos mesmos desafios e noites mal dormidas sempre com bom humor e interesse, tornaram a jornada da graduação mais divertida e menos sofrida.

Aos amigos de outros grupos, que apesar do distanciamento que os caminhos da vida causam, sempre me acompanham, mesmo que às vezes de longe.

Aos times da Solae e da Altus S.A, que acreditaram no meu potencial profissional e sempre se flexibilizaram para que eu pudesse participar das aulas e atender aos horários de estudo.

À equipe do projeto S2C2, que foi meu primeiro contato com iniciação científica e ajudou a viabilizar minha estadia na graduação e na UFRGS.

À UFRGS e ao IFSul, pela oportunidade de estudar e pela base técnica que me moldou como profissional.

Por fim, agradeço a todos que de alguma forma passaram por mim, pois todas as passagens me geraram experiência, a experiência gerou aprendizado, e o aprendizado moldou quem sou hoje.

RESUMO

A crescente integração entre a Tecnologia da Informação (IT) e a Tecnologia de Operação (OT) na Indústria 4.0 tem exposto sistemas de controle industrial a novas ameaças cibernéticas. Protocolos legados, como o Modbus TCP, dominam o chão de fábrica, mas carecem de mecanismos de segurança fundamentais, como autenticação e criptografia, tornando as redes de automação vulneráveis. Dada a inviabilidade de substituir esses sistemas legados, surge a necessidade de soluções alternativas. Este trabalho descreve o desenvolvimento e a validação de um protótipo de Sistema de Detecção de Intrusão (IDS) não-invasivo focado em redes Modbus TCP. A metodologia envolveu a criação de um *sniffer* de rede em Python, utilizando a biblioteca Scapy para captura e decodificação de pacotes, com regras baseadas em *Function Codes* e volume de tráfego. Os alertas de segurança foram publicados dinamicamente via protocolo MQTT e visualizados em um dashboard Node-RED. O sistema foi validado em um ambiente de teste controlado, composto por um Controlador Lógico Programável (CLP) físico e o simulador de planta industrial FlexFact. Os resultados demonstraram que o IDS foi capaz de detectar anomalias volumétricas e injeções de comandos não permitidos de forma imediata, comprovando a viabilidade de monitoramento de segurança sem a introdução de latência na rede de controle operacional.

Palavras-chave: Modbus TCP, Redes de Automação, Segurança Cibernética Industrial, Sistema de Detecção de Intrusão, Sniffer de Rede

ABSTRACT

The growing integration between Information Technology (IT) and Operational Technology (OT) in Industry 4.0 has exposed industrial control systems to new cybersecurity threats. Legacy protocols, such as Modbus TCP, dominate the factory floor but lack fundamental security mechanisms, such as authentication and encryption, making automation networks vulnerable. Given the infeasibility of replacing these legacy systems, the need for alternative solutions arises. This work describes the development and validation of a non-invasive Intrusion Detection System (IDS) prototype focused on Modbus TCP networks. The methodology involved creating a network *sniffer* in Python, using the Scapy library for packet capture and decoding, with rules based on Function Codes and traffic volume. Security alerts were published using MQTT protocol and visualized on a Node-RED dashboard. The system was validated in a controlled test environment composed of a physical Programmable Logic Controller (PLC) and the FlexFact industrial plant simulator. The results demonstrated that the IDS was able to detect volumetric anomalies and injection of unauthorized commands immediately, proving the feasibility of security monitoring without introducing latency in the operational control network.

Palavras-chave: Automation Networks, Industrial Cybersecurity, Intrusion Detection System, Modbus TCP, Network Sniffer

LISTA DE ILUSTRAÇÕES

1 Estrutura do frame Modbus. 18
2 Arquitetura da solução e fluxo de dados do protótipo. 24
3 Layout da planta industrial simulada no software FlexFact. 26
4 Fluxo de nós implementado no Node-RED. 32
5 Interface do Dashboard desenvolvido no Node-RED. 33
6 Dashboard do Node-RED registrando o ataque de DoS. 35
7 Dashboard do Node-RED alertando o comando de escrita por IP não autorizado. 36
8 Dashboard do Node-RED registrando a injeção de comando não autorizado com IP Spoofing. 37

LISTA DE TABELAS

1 Síntese dos Trabalhos Relacionados e Comparação com a Solução Proposta. 23
2 Resumo das regras de detecção implementadas no IDS e vulnerabilidades mitigadas. 29

LISTA DE ABREVIATURAS

BPF	Berkeley Packet Filter
CLP	Controlador Lógico Programável
DoS	Denial of Service (Negação de Serviço)
DPI	Deep Packet Inspection (Inspeção Profunda de Pacotes)
FC	Function Code (Código de Função)
IACS	Industrial Automation and Control Systems (Sistemas de Automação e Controle Industrial)
ICS	Industrial Control System (Sistema de Controle Industrial)
IDS	Intrusion Detection System (Sistema de Detecção de Intrusão)
IHM	Interface Homem-Máquina
IIoT	Industrial Internet of Things (Internet Industrial das Coisas)
IP	Internet Protocol (Protocolo de Internet)
IT	Information Technology (Tecnologia da Informação)
JSON	JavaScript Object Notation
MBAP	Modbus Application Protocol header
MITM	Man-in-the-Middle
MQTT	Message Queuing Telemetry Transport
OSI	Open Systems Interconnection
OT	Operational Technology (Tecnologia de Operação)
PDU	Protocol Data Unit (Unidade de Dados do Protocolo)
RTU	Remote Terminal Unit (Unidade Terminal Remota)
SCADA	Supervisory Control and Data Acquisition (Supervisão e Aquisição de Dados)
TCP	Transmission Control Protocol

SUMÁRIO

1	Introdução	11
1.1	Problema	12
1.2	Justificativa	13
1.3	Objetivos	14
1.3.1	Objetivo Geral	14
1.3.2	Objetivos Específicos	14
1.4	Estrutura do Trabalho	14
2	Fundamentação Teórica e Estado da Arte	15
2.1	Segurança Cibernética em Sistemas de Controle Industrial (ICS)	15
2.1.1	Métricas de Segurança em Redes OT	15
2.1.2	A Norma ISA/IEC 62443	16
2.2	O Protocolo Modbus	17
2.2.1	O Encapsulamento Modbus TCP	17
2.2.2	Vulnerabilidades Inerentes	19
2.3	Análise de Tráfego e Sniffers de Rede	19
2.3.1	Sistemas de Detecção de Intrusão (IDS) e Linhas de Base (*Baseline*)	20
2.3.2	Inspeção de Carga Útil Modbus contra Falsificação de IP	21
2.4	Trabalhos Relacionados	21
3	Metodologia	24
3.1	Arquitetura da Solução	24
3.1.1	Domínio da Rede de Automação (OT)	25
3.1.2	Domínio da Nuvem e Visualização (IT)	27
3.2	Desenvolvimento do Sniffer	28
3.2.1	Captura e Filtragem	28
3.2.2	Decodificação (Parser)	28
3.2.3	Análise e Detecção de Anomalias	29
3.2.4	Comunicação com a Nuvem	30
3.2.5	Visualização de dados	31
3.3	Metodologia de Validação e Testes	31
4	Resultados e Discussão	34
4.1	Validação da Linha de Base	34
4.2	Avaliação do Ataque de Negação de Serviço (DoS)	34
4.3	Detecção de Acesso Não Autorizado	35
4.4	Análise da Detecção de Comandos Não Autorizados e IP Spoofing	36

5 Conclusões 38
5.1 Considerações Finais 38
5.2 Cumprimento dos Objetivos 39
5.3 Limitações e Trabalhos Futuros 39
Apêndices 41
Apêndice A - Documentação do Repositório de Código 42

1 INTRODUÇÃO

Atualmente a industria tem passado por um novo período de transformação, que comumente vem sendo chamado de Quarta Revolução Industrial, ou Indústria 4.0 (LASI *et al.*, 2014). Este período tem sido caracterizado pela crescente digitalização da manufatura e pela integração entre a Tecnologia da Informação (IT) — como redes corporativas, computação em nuvem e análise de dados — e os sistemas de controle de chão de fábrica, conhecidos como Tecnologia de Operação (OT) — como controladores, sensores e atuadores (LEZZI; LAZOI; CORALLO, 2018). Essa convergência entre sistemas de chão de fábrica e sistemas externos, facilitada por tecnologias habilitadoras como a Internet Industrial das Coisas (IIoT) e os Sistemas Ciber-Físicos, tem cada vez mais conectado fábricas e plantas à Internet com o objetivo de melhorar a eficiência e eficácia (SENGUPTA; RUJ; BIT, 2021; USTUNDAG; CEVIKCAN, 2018).

Os potenciais dessa transformação são otimistas, e incluem a otimização de processos através da análise de grandes volumes de dados, a redução do tempo de inatividade de máquinas por meio de manutenção preditiva, monitoramento remoto e aumento da produtividade (LASI *et al.*, 2014; LEZZI; LAZOI; CORALLO, 2018). A IIoT, em particular, permite que objetos inteligentes monitorem, coletem, analisem e tomem decisões sem intervenção humana (SENGUPTA; RUJ; BIT, 2021), possibilitando a criação de fábricas inteligentes ("Smart Factories"). Assim, essa capacidade de interconexão de sistemas e troca de dados dinâmica pode ser vista como um motor para ganhos de competitividade.

No entanto, essa crescente conectividade dos sistemas industriais, que historicamente operavam em ambientes isolados de redes externas (ERVURAL; ERVURAL, 2018) introduz novos desafios, especialmente no que diz respeito à segurança cibernética (LEZZI; LAZOI; CORALLO, 2018). Equipamentos industriais críticos tornam-se vulneráveis a uma variedade crescente de ciberataques, podendo afetar a operação, o modelo de negócio e até a segurança de funcionários (ERVURAL; ERVURAL, 2018). Relatórios indicam que uma parcela significativa das organizações já experienciou ataques cibernéticos em sua infraestrutura de OT, e espera-se que esses ataques se estendam cada vez mais da IT para a OT (LEZZI; LAZOI; CORALLO, 2018), de forma que a falta de mecanismos de segurança nativos em muitos protocolos e sistemas industriais legados acaba por agravar essa vulnerabilidade (SENGUPTA; RUJ; BIT, 2021).

Diante desse cenário, a segurança cibernética industrial tornou-se uma prioridade (LEZZI; LAZOI; CORALLO, 2018), sendo um elemento essencial para garantir a confiabilidade e segurança dos processos produtivos na era da Indústria 4.0 (ERVURAL; ERVURAL, 2018). Dessa forma, a capacidade de monitorar e analisar o tráfego de dados nas redes OT continuamente é fundamental para detectar atividades anômalas ou maliciosas que possam indicar falhas operacionais ou tentativas de intrusão, especialmente em sistemas legado (SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025), e um monitoramento não-invasivo pode ser o primeiro passo para a implementação de medidas eficazes de proteção e resposta a incidentes nesses ambientes críticos.

1.1 Problema

Dentre os diversos protocolos industriais legados que ainda dominam o chão de fábrica, o protocolo Modbus é considerado um padrão na industria (TRUNGADI *et al.*, 2025), sendo amplamente utilizado para a comunicação entre Controladores Lógicos Programáveis (CLPs) e na comunicação entre CLPs e sistemas de Supervisão e Aquisição de Dados (SCADA) (MEKALA *et al.*, 2023). Sua popularidade deve-se à sua implementação simples, sendo um padrão aberto e de domínio público (RAHMAN *et al.*, 2022). A variante Modbus TCP, em particular, encapsula o protocolo Modbus em pacotes TCP(Transmission Control Protocol)/IP, permitindo sua utilização sobre redes Ethernet padrão (EVANGELIOU I., 2018).

Uma característica inerente ao protocolo Modbus é que ele opera sem mecanismos nativos de segurança de rede, uma vez que foi desenvolvido em 1979 para redes seriais, fechadas e isoladas (RAHMAN *et al.*, 2022; MEKALA *et al.*, 2023). Dadas as características arquiteturais do protocolo supracitadas e a inviabilidade de alterar a base legada, os ataques podem acontecer, e ferramentas de diagnóstico tornam-se essenciais. As características de projeto que o tornam suscetível incluem:

- **Falta de Autenticação:** Não há um método de autenticação robusto em nenhum nível do protocolo (MEKALA *et al.*, 2023). Qualquer dispositivo na rede pode enviar um comando a um servidor (escravo) Modbus, que não tem como verificar a legitimidade do mestre (RAHMAN *et al.*, 2022).
- **Falta de Criptografia (Confidencialidade):** Todas as mensagens Modbus são transmitidas em texto claro (sem criptografia) através da rede (EVANGELIOU I., 2018; RAHMAN *et al.*, 2022; MEKALA *et al.*, 2023).
- **Falta de Integridade:** Não existem verificações de integridade na camada de aplicação que impeçam a modificação maliciosa dos dados de um pacote (EVANGELIOU I., 2018; MEKALA *et al.*, 2023).

Devido à essas vulnerabilidades, individuos podem explorar essas falhas para realizar operações maliciosas, como alterar a lógica de um CLP, parar e ligar equipamentos, abrir ou fechar válvulas(FENG; LAI; LIU, 2020), potencialmente causando danos físicos para o maquinário, processo e até funcionários. Os ataques mais comuns incluem o Reconhecimento/Escaneamento de rede), onde quem está atacando a rede pode descobrir dispositivos e mapear a arquitetura do sistema (EVANGELIOU I., 2018); Injeção de Comandos Falsos, onde comandos são enviados por um mestre ilegítimo (EVANGELIOU I., 2018; RAHMAN *et al.*, 2022); Interceptação e Modificação de Respostas (ataques *Man-in-the-Middle*(MITM)) (RAHMAN *et al.*, 2022); e Ataques de Negação de Serviço (DoS), que podem inundar um dispositivo com requisições, tornando-o incapaz de responder a comandos legítimos e causando um impacto fatal na estabilidade do sistema. Incidentes reais, como a paralisação de uma usina siderúrgica alemã, demonstram que tais ataques podem resultar em danos físicos significativos aos equipamentos (MILLER *et al.*, 2021).

Como ferramentas de segurança de IT tradicionais geralmente não são adequadas para o ambiente OT, que possui requisitos distintos de alta disponibilidade e tempo real, o diagnóstico dessas atividades maliciosas ou falhas operacionais acaba não sendo trivial (RAHMAN *et al.*, 2022). Além disso, soluções de segurança ativas (como um novo protocolo) são difíceis de implementar em sistemas legados, pois exigiriam uma substituição

completa e custosa dos equipamentos, que não oferecem suporte a novos protocolos ou monitoramento (RAHMAN *et al.*, 2022; MEKALA *et al.*, 2023). Surge, dessa forma, o desafio de como monitorar essas redes vulneráveis de forma não-invasiva, ou seja, sem interferir no tráfego de controle crítico e causar disturbios na rede. Há uma necessidade de ferramentas de diagnóstico, como *sniffers* de rede e Sistemas de Detecção de Intrusão (IDS), que possam capturar passivamente o tráfego Modbus TCP, decodificá-lo e analisá-lo de forma contínua e automatizada para identificar anomalias e padrões que possam se encaixar nas características de ataques (EVANGELIOU I., 2018).

1.2 Justificativa

As vulnerabilidades teóricas dos protocolos OT, como o Modbus TCP têm sido exploradas em incidentes com consequências severas no mundo físico. Exemplos históricos, como o ataque a uma usina siderúrgica alemã em 2014, onde a falha em desligar um alto-forno resultou em danos físicos significativos (MILLER *et al.*, 2021), e os ataques à rede elétrica da Ucrânia (BlackEnergy) (ERVURAL; ERVURAL, 2018), demonstram inequivocamente que ciberataques podem sabotar infraestruturas críticas (RAKAS; STOJANOVIĆ; MARKOVIĆ-PETROVIĆ, 2020), causar danos a equipamentos e até mesmo colocar vidas em risco (LEZZI; LAZOI; CORALLO, 2018).

Diante dessa ameaça, a solução aparente seria substituir os protocolos legados por versões seguras. No entanto, esta abordagem acaba sendo inviável na prática, pois o protocolo Modbus/TCP ainda domina a indústria devido à sua simplicidade e vasta base instalada de dispositivos (RAHMAN *et al.*, 2022; MEKALA *et al.*, 2023). A migração para um "Secure Modbus" precisaria de uma mudança de paradigma e estrutura, sendo um processo custoso que não é suportado pela maioria dos equipamentos legados (RAHMAN *et al.*, 2022; MEKALA *et al.*, 2023). Paradoxalmente, o uso de criptografia em novos protocolos também pode ser um desafio, pois impede que ferramentas de monitoramento de rede, como firewalls e IDS, inspecionem o tráfego em busca de anomalias (RAHMAN *et al.*, 2022).

Por essa razão, a literatura aponta o monitoramento de rede não-invasivo como a contramedida mais adequada e prática para sistemas OT em operação (SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025). Sistemas de Detecção de Intrusão (IDS) baseados em rede são considerados um componente essencial para a segurança de sistemas SCADA (RAKAS; STOJANOVIĆ; MARKOVIĆ-PETROVIĆ, 2020). Uma abordagem passiva, como um *sniffer*, permitiria a análise contínua do tráfego sem introduzir latência ou pontos de falha na rede de controle crítica, uma propriedade fundamental em redes de produção (SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025).

Assim, este trabalho se justifica pela necessidade crescente e o impacto real dos riscos cibernéticos em ambientes industriais, além da necessidade de desenvolver e validar ferramentas de baixo custo e código aberto para mitigar esse problema, de forma que a construção de um protótipo de *sniffer* especializado em Modbus TCP oferece uma contribuição de valor. Primeiramente, atende à necessidade de segurança, fornecendo um meio de detectar anomalias que indiquem tentativas de intrusão ou ataques em andamento (RAKAS; STOJANOVIĆ; MARKOVIĆ-PETROVIĆ, 2020). Em segundo lugar, serve como uma ferramenta de diagnóstico e manutenção para a engenharia de automação. A capacidade de decodificar e analisar o tráfego Modbus permite identificar não apenas ataques, mas também falhas operacionais, como erros de comunicação e dispositivos com

mau funcionamento (EVANGELIOU I., 2018).

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver e validar um protótipo de um Sistema de Detecção de Anomalias e Intrusão (IDS) não-invasivo para redes de automação baseadas em Modbus TCP.

1.3.2 Objetivos Específicos

- Desenvolver um software em Python utilizando a biblioteca Scapy para a captura e filtragem de pacotes em uma interface de rede.
- Implementar a lógica de decodificação (*parsing*) para extrair informações relevantes dos pacotes Modbus TCP.
- Desenvolver um módulo de análise para detectar anomalias pré-definidas, como acesso por IPs (Internet Protocol) não autorizados e ataques DoS.
- Integrar o protótipo com uma plataforma de nuvem utilizando o protocolo MQTT (Message Queuing Telemetry Transport) para o envio de alertas de segurança.
- Construir um dashboard de visualização em Node-RED para monitoramento contínuo dos alertas de segurança.
- Validar o funcionamento do protótipo em um ambiente de teste com uma planta industrial simulada (utilizando o software FlexFact) e um CLP físico.

1.4 Estrutura do Trabalho

O Capítulo 2 apresenta a fundamentação teórica sobre segurança cibernética industrial, o protocolo Modbus TCP e as técnicas de análise de tráfego, além de revisar os trabalhos já feitos na área. O Capítulo 3 detalha a metodologia utilizada para a concepção e implementação do protótipo de IDS e a arquitetura do ambiente de testes. O Capítulo 4 expõe a validação técnica da solução, analisando os testes de intrusão executados e a eficácia das regras de detecção. Por fim, o Capítulo 5 apresenta as considerações finais e perspectivas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E ESTADO DA ARTE

2.1 Segurança Cibernética em Sistemas de Controle Industrial (ICS)

Os Sistemas de Controle Industrial (ICS) constituem a base tecnológica de infraestruturas críticas e processos de manufatura automatizada, abrangendo desde a geração de energia e tratamento de águas até linhas de produção complexas (da silva *et al.*, 2025). Historicamente, esses sistemas operavam em ambientes isolados das redes corporativas e da internet — sob o paradigma do *air-gap* —, o que lhes conferia uma camada de proteção baseada no isolamento físico e na obscuridade de protocolos proprietários (cindrić; jurčević; hadjina, 2025).

Contudo, a ascensão da Indústria 4.0 e a integração da Internet Industrial das Coisas (IIoT) promoveram uma convergência profunda entre a Tecnologia da Informação (IT) e a Tecnologia de Operação (OT) (cindrić; jurčević; hadjina, 2025; da silva *et al.*, 2025). Essa interconectividade, embora tenha viabilizado ganhos significativos em eficiência e monitoramento remoto, expôs ativos industriais anteriormente protegidos a uma vasta gama de ameaças cibernéticas (brancati *et al.*, 2025). Diferentemente do ambiente de IT convencional, o domínio de OT é caracterizado por ativos com ciclos de vida extensos (muitas vezes superiores a 20 anos) e requisitos rigorosos de determinismo temporal, o que torna a aplicação de contramedidas de segurança tradicionais, como atualizações frequentes (*patching*) e criptografia pesada, um desafio de engenharia complexo (da silva *et al.*, 2025; brancati *et al.*, 2025).

Diante desse cenário, a proteção de ambientes ICS exige uma abordagem especializada que considere a interdependência entre a segurança lógica (*Cybersecurity*) e a segurança física (*Safety*). Para compreender como as estratégias de defesa devem ser moldadas para o chão de fábrica, é fundamental analisar as métricas que regem o gerenciamento de riscos industriais e os padrões normativos internacionais que orientam a segmentação dessas redes, conforme discutido nas subseções a seguir.

2.1.1 Métricas de Segurança em Redes OT

A avaliação da segurança da informação é tradicionalmente pautada em um modelo fundamental conhecido como a Tríade CIA, que engloba a Confidencialidade, a Integridade e a Disponibilidade (*Availability*) (cindrić; jurčević; hadjina, 2025). Em redes corporativas convencionais de Tecnologia da Informação (IT), a prioridade máxima recai quase invariavelmente sobre a Confidencialidade e a Integridade dos dados (da silva *et al.*, 2025). Nesse contexto, a exposição de informações sensíveis ou violação de dados de negócio representa o maior risco para a organização, enquanto atrasos ou interrupções

temporárias (impacto na disponibilidade) são eventos frequentemente tolerados em prol da segurança (DA SILVA *et al.*, 2025; CINDRIĆ; JURČEVIĆ; HADJINA, 2025).

No entanto, o gerenciamento de riscos em ambientes de Tecnologia de Operação (OT) e Sistemas de Controle Industrial (ICS) obedece a um paradigma fundamentalmente distinto (DA SILVA *et al.*, 2025). A prioridade da Tríade CIA inverte-se de forma drástica devido à natureza determinística e crítica dos processos físicos controlados (CINDRIĆ; JURČEVIĆ; HADJINA, 2025). Em redes OT, a Disponibilidade e a Integridade dos dados caminham lado a lado como os requisitos de segurança absolutos e mais rigorosos (CINDRIĆ; JURČEVIĆ; HADJINA, 2025).

A indisponibilidade de um sistema industrial ou a latência na comunicação podem levar a interrupções severas da produção, resultando não apenas em perdas financeiras catastróficas, mas também colocando em risco direto a segurança física (*Safety*) de operadores, dos equipamentos e do meio ambiente (DA SILVA *et al.*, 2025; CINDRIĆ; JURČEVIĆ; HADJINA, 2025). A integridade é igualmente vital, pois a manipulação maliciosa ou não intencional dos dados de sensores, ou a falsificação de comandos de atuadores, induzem o controlador lógico a tomar ações incorretas (CINDRIĆ; JURČEVIĆ; HADJINA, 2025). Por outro lado, a Confidencialidade é historicamente considerada o requisito menos crítico na hierarquia OT, uma vez que a leitura não autorizada de uma variável de processo pontual (sem alterá-la) raramente causa interrupções de curto prazo (CINDRIĆ; JURČEVIĆ; HADJINA, 2025).

Essa inversão de prioridades justifica a dificuldade técnica de se implementar soluções de segurança de IT (como *firewalls* restritivos ou criptografia pesada) diretamente no chão de fábrica (CINDRIĆ; JURČEVIĆ; HADJINA, 2025). Mecanismos que exigem bloqueios imediatos mediante falsos positivos ou que adicionem tempo de processamento afetam diretamente o princípio de alta disponibilidade (CINDRIĆ; JURČEVIĆ; HADJINA, 2025). Por conseguinte, a utilização de estratégias não invasivas — como o monitoramento passivo do tráfego através de Sistemas de Detecção de Intrusão (IDS) baseados em anomalias de *baseline* — apresenta-se como a contramedida técnica mais compatível com os rigorosos requisitos de continuidade do serviço da rede OT (CINDRIĆ; JURČEVIĆ; HADJINA, 2025).

2.1.2 A Norma ISA/IEC 62443

Para orientar a mitigação desses riscos e padronizar a proteção de infraestruturas industriais, a série de normas ISA/IEC 62443 estabeleceu-se como a referência global para conceitos, procedimentos e requisitos de cibersegurança em Sistemas de Automação e Controle Industrial (IACS) (JAATUN *et al.*, 2024; BRANCATI *et al.*, 2025; CINDRIĆ; JURČEVIĆ; HADJINA, 2025). A norma adota uma abordagem sistêmica com o objetivo de aprimorar a robustez do sistema, permitindo que a planta opere sob ameaças, e a sua resiliência, garantindo a rápida recuperação após um evento cibernético malicioso (CINDRIĆ; JURČEVIĆ; HADJINA, 2025).

A pedra angular arquitetural delineada pela IEC 62443-1-1 e aprofundada nos processos de avaliação de risco (IEC 62443-3-2) é a estratégia de Defesa em Profundidade através da segmentação da rede em "Zonas e Conduítes" (*Zones and Conduits*) (JAATUN *et al.*, 2024; BRANCATI *et al.*, 2025). Uma Zona de segurança agrupa ativos lógicos ou físicos que compartilham os mesmos requisitos de segurança e níveis de confiança operacionais (JAATUN *et al.*, 2024; BRANCATI *et al.*, 2025). Os Conduítes, por sua vez, são as vias lógicas de comunicação estritamente controladas que interligam diferentes zonas (JAATUN

et al., 2024; BRANCATI *et al.*, 2025). Essa compartimentação restringe a capacidade de propagação e o movimento lateral de um atacante caso uma fronteira externa seja violada (JAATUN *et al.*, 2024).

Além da estruturação em zonas, a norma estabelece a definição de Níveis de Segurança (*Security Levels* - SL), que variam do SL 1, voltado à proteção contra violações casuais, até o SL 4, que exige proteção contra ameaças persistentes avançadas, altamente motivadas e com habilidades específicas no domínio OT (BRANCATI *et al.*, 2025). A principal problemática enfrentada pela engenharia é que dispositivos legados, operando nativamente com o protocolo Modbus TCP, carecem dos mecanismos inerentes para atingir Níveis de Segurança elevados. Para solucionar esse abismo tecnológico sem descontinuar os ativos industriais já instalados, a norma ISA/IEC 62443 suporta o uso de contramedidas técnicas compensatórias (JAATUN *et al.*, 2024; BRANCATI *et al.*, 2025). O acoplamento de sistemas de Inspeção Profunda de Pacotes (DPI) na monitorização dos conduítes encaixa-se perfeitamente nessa diretriz, adicionando camadas ativas de auditoria capazes de suportar o SL-Target (Alvo) estipulado para as zonas da rede de automação (JAATUN *et al.*, 2024; BRANCATI *et al.*, 2025).

2.2 O Protocolo Modbus

O Modbus, desenvolvido originalmente pela Modicon (atual Schneider Electric) em 1979, é um protocolo de camada de aplicação para comunicação entre CLPs e também entre CLP e SCADA. Devido à sua simplicidade, robustez e ao fato de ser um padrão aberto e de domínio público, ele rapidamente se consolidou e virou padrão na indústria, sendo amplamente suportado por diversos fabricantes de equipamentos de automação (RAHMAN *et al.*, 2022; MEKALA *et al.*, 2023).

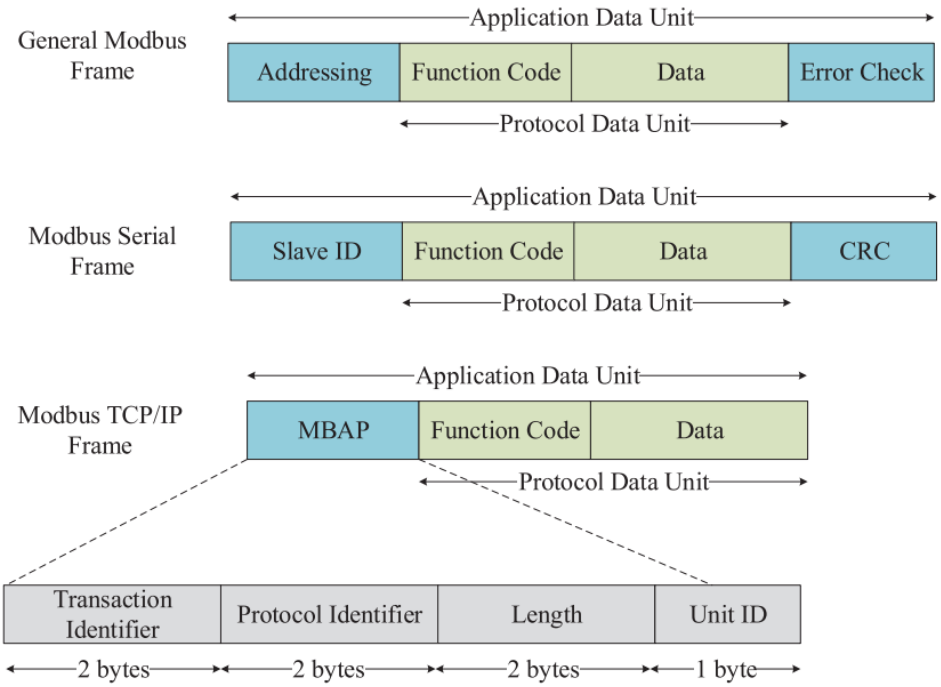
O protocolo opera em um modelo mestre-escravo (ou cliente-servidor no caso TCP). Isso quer dizer que, em uma comunicação Modbus, um dispositivo mestre (cliente), como uma IHM ou um sistema SCADA, inicia uma requisição (consulta) para um dispositivo escravo (servidor), como um CLP. O escravo então executa a ação solicitada e envia uma resposta de volta ao mestre (MODBUS ORGANIZATION, INC., 2012; PLIATSIOS *et al.*, 2020; AL-MAITAH; AL-KHAYER; DARAWSHEH, 2020).

A unidade de dados central do protocolo, também conhecido como **PDU (Protocol Data Unit)** é composto por dois campos principais:

- **Código de Função (1 byte):** Especifica a ação que o escravo deve executar, como "Ler Coils" (Código 01), "Escrever Registrador Único" (Código 06) ou "Escrever Múltiplos Registradores" (Código 16) (MODBUS ORGANIZATION, INC., 2012; EVANGELIOU I., 2018).
- **Dados (N bytes):** Contém as informações necessárias para a execução da função, como o endereço do registrador a ser lido/escrito e os dados a serem gravados (MODBUS ORGANIZATION, INC., 2012; PLIATSIOS *et al.*, 2020).

Quando implementado sobre redes seriais (Modbus RTU/ASCII), o PDU é encapsulado com um endereço de escravo e um campo de verificação de erros. No entanto, para integração com redes Ethernet modernas, foi criada a variante Modbus TCP (PLIATSIOS *et al.*, 2020). A estrutura completa do frame Modbus em suas variações RTU e TCP, incluindo o cabeçalho MBAP e o PDU, é ilustrada na Figura 1.

Figura 1: *Estrutura do frame Modbus.*



Fonte: Adaptado de Pliatsios *et al.* (2020).

2.2.1 O Encapsulamento Modbus TCP

O Modbus TCP, descrito na especificação "Modbus Messaging on TCP/IP Implementation Guide" (MODBUS ORGANIZATION, INC., 2006), simplesmente encapsula o PDU (Protocol Data Unit) Modbus dentro de um pacote TCP/IP padrão. A comunicação geralmente ocorre através da porta TCP 502, que é reservada para aplicações Modbus (RAHMAN *et al.*, 2022; MODBUS ORGANIZATION, INC., 2006).

Para gerenciar as transações sobre TCP, o Modbus TCP substitui o endereçamento de escravo e a verificação de erros serial por um cabeçalho específico chamado **MBAP (Modbus Application Protocol header)**.

Este cabeçalho de 7 bytes precede o PDU (PLIATSIOS *et al.*, 2020; AL-MAITAH; AL-KHAYER; DARAWSHEH, 2020; FENG; LAI; LIU, 2020; MODBUS ORGANIZATION, INC., 2006) e é composto por:

- **Identificador de Transação (2 bytes):** Um número usado pelo cliente para associar respostas a requisições, permitindo múltiplas requisições (PLIATSIOS *et al.*, 2020; RAHMAN *et al.*, 2022; MODBUS ORGANIZATION, INC., 2006).
- **Identificador de Protocolo (2 bytes):** É sempre zero (0x0000) para Modbus (PLIATSIOS *et al.*, 2020; RAHMAN *et al.*, 2022; MODBUS ORGANIZATION, INC., 2006).
- **Campo de Comprimento (2 bytes):** Define o número de bytes que se seguem (Unit ID + PDU) (PLIATSIOS *et al.*, 2020; RAHMAN *et al.*, 2022; MODBUS ORGANIZATION, INC., 2006).

- **Identificador de Unidade (1 byte):** Usado para roteamento em gateways que conectam a rede TCP/IP a uma sub-rede serial Modbus RTU (PLIATSIOS *et al.*, 2020; RAHMAN *et al.*, 2022; MODBUS ORGANIZATION, INC., 2006).

A verificação de erros do Modbus serial é omitida, pois o protocolo confia na verificação de erros já fornecida pela camada TCP (EVANGELIOU I., 2018).

2.2.2 Vulnerabilidades Inerentes

Conforme estabelecido na seção 1.1, a maior fraqueza do Modbus TCP deriva diretamente de sua concepção. Ele foi criado em 1979 para redes físicas isoladas, onde a confiança era implícita e a segurança não era uma preocupação (RAHMAN *et al.*, 2022; EVANGELIOU I., 2018; AL-MAITAH; AL-KHAYER; DARAWSHEH, 2020). Ao ser transportado para redes IP interconectadas, suas vulnerabilidades fundamentais foram expostas (PLIATSIOS *et al.*, 2020).

- **Falta de Confidencialidade:** O protocolo não especifica nenhum método de criptografia. Todos os dados, incluindo códigos de função, endereços de registradores e valores de processo, são transmitidos em "texto claro" (EVANGELIOU I., 2018; AL-MAITAH; AL-KHAYER; DARAWSHEH, 2020; MEKALA *et al.*, 2023; RAHMAN *et al.*, 2022). Isso torna o tráfego trivialmente legível por qualquer um que consiga escutar a rede (MEKALA *et al.*, 2023).

- **Falta de Autenticação:** A especificação do Modbus não inclui mecanismos de autenticação (EVANGELIOU I., 2018; AL-MAITAH; AL-KHAYER; DARAWSHEH, 2020; MEKALA *et al.*, 2023). Um servidor (escravo) não possui meios de verificar a identidade do cliente (mestre) que está enviando um comando. Qualquer dispositivo na rede pode forjar um pacote Modbus e enviá-lo à porta 502 de um CLP, que o processará como uma requisição legítima (RAHMAN *et al.*, 2022).

- **Falta de Integridade:** O protocolo não oferece proteção contra a modificação maliciosa de dados em trânsito (EVANGELIOU I., 2018; AL-MAITAH; AL-KHAYER; DARAWSHEH, 2020; MEKALA *et al.*, 2023). Embora o TCP forneça um *checksum* para detectar corrupção acidental de dados, ele não previne que um dispositivo em uma posição de MITM intercepte um pacote, altere seu conteúdo (ex: mude o valor de um sensor de 100 para 50) e recalcule o *checksum* antes de encaminhá-lo.

Estas falhas de projeto são a causa raiz que permite um vasto leque de ciberataques. A falta de confidencialidade permite o reconhecimento da rede e o *sniffing* de dados de produção (MEKALA *et al.*, 2023). A falta de autenticação e integridade permite ataques ativos, como injeção de comandos, *spoofing* de respostas e ataques *Man-in-the-Middle*, onde quem ataca pode ativamente perturbar ou assumir o controle do processo industrial (AL-MAITAH; AL-KHAYER; DARAWSHEH, 2020). Além disso, a simplicidade do protocolo e o baixo poder de processamento de muitos dispositivos escravos os tornam altamente suscetíveis a ataques DoS por inundação de pacotes (RAHMAN *et al.*, 2022).

2.3 Análise de Tráfego e Sniffers de Rede

Dada a inviabilidade de alterar os protocolos legados em sistemas críticos em operação (RAHMAN *et al.*, 2022), uma possível estratégia de defesa passa a ser o monitoramento da rede. A análise de tráfego de rede, comumente conhecida como *sniffing*, é o processo de capturar e inspecionar os pacotes de dados que trafegam por uma rede de computadores. Esta é uma técnica intrinsecamente passiva: o *sniffer* (farejador) apenas "escuta" a rede, copiando os pacotes sem introduzir distúrbios como latência ou interferir na comunicação original.

Embora o *sniffing* seja uma ferramenta primária para atacantes em fases de reconhecimento (MEKALA *et al.*, 2023), ele é também uma ferramenta defensiva essencial para administradores de rede, permitindo o diagnóstico de problemas e a detecção de atividades maliciosas (LI; QIN, 2017). Uma ferramenta de análise de pacotes muito conhecida para este fim é o Wireshark, amplamente utilizado tanto para análise manual e depuração (FENG; LAI; LIU, 2020) quanto para a captura de tráfego em testes de segurança (RADOGLOU-GRAMMATIKIS *et al.*, 2020).

Quando a análise de tráfego é automatizada para fins de segurança, ela forma a base de um Sistema de Detecção de Intrusão (IDS). Um IDS de rede é uma contramedida essencial e amplamente pesquisada para a segurança de redes SCADA (BOŠTJANČIČ RAKAS; STOJANOVIĆ; MARKOVIĆ-PETROVIĆ, 2020; MEKALA *et al.*, 2023; RADOGLOU-GRAMMATIKIS *et al.*, 2020). De acordo com a literatura (MEKALA *et al.*, 2023; EVANGELIOU I., 2018; BOŠTJANČIČ RAKAS; STOJANOVIĆ; MARKOVIĆ-PETROVIĆ, 2020), os IDS são tipicamente classificados em duas categorias principais:

- **IDS Baseado em Assinatura:** Esta abordagem funciona de forma análoga a um software antivírus. Ele compara o tráfego capturado com um banco de dados de padrões (assinaturas) de ataques já conhecidos. Ferramentas como o Snort são um exemplo clássico desta técnica, porém possuem a limitação de não conseguir detectar ataques novos ou desconhecidos (ataques de "dia-zero") (EVANGELIOU I., 2018).
- **IDS Baseado em Anomalia:** Esta abordagem, que é o foco deste trabalho, primeiro estabelece uma "linha de base" do comportamento normal da rede e, em seguida, alerta sobre qualquer tráfego que se desvie significativamente desse padrão. Esta técnica é considerada particularmente eficaz para redes de Sistemas de Controle Industrial (ICS) e SCADA, pois o tráfego de controle industrial é, em geral, altamente previsível, repetitivo e determinístico, tornando qualquer desvio um forte indicador de anomalia ou ataque (RADOGLOU-GRAMMATIKIS *et al.*, 2020; SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025).

2.3.1 Sistemas de Detecção de Intrusão (IDS) e Linhas de Base (Baseline)

A detecção de anomalias apoia-se fortemente em uma característica inerente às redes operacionais (OT): o determinismo. Diferentemente das redes corporativas de TI, onde o comportamento dos usuários gera picos estocásticos de banda, as redes de automação industrial executam processos cíclicos de tempo real. Controladores Lógicos Programáveis (CLPs) realizam o *polling* (pesquisa) de sensores e atuadores em intervalos regulares e estritos de tempo (LI; QIN, 2017; SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025).

Essa previsibilidade permite o estabelecimento de uma Linha de Base (*Baseline*), um modelo matemático e estatístico rigoroso que define o comportamento "normal" e saudável da planta (SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025). A partir desse *baseline*, o IDS monitora a telemetria da rede continuamente. Métricas essenciais, como a taxa de chegada de pacotes (vazão) e o volume de dados, são calculadas, comumente através de algoritmos de médias móveis ou janelas deslizantes (*Sliding Windows*), para absorver flutuações naturais da rede (como *jitter*) (SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025). Qualquer desvio sustentado em relação a esse padrão — como o aumento exponencial de requisições oriundo de um ataque de Negação de Serviço (DoS) voltado ao esgotamento computacional do CLP (RAHMAN *et al.*, 2022) — quebra a fronteira do *baseline* e aciona um alarme de anomalia volumétrica.

2.3.2 Inspeção de Carga Útil Modbus contra Falsificação de IP

Embora o estabelecimento de um *baseline* volumétrico seja eficaz contra ataques de força bruta, ele é insuficiente para mitigar ameaças que injetam comandos cirúrgicos de baixa frequência. Uma prática comum em redes industriais legadas é a utilização de *Firewalls* de Camada 3 (Rede) ou *whitelists* no IDS, que avaliam apenas os endereços IP de origem e destino para autorizar a comunicação (EVANGELIOU I., 2018; PARIAN; GULDIMANN; BHATIA, 2020).

O equívoco desta estratégia reside no pressuposto de que o endereço IP é uma prova irrefutável de identidade. Como o protocolo Modbus TCP não possui mecanismos nativos de criptografia ou autenticação mútua (MEKALA *et al.*, 2023), um atacante pode empregar técnicas de Falsificação de Identidade (*IP Spoofing*). Nesse cenário, o invasor forja o cabeçalho IP dos pacotes que transmite, mascarando sua origem para que coincida com o IP de um CLP ou IHM presente na *whitelist*. Ao ser confrontado com um pacote falsificado, um *firewall* convencional de Camada 3 autorizará a passagem da ameaça, permitindo a passagem dos comandos dados pelo invasor.

Para neutralizar o tipo anterior de invasão e a falha da *whitelist*, a literatura define como indispensável o uso da Inspeção Profunda de Pacotes (*Deep Packet Inspection* - DPI) (PARIAN; GULDIMANN; BHATIA, 2020; SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025). Sistemas DPI operam na Camada 7 (Aplicação) do Modelo OSI. Em redes Modbus, um IDS equipado com DPI realiza o *parsing* do cabeçalho MBAP e da PDU. Ao abrir o pacote, o sistema analisa os Códigos de Função (*Function Codes*), identificando exatamente qual operação está sendo solicitada (FENG; LAI; LIU, 2020).

Essa granularidade permite a implementação de uma ***Whitelist* Comportamental**. Ao invés de uma permissão binária por endereço, o DPI impõe restrições contextuais, garantindo que mesmo um IP autêntico (mascarado por um invasor) não consiga executar comandos críticos que não correspondam à sua estrita função operacional na planta industrial, barrando a injeção de comandos não permitidos (SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025; PARIAN; GULDIMANN; BHATIA, 2020).

Para a construção e implementação prática dessas ferramentas customizadas de análise de rede e DPI, bibliotecas em linguagem Python destacam-se na literatura técnica. A biblioteca Scapy, por exemplo, permite capturar, analisar, decodificar e até forjar pacotes com baixo nível de detalhe (KOBAYASHI *et al.*, 2007; PARIAN; GULDIMANN; BHATIA, 2020). Embora seja amplamente utilizada para simular tráfego malicioso em testes de penetração (RAHMAN *et al.*, 2022; RODRÍGUEZ *et al.*, 2023; EKISA; Ó BRIAIN;

KAVANAGH, 2024), sua flexibilidade a torna uma base robusta para a prototipagem de motores de IDS defensivos. Similarmente, bibliotecas como a Pymodbus validam o uso de Python para interações industriais, sendo frequentemente empregadas na estruturação da comunicação Modbus em gateways e clientes de monitoramento (MAHMOOD; ALI, 2022; BORDIGNON, 2024).

2.4 Trabalhos Relacionados

Um trabalho de destaque é o de Kobayashi *et al.* (2007), que foi um dos primeiros a propor uma extensão da biblioteca Scapy para a análise de protocolos industriais, incluindo o Modbus. Os autores demonstraram o potencial da ferramenta para análise de segurança, e este trabalho baseou-se nessa premissa, aplicando a flexibilidade do Scapy não apenas para análise, mas para a construção de um sistema de detecção contínua de anomalias.

Diversos trabalhos subsequentes utilizaram o Scapy, mas focando primariamente no lado ofensivo da segurança, ou seja, para *implementar* ataques e validar vulnerabilidades. Parian, Guldimann e Bhatia (2020) e Rahman *et al.* (2022) utilizaram Scapy para criar e injetar pacotes Modbus maliciosos, simulando ataques de *Man-in-the-Middle* (MITM) e DoS em ambientes de teste. Da mesma forma, Rodríguez *et al.* (2023) desenvolveu o *toolkit* MOSTO, que utiliza Scapy para realizar auditorias de segurança e testes de penetração em dispositivos Modbus/TCP. O presente trabalho utilizou a mesma ferramenta, Scapy, mas com o propósito de detecção defensiva de tais ataques.

No campo específico de detecção de anomalias para Modbus, outros trabalhos relevantes propuseram soluções utilizando diferentes ferramentas e técnicas. Suchorab, Plamowski e Ławryńczuk (2025) apresentou um sistema de detecção de anomalias para dados Modbus baseado na ferramenta de monitoramento de rede Zeek e em cadeias de Markov para modelar o comportamento normal do tráfego. Similarmente, Radoglou-Grammatikis *et al.* (2020) propôs um IDS capaz de analisar e classificar o tráfego Modbus TCP contra ataques de Negação de Serviço. Estes trabalhos validaram a abordagem de detecção de anomalias, e o presente estudo diferenciou-se deles pela arquitetura de implementação, que utilizou Scapy em Python para a captura e decodificação, oferecendo uma solução potencialmente mais leve e customizável, e pela integração com uma plataforma de nuvem MQTT para alertas.

Outros pesquisadores focaram na criação de *testbeds* para simular redes industriais e avaliar tanto ataques quanto defesas (D'AMBROSIO *et al.*, 2025; PLAMOWSKI *et al.*, 2024; EKISA; Ó BRIAIN; KAVANAGH, 2024). Este trabalho beneficiou-se desses estudos ao adotar uma arquitetura de teste similar, composta por elementos reais (CLP físico) e simulados (FlexFact).

Finalmente, trabalhos como o de Bordignon (2024) exploram a integração de redes Modbus com a nuvem através de um *gateway* IIoT, utilizando protocolos como MQTT para a transmissão de dados. Embora o objetivo de Bordignon (2024) fosse o monitoramento de processos (coleta de dados para a nuvem), e não a segurança, ele validou a arquitetura de comunicação com a nuvem (MQTT e Node-RED) que este trabalho utilizou para visualização dos alertas de segurança gerados pelo *sniffer*.

Na Tabela 1 é apresentada uma relação entre os trabalhos relacionados e uma comparação com o escopo do presente trabalho.

Tabela 1: *Síntese dos Trabalhos Relacionados e Comparação com a Solução Proposta.*

Trabalho	**Foco**	**Ferramentas / Métodos**	**Relação com este Trabalho**
Kobayashi *et al.* (2007)	Análise	Scapy, dissecção Modbus	Valida o uso do Scapy na decodificação do protocolo.
Parian, Guldimann e Bhatia (2020)	Ofensivo	Scapy, IP Spoofing, MITM	Demonstra a necessidade de inspeção na camada de aplicação.
Rahman *et al.* (2022)	Ofensivo	DoS, Injeção	Fundamenta os ataques volumétricos testados neste projeto.
Rodríguez *et al.* (2023)	Ofensivo	Toolkit MOSTO, Scapy	Utiliza o Scapy para implementar um toolkit chamado MOSTO.
Suchorab, Plamowski e Ławryńczuk (2025)	Defensivo	Zeek, Cadeias de Markov	Corrobora a eficácia de detecção de anomalias, difere pela complexidade da ferramenta (Zeek).
Radoglou-Grammatikis *et al.* (2020)	Defensivo	Smod Toolkit, IDS por anomalia, DoS	Embaseia a detecção de ataques volumétricos, difere pela arquitetura de alertas.
Bordignon (2024)	Monitoramento	Gateway IIoT, MQTT	Valida a arquitetura de comunicação e uso do Node-RED/MQTT para visualização.
Este Trabalho	**Monitoramento**	**Scapy, MQTT, Node-RED**	**Une o Scapy para inspeção de pacotes com Node-RED/MQTT para alertas e monitoramento de baixo custo.**

Fonte: Elaborado pelo autor.

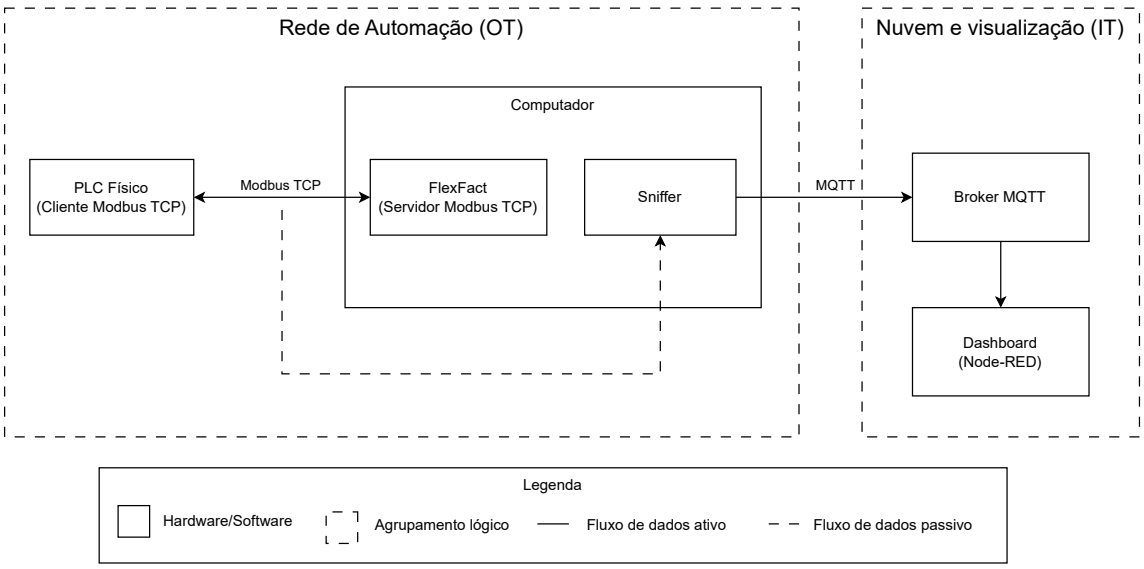
3 METODOLOGIA

Este trabalho baseou-se no desenvolvimento e validação de um protótipo em um ambiente de teste que combinou equipamentos reais e virtuais. Essa abordagem é uma prática validada na literatura para a pesquisa em segurança de Sistemas de Controle Industrial (ICS), pois permite a simulação e análise de ameaças sem colocar em risco um sistema de produção real (EKISA; Ó BRIAIN; KAVANAGH, 2024; D'AMBROSIO *et al.*, 2025; PLAMOWSKI *et al.*, 2024; RADOGLOU-GRAMMATIKIS *et al.*, 2020).

3.1 Arquitetura da Solução

A arquitetura do protótipo foi projetada para ser não-invasiva e está dividida em dois domínios funcionais principais, conforme ilustrado na Figura 2: (1) a Rede de Automação (OT), onde a comunicação de controle ocorre; e (2) a Plataforma de Nuvem (IT), para onde os alertas de segurança são enviados.

Figura 2: *Arquitetura da solução e fluxo de dados do protótipo.*



Fonte: Elaborado pelo autor.

3.1.1 Domínio da Rede de Automação (OT)

Este domínio representa a rede industrial de chão de fábrica onde o processo é controlado e monitorado. Neste trabalho, ela é composta por dois elementos principais conectados diretamente por um cabo Ethernet:

- **Controlador Lógico Programável (CLP) Físico:** Atua como o Cliente Modbus TCP (mestre). O CLP foi programado para realizar requisições cíclicas de leitura e escrita de dados, simulando o comportamento de um controlador real que monitora e comanda um processo industrial.
- **Computador (PC da Simulação e Análise):** Este computador hospeda dois componentes de software cruciais que operam em paralelo:
  - O **Software FlexFact**, que atua como o Servidor Modbus TCP (escravo). O FlexFact simula a planta industrial, recebendo os comandos do CLP e respondendo às suas requisições na porta TCP 1502.
  - O **Software IDS (Sniffer)**, o protótipo central deste trabalho. Foi desenvolvido em Python com o uso das bibliotecas Scapy para a captura e analise de pacotes, além da biblioteca Paho-MQTT para externalizar os alertas de rede.

O FlexFact (FG DES, 2025) é um software que simula o comportamento dinâmico de componentes eletromecânicos para formar um sistema de manufatura flexível. A sua relevância para este trabalho reside no fato de que, apesar de ser um ambiente virtual, ele foi projetado para que a simulação se comporte como o sistema físico respectivo. Para a interconexão com o controlador (o CLP), o FlexFact implementa uma interface "Slave" (servidor) Modbus/TCP, permitindo o acesso direto ao processo através do protocolo Modbus padrão. Além disso, o FlexFact permite que o usuário projete seus próprios layouts de fábrica, escolhendo e posicionando diferentes componentes, possibilitando a criação de múltiplos estudos de caso para testar o *sniffer*, desde os cenários operacionais mais simples até os mais complexos.

O fluxo de dados de controle, representado no diagrama (Figura 2) pela linha sólida bidirecional, consiste na comunicação de requisição-resposta do protocolo Modbus TCP entre o CLP Físico e o Software FlexFact. Paralelamente, o *Sniffer* opera de forma passiva, "escutando" a interface de rede física do computador e, conforme representado pela linha tracejada, copia os pacotes Modbus TCP trocados entre o CLP e o FlexFact.

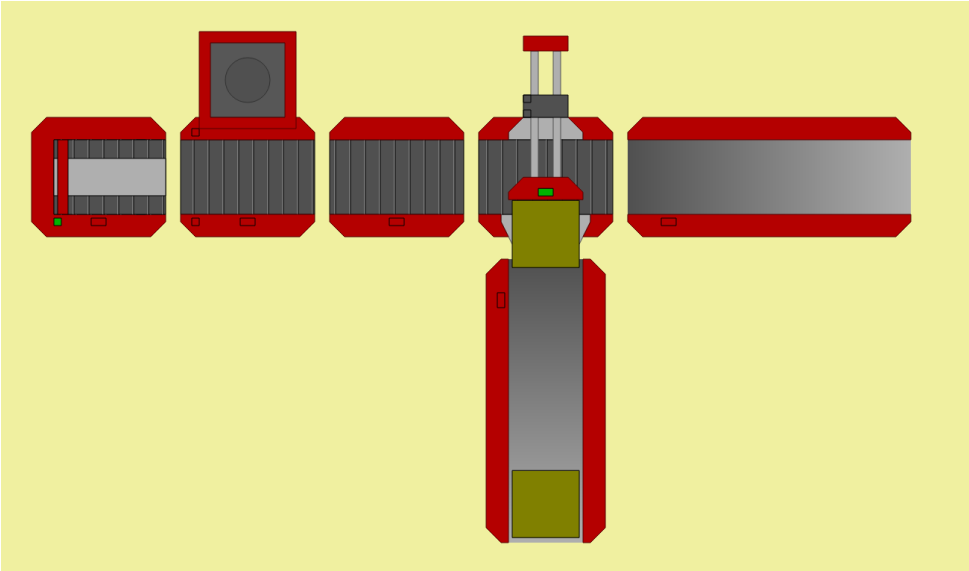
Cenário de Simulação e Lógica de Controle

Para a validação do protótipo, configurou-se no simulador FlexFact uma planta industrial composta por um alimentador de peças (*Stack Feeder*), uma máquina de processamento (*Processing Machine*), um sistema de distribuição (*Distribution System*) e duas rampas de saída (*Exit Slides*). A disposição física desses componentes no simulador pode ser visualizada na Figura 3.

A lógica de simulação foi estruturada da seguinte forma:

- **Fluxo de Processo Inicial:** A operação é iniciada através do *Stack Feeder*, onde novos itens são inseridos na planta mediante o clique do usuário. O produto é então encaminhado pela esteira principal até a *Processing Machine*.

Figura 3: *Layout da planta industrial simulada no software FlexFact.*



Fonte: Elaborado pelo autor.

- **Processamento (Máquina de Estados):** O controle da *Processing Machine* foi estruturado através de uma máquina de estados desenvolvida no ambiente Studio 5000. Ao detectar a chegada do objeto na estação através do sensor de presença, o CLP emite comandos Modbus TCP sequenciais para: (1) interromper o acionamento da esteira, (2) posicionar o atuador de processamento sobre a peça, (3) aguardar o tempo de execução do processo simulado, (4) retrair o mecanismo para a posição de repouso e, por fim, (5) reativar o motor da esteira para a liberação do item.

- **Controle e Roteamento via Modbus TCP:** Após o processamento, o item chega ao *Distribution System*. Neste ponto, o CLP físico realiza leituras para detectar a presença do item através dos sensores do FlexFact. Ao receber o sinal, o CLP executa a lógica e envia um comando de escrita via Modbus TCP, instruindo o sistema a acionar o cilindro pneumático para transferir o produto para a primeira linha de saída.

- **Validação de Latência (Rampas de Saída):** Para validação de reatividade da rede e do processo, o controle é testado pelas duas rampas de saída (*Exit Slides*). Se a comunicação de rede e o tempo de varredura do CLP fossem rápidos o suficiente, o produto era empurrado com sucesso para o primeiro *Exit Slide*. Contudo, caso a rede sofresse atrasos (impedindo a troca de pacotes Modbus em tempo hábil), o cilindro não atuaria a tempo e o produto passaria reto para o segundo *Exit Slide*, caracterizando uma falha de processo induzida por perda de determinismo na rede.

Essa arquitetura permitiu que, em condições de operação, o volume de pacotes na rede fosse constante e o tempo de resposta perfeitamente ajustado ao processo. Desvios no tempo de ciclo, polling rate ou latência podem gerar um “descoordenamento” na simulação ao mesmo tempo que é detectado pelo IDS.

Especificações de Hardware e Infraestrutura de Rede

A topologia física foi montada sob a premissa de um *testbed* de baixo custo para que seja possível a reprodução dos experimentos. O hardware central utilizado como mestre da rede foi um controlador da família **Allen-Bradley CompactLogix** (Série 5370), fabricado pela Rockwell Automation. Este equipamento foi selecionado por sua ampla adoção no cenário industrial global e por possuir uma pilha de protocolos TCP/IP nativa. A lógica de controle e a configuração das mensagens (*MSG instructions*) para o *polling* cíclico foram desenvolvidas no ambiente de programação **Studio 5000 Logix Designer** e programadas em Ladder e ST (Structured Text).

A conexão física entre o CLP e o computador de simulação (hospedeiro do FlexFact e do IDS) foi realizada utilizando as portas LAN traseiras de um roteador comercial. Neste arranjo, o roteador operou estritamente na Camada de Enlace (Camada 2 do Modelo OSI), desempenhando o papel de um *switch* não gerenciável. Os ativos foram configurados na mesma sub-rede, com o CLP recebendo o IP estático `192.168.0.50` e o computador alvo o IP `192.168.0.10`.

Esta arquitetura eliminou a necessidade de dispositivos intermediários de interceptação de tráfego, tais como *Network Taps* ou *switches* gerenciáveis com suporte a espelhamento de portas (*Port Mirroring*/SPAN). Como o computador era o destino final das requisições do CLP na porta TCP 1502, os pacotes trafegavam nativamente pelo computador, e o *sniffer* em Python funcionou mapeando tanto a interface física quanto a virtual de *Loopback* (`127.0.0.1`) para ter acesso total ao tráfego da rede.

3.1.2 Domínio da Nuvem e Visualização (IT)

Este domínio representa a camada superior da arquitetura, responsável por centralizar as informações extraídas do chão de fábrica e disponibilizá-las para a gestão. Para garantir que o monitoramento não interferisse no determinismo da rede de controle (OT), adotou-se o paradigma de comunicação de publicação e assinatura (*Publish/Subscriber*), atuando como uma ponte unidirecional para a rede externa.

A arquitetura lógica deste domínio foi elaborada de forma a segmentar os tráfegos de dados em duas vias independentes:

1. **Canal Contínuo de Vazão:** Uma rota para receber dados periódicos e determinísticos sobre o volume de tráfego da rede, permitindo o monitoramento da integridade da comunicação.

2. **Canal Assíncrono de Incidentes:** Uma rota prioritária e silenciosa, ativada exclusivamente quando o motor de regras detecta uma violação, projetada para transportar os alertas com suas informações detalhadas.

O protocolo escolhido para viabilizar essa arquitetura foi o MQTT (*Message Queuing Telemetry Transport*), justificado por sua alta escalabilidade e baixo *overhead*, sendo um padrão consolidado na literatura para a integração de sistemas ciber-físicos à nuvem (BORDIGNON, 2024; NGUYEN-HOANG; VO-TAN, 2019; TIETZ; BRANDÃO; ALVES, 2018). A interface final de visualização foi designada para ser hospedada na plataforma Node-RED, atuando como o ponto final de consumo desses dados.

3.2 Desenvolvimento do Sniffer

O software IDS/*Sniffer* foi desenvolvido em linguagem Python[1], utilizando uma abordagem modularizada para garantir a eficiência no tratamento de pacotes e facilitar a validação independente de cada componente lógico. O sistema operou de forma assíncrona, estruturado em módulos de captura, decodificação, análise estatística/DPI e comunicação, conforme detalhado a seguir.

3.2.1 Captura e Filtragem

O primeiro módulo consistiu na captura e monitoramento de forma contínua das interfaces de rede do computador. Para esta tarefa, utilizou-se a biblioteca Scapy. No quesito utilidade, o Scapy atuou como uma interface de alto nível para bibliotecas de captura nativas do sistema operacional, que trabalham em baixo nível (kernel) para realizar o monitoramento da rede. No caso do ambiente Windows utilizado, a arquitetura apoiou-se no driver **Npcap** para acessar a placa de rede.

Para otimizar o desempenho do código Python e capturar diretamente os pacotes que eram desejados, aplicou-se um filtro de captura na biblioteca base utilizando a sintaxe *Berkeley Packet Filter* (BPF). O filtro foi configurado como `tcp dst port 1502`, instruindo o driver a encaminhar ao Scapy exclusivamente pacotes TCP direcionados à porta de testes do Modbus. Desta forma, pacotes irrelevantes foram descartados antes mesmo de entrar no escopo do código Python.

É fundamental ponderar, sob a ótica de projeto, as limitações de vazão dessa ferramenta. Por ser executado no espaço de usuário e de forma interpretada, o Scapy possui uma latência de processamento inerente. Em redes industriais de grande porte com altíssima vazão efetiva de dados, essa arquitetura poderia resultar no estouro dos *buffers* e consequentemente em perdas de pacotes. Contudo, a adoção da ferramenta justificou-se pela natureza determinística e de baixa vazão nominal da célula de automação avaliada ($\lambda \approx 14$ pkt/s), além de aplicações em que equipamentos fornecidos para uma planta industrial incluem um PLC standalone para controle do equipamento, que deve trocar alguns dados com o PLC de controle da planta. Neste regime controlado, o filtro BPF em nível de kernel provou-se capaz de mitigar perdas, viabilizando o monitoramento íntegro da camada de aplicação.

3.2.2 Decodificação (Parser)

Uma vez que um pacote é capturado pelo filtro, seu conteúdo precisa ser decodificado. Utilizando as propriedades de inspeção de camadas do Scapy, incluindo o Modbus (conforme demonstrado em Kobayashi *et al.* (2007)), foi possível extrair os campos críticos do protocolo diretamente do campo de carga útil (*payload*) de dados brutos (*Raw Layer*), condicionados a um comprimento mínimo de 8 bytes para garantir que o cabeçalho MBAP está completo.

Assim, o Scapy disponibilizou diretamente os seguintes itens do objeto:

[1]Todo o código desenvolvido, incluindo o sniffer (`sniffer_ids.py`), script de teste dos ataques (`pentest.py`), fluxos do Node-RED, lógicas de CLP e planta do FlexFact estão disponíveis no repositório ModbusTCP-IDS no GitHub: `https://github.com/Therocyn/ModbusTCP-IDS`

- **Cabeçalho TCP/IP:** O IP de origem e o IP de destino.
- **Cabeçalho MBAP:** O Identificador de Transação e o Identificador de Unidade.
- **PDU:** O Código de Função e os dados específicos da função.

3.2.3 Análise e Detecção de Anomalias

A concepção preliminar deste trabalho previa a implementação de métricas tradicionais de segurança baseadas exclusivamente em cabeçalhos de rede. Contudo, durante a fase de desenvolvimento e alinhamento com a arquitetura descrita por Parian, Guldimann e Bhatia (2020), constatou-se que o protocolo Modbus TCP carece nativamente de autenticação e integridade, o que exige um monitoramento mais profundo. Para mitigar essas falhas e adequar o sistema às exigências de detecção de anomalias (SUCHORAB; PLAMOWSKI; ŁAWRYŃCZUK, 2025), o IDS foi estruturado com três regras principais operando em paralelo, e a Tabela 2 resume a estratégia de detecção, correlacionando cada regra à camada do modelo OSI, ao seu objetivo de mitigação e à vulnerabilidade mitigada:

Tabela 2: *Resumo das regras de detecção implementadas no IDS e vulnerabilidades mitigadas.*

Regra	**Camada OSI**	**Objetivo Prático**	**Vulnerabilidade Mitigada**
1. Volumétrica	Camada 4 (Transporte)	Detecção de Ataques DoS	Perda de Disponibilidade e Falta de Autenticação
2. Acesso IP	Camada 3 (Rede)	Detecção de Origens não Autorizadas	Falta de Autenticação
3. Inspeção de Carga Útil (FC)	Camada 7 (Aplicação)	Detecção de Comandos Não Permitidos e *IP Spoofing*	Falta de Integridade

Abaixo, detalha-se a fundamentação técnica de cada regra:

Regra 1: Detecção de Anomalia Volumétrica. Frente à suscetibilidade de equipamentos legados a inundações de pacotes (RAHMAN *et al.*, 2022), implementou-se um algoritmo matemático para a detecção de ataques de Negação de Serviço (DoS). Em vez de utilizar contadores estáticos, adotou-se o modelo de Janela Deslizante contínua com intervalo de um segundo ($W = 1.0$s), priorizando a detecção quase instantânea exigidas em ambientes de Tecnologias de Operação (OT). Sabendo que o código Modbus do Controlador Lógico Programável (CLP) físico operou com um intervalo de *polling rate* de 80 ms, a vazão determinística nominal aferida empiricamente na rede resulta em aproximadamente $\lambda \approx 14$ pkt/s, valor este que já absorve o *overhead* da pilha TCP/IP e a latência de processamento do escravo. O IDS calcula a taxa de vazão no último segundo, sendo o limiar de detecção (τ) calibrado para $25,0$ pkt/s. Esta margem de tolerância em relação à vazão nominal da rede (≈ 14 pkt/s) foi desenhada para absorver variações legítimas como *jitter*, sistemas de polling assíncrono, e leituras em rajada acionadas por sistemas SCADA sem acionar falsos positivos frequentes. Além disso, para mitigar falsos positivos gerados por retransmissões ou pacotes de controle de sessão TCP (como `SYN` ou `ACK`), o laço de captura é condicionado a inspecionar exclusivamente pacotes que contenham a carga útil Modbus (mínimo de 8 bytes).

Regra 2: Controle de Acesso por IP. Para que apenas dispositivos autorizados interajam ativamente com o simulador da planta, o sistema implementa uma verificação de endereços de origem para operações críticas. A regra estabelece o IP do CLP físico (`192.168.0.50`) como a única fonte confiável para o envio de comandos de alteração de estado. Pacotes originados de endereços IP estranhos ou desconhecidos que tentem interagir com os atuadores da planta são imediatamente detectados e classificados como intrusão externa. Esta barreira protege a rede operacional contra dispositivos não autorizados conectados indevidamente ao ambiente OT.

Regra 3: Inspeção de Carga Útil para Function Code. Como o protocolo Modbus é vulnerável à falsificação de identidade (*IP Spoofing*), um atacante poderia burlar a Regra 2 simplesmente mascarando seu endereço IP para se passar pelo CLP. Para capturar esse tipo de ataque, o IDS possui uma regra de Inspeção da Carga Útil Modbus na Camada 7 do modelo OSI, focada nos *Function Codes*. O analisador aplica uma *Whitelist Comportamental*, onde a operação de escrita de múltiplos *coils* (Código 15) é a única ação de alteração de estado permitida. Qualquer outro código de escrita (como Códigos 05, 06 ou 16) injetado na rede é detectado pelo *sniffer*, mesmo que origine do IP legítimo do CLP. Essa inspeção de comandos garante a detecção tanto de funções estranhas ao processo quanto de atacantes camuflados.

Ressalta-se que os comandos de leitura (Códigos 01, 02, 03 e 04) não geram alertas de segurança neste protótipo. Esta decisão está de acordo com as diretrizes de gerenciamento de risco em Tecnologias de Operação (OT), embasadas na norma ISA/IEC 62443 e corroboradas pela literatura recente (CINDRIĆ; JURČEVIĆ; HADJINA, 2025; DA SILVA *et al.*, 2025). Conforme discutido na fundamentação deste trabalho, em ambientes industriais a prioridade recai sobre a alta disponibilidade, a integridade do processo e a segurança física, mesmo que signifique detrimento da confidencialidade. Basicamente, uma requisição de leitura não autorizada (como consultar a temperatura de uma caldeira ou o nível de um tanque) compromete o sigilo da informação, mas não afeta a dinâmica da planta, enquanto a injeção de um comando de escrita (como acionar um motor ou fechar uma válvula crítica) interage diretamente com o processo físico, podendo provocar grandes incidentes. Dessa forma, ao focar a geração de alertas exclusivamente em comandos de escrita, o IDS prioriza eventos que apresentem risco operacional real ao processo, ao mesmo tempo em que é evitado o excesso de notificações irrelevantes.

3.2.4 Comunicação com a Nuvem

Quando o módulo de análise (Seção 3.2.3) detecta uma anomalia, ele deve registrá-la e notificar um sistema centralizado. Para isso, foi utilizado o protocolo MQTT, seguindo a arquitetura de integração IIoT validada em trabalhos como (BORDIGNON, 2024).

A implementação de software apoiou-se na biblioteca `paho-mqtt` para Python, responsável por encapsular as informações e publicar os dados na porta padrão 1883 usando uma estrutura baseada em JSON. O script foi estruturado para manter uma sessão no *broker* público HiveMQ (`broker.hivemq.com`), usando o método `loop_start()`.

A lógica de transmissão foi dividida em dois fluxos de dados, funcionando de formas distintas:

1. **Publicação Periódica da Vazão de pacotes:** A cada intervalo de 1 segundo, controlado pela variável `INTERVALO_TELEMETRIA`, o IDS calcula a taxa instantânea

de pacotes processados na janela móvel. Esse indicador é publicado diretamente no tópico `ufrgs/tcc/vitor/ids/vazao`, servindo como alimentação contínua para o monitor de estado da rede.

2. **Disparo Assíncrono de Eventos/Alertas:** Mediante a ocorrência de um ataque, a rotina `publicar_alerta()` é acionada de forma imediata. Esse método gera um dicionário contendo as chaves estruturadas: `timestamp`, `tipo_alerta`, `ip_origem`, `gravidade` e `detalhes`. Esta estrutura é serializada em uma string através do método `json.dumps()` e publicada no tópico `ufrgs/tcc/vitor/ids/alertas`.

Essa abordagem desacoplacada facilitou a organização e diminuiu a complexidade do código, ao mesmo tempo em que ajudou a otimizar o consumo de banda de rede externa sem acrescentar atrasos ou retenção de memória ao ciclo de captura do *sniffer*.

3.2.5 Visualização de dados

O desenvolvimento da interface de monitoramento foi integralmente construído na plataforma Node-RED. O ambiente foi configurado com um fluxo de nós lógicos projetado para atuar como cliente MQTT ouvinte (*subscriber*).

A topologia de blocos implementada, visível na Figura 4, operou da seguinte forma:

- Dois nós do tipo "MQTT In" foram configurados para escutar de forma simultânea os tópicos de `vazao` e `alertas` no *broker* HiveMQ.
- O nó atrelado ao fluxo de vazão encaminhou o dado numérico bruto diretamente para um elemento gráfico do pacote *Node-RED Dashboard 2.0*, configurado como um medidor do estilo Gauge.
- O nó atrelado ao fluxo de alertas encaminhou a *string* recebida para um bloco conversor "JSON", que extraiu os atributos do ataque para popular um log de alertas e disparar caixas de notificação na tela para chamar atenção do usuário.

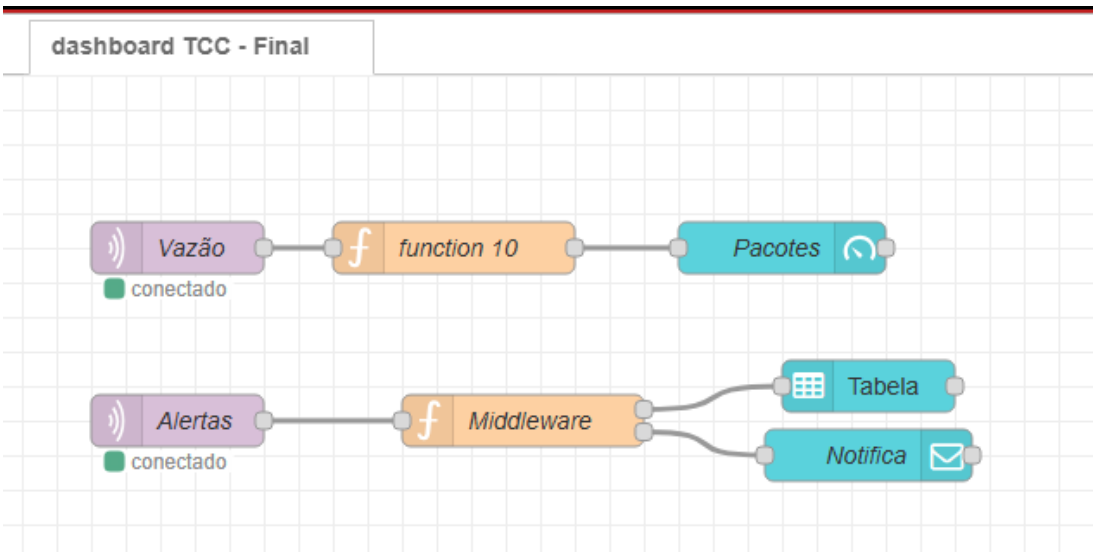
O *dashboard* resultante desta integração pode ser visualizado na Figura 5, fornecendo ciência visual ao usuário ao passo que incidentes ocorrem.

3.3 Metodologia de Validação e Testes

Para atestar a robustez das três regras lógicas desenvolvidas para o protótipo, estruturou-se uma metodologia de validação baseada em cenários de testes práticos. Os ataques foram gerados a partir de um nó secundário na rede usando 2 bibliotecas em Python: a biblioteca **Pymodbus** foi utilizada para simular o comportamento de um atacante na camada de aplicação, gerando os ataques DoS e as injeções de comandos sob acesso não autorizado; enquanto a biblioteca **Scapy** foi empregada para a manipulação artesanal de cabeçalhos de rede e forja de pacotes, viabilizando o ataque de *IP Spoofing*.

- **Teste 1: Acesso Não Autorizado (Validação da Regra 2).**
  - *Ação:* O computador atacante, operando com um IP não autorizado, gerou e enviou requisições de escrita Modbus destinadas aos atuadores do FlexFact.

Figura 4: *Fluxo de nós implementado no Node-RED.*



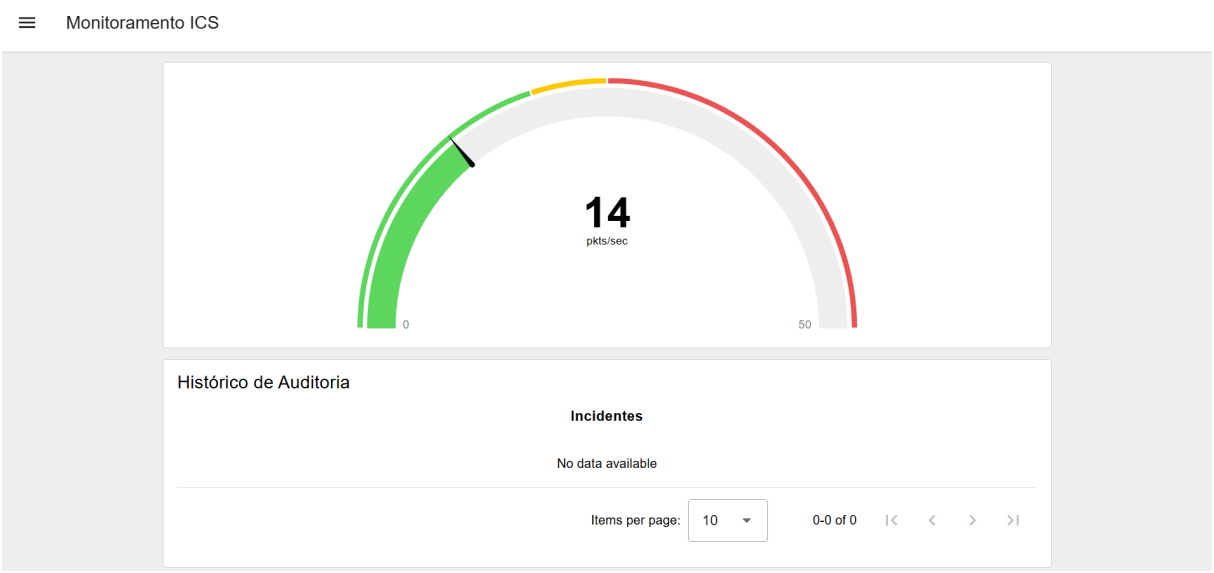
Fonte: Elaborado pelo autor.

  - *Verificação:* Avaliou-se se o *sniffer* detectou o comando originado do IP suspeito e se o alerta correspondente foi gerado no *dashboard* via MQTT.

- **Teste 2: Injeção de Comandos Não Autorizados / IP Spoofing (Validação da Regra 3).**
  - *Ação:* O atacante mascarou seu endereço IP (*IP Spoofing*) para se passar pelo CLP legítimo (`192.168.0.50`) e enviou comandos de escrita válidos no protocolo Modbus, porém utilizando Function Codes que violam a lógica da planta (como o Código 05 ou 06).
  - *Verificação:* Validou-se a capacidade da Inspeção de Carga Útil Modbus em ignorar a aparente legitimidade do IP, identificar a violação comportamental na Camada 7 e alertar sobre a injeção de Function Codes não autorizados.

- **Teste 3: Negação de Serviço - DoS (Validação da Regra 1).**
  - *Ação:* O atacante utilizou rotinas de disparo em alta velocidade para enviar uma inundação de pacotes Modbus contínuos contra a porta 1502 do simulador.
  - *Verificação:* Monitorou-se a efetividade da Janela Deslizante em detectar a violação do limiar de 25 pkt/s, emitindo o alerta volumétrico enquanto se avaliava fisicamente o impacto da latência na rede atrasando o funcionamento do *Distribution System* no simulador.

Figura 5: *Interface do Dashboard desenvolvido no Node-RED.*



Fonte: Elaborado pelo autor.

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados obtidos durante a execução dos cenários de teste propostos na Seção 3.3. A validação fundamentou-se na análise dos registros de rede gerados pelo *sniffer* e na respectiva externalização dos alertas de segurança via protocolo MQTT para a interface visual hospedada no Node-RED.

4.1 Validação da Linha de Base

A primeira etapa de validação consistiu em atestar a estabilidade do monitoramento sob condições nominais de operação da planta FlexFact, garantindo a premissa de não-invasividade. Durante os ciclos regulares do processo (onde o Controlador Lógico Programável físico executava o *polling* periódico e atuava no *Distribution System*), o *sniffer* operou passivamente na interface de rede.

A lógica de telemetria contínua processou a janela deslizante de 1 segundo, publicando a vazão no tópico `ufrgs/tcc/vitor/ids/vazao`. O *dashboard* Node-RED decodificou os dados e atualizou o medidor conforme os dados eram recebidos. A vazão manteve-se estável em aproximadamente 14 pkt/s, situando-se perfeitamente dentro da zona segura configurada na interface gráfica (cor verde, faixa de 0 a 20 pkt/s).

4.2 Avaliação do Ataque de Negação de Serviço (DoS)

O primeiro cenário de ataque validou a detecção de anomalias volumétricas (Regra 1). Ao acionar a rotina `ataque_dos()` na ferramenta de auditoria, o sistema atacante estabeleceu uma conexão TCP com o servidor FlexFact, inundando a rede com requisições Modbus do tipo *Read Holding Registers* de forma contínua.

O motor do Sistema de Detecção de Intrusão (IDS) calculou imediatamente o acúmulo de pacotes pelo algoritmo de janela deslizante. Assim que a equação da taxa instantânea atingiu o teto tolerável configurado de 25,0 pkt/s, o sistema publicou um incidente no tópico de alertas classificando o evento como "Ataque de Negação de Serviço (DoS)". No *dashboard* Node-RED, essa violação elevou instantaneamente o medidor para a zona vermelha e registrou a ocorrência na tabela de histórico de auditoria, conforme demonstrado na Figura 6.

Apesar da detecção lógica e precisa da anomalia de rede pelo IDS, a observação física da simulação revelou que a injeção volumétrica não causou latência perceptível no ambiente de automação. A comunicação determinística entre o Controlador Lógico Programável (CLP) e o simulador manteve-se estável, e os atuadores, como o cilindro

Figura 6: *Dashboard do Node-RED registrando o ataque de DoS.*



Fonte: Elaborado pelo autor.

pneumático do *Distribution System*, operaram de forma síncrona e sem falhas, roteando as peças corretamente.

Este comportamento pode ser atribuído à discrepância entre a capacidade de processamento da rede e o método de injeção da ameaça. O *script* ofensivo, limitado por execuções *single-thread* e pela natureza síncrona da biblioteca `pymodbus`, gerou uma carga de aplicação insuficiente para causar contenção na camada de enlace do *switch* ou para esgotar o *stack* TCP/IP do computador hospedeiro, onde foi verificado o isolamento das sessões do CLP e do atacante em conexões de *sockets* distintas. Contudo, o cenário cumpre seu papel ao atestar a eficácia preventiva do IDS com o sistema provando-se capaz de identificar o desvio na linha de base da rede conforme o aumento na vazão de pacotes.

Por fim, ao avaliar a resiliência da própria ferramenta de monitoramento, cabe ressaltar que o sniffer sobreviveu e detectou o ataque neste ensaio porque a vazão ofensiva, limitada pelo script síncrono, não excedeu o poder de processamento da máquina. Contudo, em um cenário de ataque DoS volumétrico extremo (na escala de Gigabits), a arquitetura do sniffer operando em espaço de usuário (user-space) via Python também seria afetada, podendo sofrer estouro de buffers na placa de rede e perda de pacotes, comprometendo assim sua capacidade de analisar o tráfego de forma integral.

4.3 Detecção de Acesso Não Autorizado

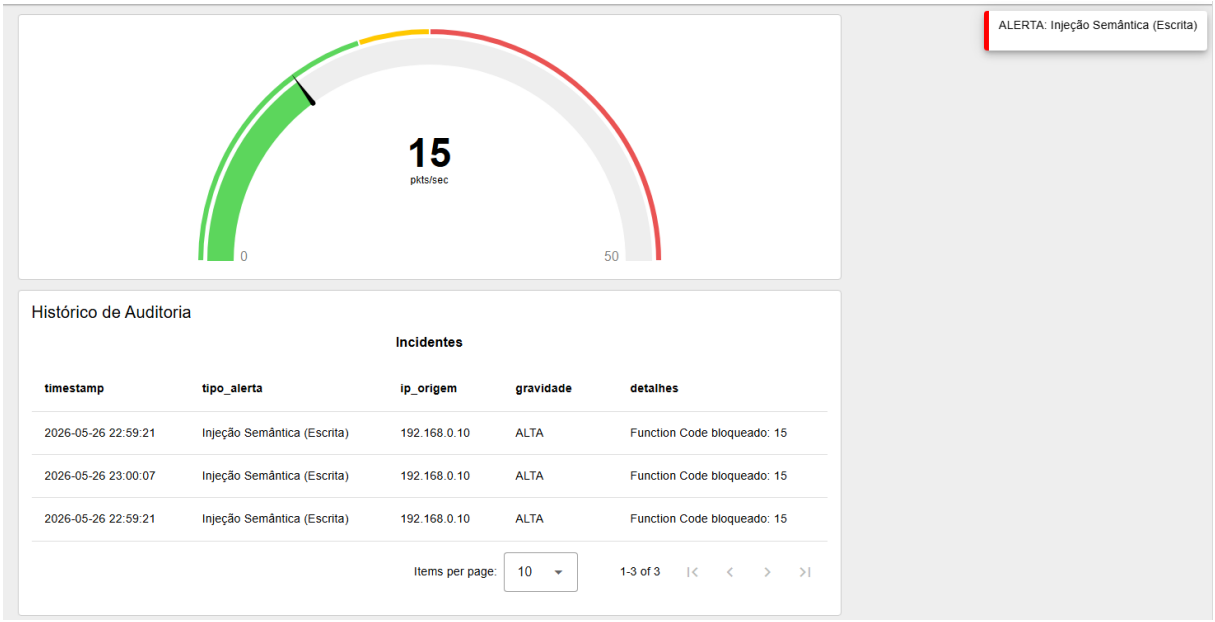
No segundo cenário de testes, o objetivo foi avaliar a eficácia do sistema em capturar comandos invasivos originados de um dispositivo não autorizado. Utilizando a ferramenta de auditoria, simulou-se um atacante operando com um endereço IP distinto do PLC oficial (por exemplo, `192.168.0.10`), que injetou um comando Modbus reconhecido como padrão operacional da rede *Write Multiple Coils* (FC 15) diretamente no servidor FlexFact para manipular o estado de um atuador.

O *sniffer* operou conforme o esperado pela arquitetura delineada na Metodologia. O filtro BPF encaminhou o tráfego à rotina de análise, que executou o *parsing* da camada de

aplicação. A lógica do motor de regras identificou imediatamente a anomalia: a origem do pacote não pertencia à *whitelist* do endereço IP do Controlador Lógico Programável (CLP) legítimo (`192.168.0.50`), mesmo que o comando injetado (FC 15) fosse o comportamento padrão de comunicação na rede.

Como resultado, o evento foi instantaneamente reportado via MQTT. A Figura 7 exibe o *dashboard* Node-RED registrando a notificação de "Injeção Semântica (Escrita)", sinalizando a origem do IP invasor e mostrando o *Function Code* utilizado.

Figura 7: *Dashboard do Node-RED alertando o comando de escrita por IP não autorizado.*



Fonte: Elaborado pelo autor.

4.4 Análise da Detecção de Comandos Não Autorizados e IP Spoofing

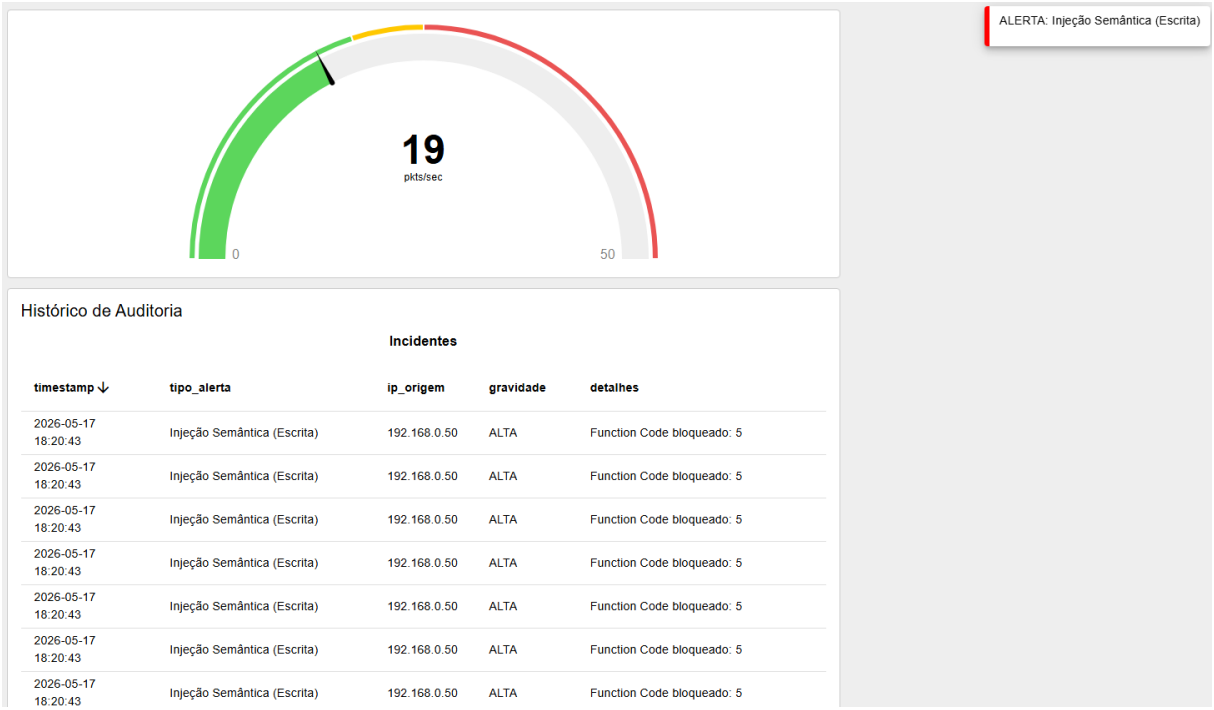
O terceiro tipo de cenário simulou um atacante buscando burlar a detecção por meio da utilização de um IP já conhecido na rede. O *script* ofensivo construiu artesanalmente pacotes nas camadas do modelo OSI, falsificando o IP de origem para coincidir exatamente com o IP do CLP físico legítimo (`192.168.0.50`) e encapsulando o comando de *Write Single Coil* (FC 05).

Caso a defesa estivesse restrita a um *firewall* convencional de rede, o pacote mascarado teria sido validado como tráfego confiável e a intrusão passaria despercebida. Contudo, a aplicação da Regra 3 de Inspeção de Carga Útil Modbus comprovou-se fundamental. O módulo de análise do IDS inspecionou a PDU do Modbus TCP e aplicou a *whitelist* comportamental na Camada 7, a qual estipula que o IP do CLP só tem permissão para realizar operações de escrita sob o Function Code `15`.

Ao comparar o código `05` extraído do pacote falsificado com a lista de códigos permitidos, o sistema detectou a injeção de um FC não autorizado e ignorou a aparente legitimidade do IP. A Figura 8 ilustra o ambiente de monitoramento capturando o incidente,

evidenciando que a ferramenta foi capaz de diagnosticar a tentativa de *spoofing* e alertar o usuário.

Figura 8: *Dashboard do Node-RED registrando a injeção de comando não autorizado com IP Spoofing.*



Fonte: Elaborado pelo autor.

5 CONCLUSÕES

O presente trabalho concentrou-se no desafio técnico de prover maiores níveis de segurança a redes de automação baseadas no protocolo Modbus TCP, um padrão industrial amplamente difundido, porém carente de mecanismos nativos de proteção. A convergência entre ambientes de IT e OT, característica da Indústria 4.0, tornou esses sistemas legados vulneráveis a ameaças cibernéticas que, se não mitigadas, podem comprometer a disponibilidade e a segurança física dos processos produtivos.

Para responder a essa problemática, foi desenvolvido e validado um protótipo de Sistema de Detecção de Intrusão (IDS) não-invasivo. Através da análise de tráfego passiva e inspeção da carga útil dos pacotes Modbus, demonstrou-se ser possível implementar uma camada de auditoria capaz de identificar anomalias volumétricas e injeções de comandos não permitidos, mantendo a integridade e o determinismo essenciais ao chão de fábrica.

Este capítulo final consolida a análise dos resultados obtidos ao longo do desenvolvimento, detalha as contribuições do trabalho, aponta as limitações observadas durante a experimentação e propõe perspectivas para desenvolvimentos futuros na área.

5.1 Considerações Finais

A crescente integração dos Sistemas de Controle Industrial com as redes de IT (Tecnologia da Informação), impulsionada pela Indústria 4.0, escancarou as fragilidades de protocolos legados que não foram projetados para operar em ambientes hostis. O Modbus TCP, amplamente adotado no chão de fábrica, carece de mecanismos fundamentais como criptografia e autenticação. A substituição repentina de toda a base de controladores instalada é inviável, tornando indispensável o desenvolvimento de camadas de defesa adicionais.

Neste cenário, a premissa de que a segurança não pode comprometer a alta disponibilidade do processo industrial guiou a estruturação deste trabalho. O desenvolvimento e a implementação do Sistema de Detecção de Intrusão (IDS) validaram a tese de que é possível auditar ativamente um ambiente de Tecnologia de Operação (OT) sem introduzir latência na malha de controle. A utilização de ferramentas de código aberto, notadamente Python e Scapy, demonstrou ser uma alternativa técnica sólida e de baixo custo para extrair telemetria profunda da rede industrial e exportá-la de forma segura para sistemas de monitoramento como o Node-RED.

5.2 Cumprimento dos Objetivos

O desenvolvimento deste projeto culminou na entrega de um protótipo funcional[1], atestando o cumprimento do objetivo geral. Em relação aos objetivos específicos propostos, todos foram integralmente alcançados:

- Desenvolveu-se um software especializado de análise de pacotes (*sniffer*) em Python, que capturou e decodificou de forma assíncrona o tráfego da rede, extraindo os campos vitais do protocolo Modbus TCP.
- As regras do IDS se mostraram eficazes em três camadas operacionais distintas, detectando escritas não autorizadas, falsificações e inundações volumétricas sem disparar falsos positivos durante a operação da linha de base de 14 pkt/s.
- A regra de Inspeção de Carga Útil Modbus operou com êxito na Camada 7, bloqueando comandos de escrita ílegitimos.
- Foi validada em ambiente prático, a arquitetura proposta, composta pela ponte IIoT através do protocolo MQTT e visualização de dados via um *dashboard* em Node-RED que informa e alerta os eventos críticos para o usuário.
- A infraestrutura híbrida composta pelo CLP CompactLogix físico e o ambiente FlexFact atestou a aplicabilidade das soluções testadas em condições de automação industrial similares.

5.3 Limitações e Trabalhos Futuros

Apesar do êxito na detecção dos ataques propostos, o escopo adotado apresenta certas limitações inerentes à estratégia de *Whitelist Comportamental* empregada. O IDS atual baseia-se na restrição de IP e do Código de Função (FC) do Modbus. Existe uma vulnerabilidade lógica contundente conhecida como *False Data Injection* (Injeção de Dados Falsos). Caso um atacante extremamente sofisticado execute o *IP Spoofing* (assumindo o IP legítimo da rede) e injete o *mesmo código de função* permitido pelo CLP (FC 15), manipulando apenas o conteúdo da carga útil (*payload*) para alterar uma malha de controle (ex: alterando os *coils* para parar a máquina indevidamente), o ataque será capaz de passar pelo filtro do *sniffer*. Para neutralizar este cenário, o IDS precisaria conhecer o contexto físico e dinâmica do processo, correlacionando o dado de rede com a variável de processo real e limitando a lógica do atuador.

Ademais, destaca-se que os testes e a aferição de não-invasividade foram conduzidos em um ambiente laboratorial restrito a uma única célula de manufatura (o sistema FlexFact), cuja vazão nominal de tráfego foi baixa (na casa dos 14 pacotes por segundo). Faltou submeter a arquitetura assíncrona da biblioteca Scapy (que roda em *user-space*) a um teste de estresse realístico contendo o tráfego denso de centenas de dispositivos comunicando-se simultaneamente (em escala de *Gigabits*). Permanece a incerteza se, sob grande tráfego de rede, a captura passiva em Python causaria esgotamento de *buffers* ou perdas de pacote, o que inviabilizaria a métrica de cálculo da janela deslizante do IDS.

Tendo em vista as limitações citadas acima, sugerem-se como trabalhos futuros:

[1]Todo o código desenvolvido, incluindo o sniffer (`sniffer_ids.py`), script de teste dos ataques (`pentest.py`), fluxos do Node-RED, lógicas de CLP e planta do FlexFact estão disponíveis no repositório ModbusTCP-IDS no GitHub: `https://github.com/Therocyn/ModbusTCP-IDS`

- Testes em Ambiente Real: Empregar o protótipo de IDS em uma planta densa, sujeitando-o à um grande tráfego de rede visando validar se o sniffer funcionaria em um ambiente mais próximo ao real.
- Analise do Protocolo EtherNet/IP: Em ambientes industriais modernos, um dos protocolos dominantes é o protocolo CIP (Common Industrial Protocol) encapsulado em Ethernet (EtherNet/IP). Ajustar a decodificação do *sniffer* para suportar DPI de requisições CIP permitiria estender a proteção aos sistemas mais atualizados.
- Transição de IDS para IPS de Rede: Estudar a viabilidade de utilizar o motor de detecção para atuar ativamente na rede, permitindo que, além de detectar anomalias, o protótipo possa descartar pacotes maliciosos e interromper o ataque antes que este alcance o dispositivo alvo.

REFERÊNCIAS

BORDIGNON, G. D. *Desenvolvimento de um gateway de protocolo com suporte a Modbus e MQTT*. 2024. Diss. (Mestrado) – Universidade Federal de Santa Catarina, Florianópolis.

BOŠTJANČIČ RAKAS, S. V.; STOJANOVIĆ, M. D.; MARKOVIĆ-PETROVIĆ, J. D. A Review of Research Work on Network-Based SCADA Intrusion Detection Systems. *IEEE Access*, v. 8, p. 93099–93118, 2020. DOI: 10.1109/ACCESS.2020.2994961.

BRANCATI, F. *et al.* A cybersecurity risk assessment methodology for industrial automation control systems. *International Journal of Information Security*, v. 24, n. 1, p. 76, 2025. DOI: 10.1007/s10207-025-00990-9.

CINDRIĆ, I.; JURČEVIĆ, M.; HADJINA, T. Mapping of Industrial IoT to IEC 62443 Standards. *Sensors*, v. 25, n. 3, p. 728, 2025. DOI: 10.3390/s25030728.

D'AMBROSIO, N. *et al.* SCASS: Breaking into SCADA Systems Security. *Computers & Security*, v. 151, p. 104315, 2025. DOI: 10.1016/j.cose.2024.104315.

DA SILVA, M. *et al.* Safety-security convergence: Automation of IEC 62443-3-2. *Computers & Security*, v. 156, p. 104477, 2025. DOI: 10.1016/j.cose.2025.104477.

EKISA, C.; Ó BRIAIN, D.; KAVANAGH, Y. Modelling and Simulating Advanced Cyber-threats to Industrial Control Systems with an Emulated Testbed. *In:* 2024 35th Irish Signals and Systems Conference (ISSC). [*S. l.: s. n.*], 2024. DOI: 10.1109/ISSC61953.2024.10603140.

ERVURAL, B. C.; ERVURAL, B. Overview of Cyber Security in the Industry 4.0 Era. *In: Industry 4.0: Managing The Digital Transformation*. Edição: Alp Ustundag e Emre Cevikcan. Cham: Springer International Publishing, 2018. p. 267–284. ISBN 978-3-319-57870-5. DOI: 10.1007/978-3-319-57870-5_16.

EVANGELIOU I., E. *Vulnerabilities of the Modbus Protocol*. 2018. Diss. (Mestrado) – University of Piraeus, Athens, Greece.

FENG, W.; LAI, Y.; LIU, Z. Vulnerability mining for Modbus TCP based on exception field positioning. *Simulation Modelling Practice and Theory*, v. 102, p. 101989, 2020. DOI: 10.1016/j.simpat.2019.101989.

FG DES. *FG DES - Manufacturing System Simulator*. Friedrich-Alexander-Universität Erlangen-Nürnberg. Disponível em: https://www.fgdes.tf.fau.de/flexfact.html. Acesso em: 12 nov. 2025.

JAATUN, M. G. *et al.* Security Aspects of Zones and Conduits in IEC 62443. *Journal of Cybersecurity and Privacy*, v. 4, p. 367–382, 2024. DOI: 10.3390/jcp6020052.

KOBAYASHI, T. H. *et al.* Using a Packet Manipulation Tool for Security Analysis of Industrial Network Protocols. *In:* 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007). [*S. l.: s. n.*], 2007. p. 744–747. DOI: 10.1109/EFTA.2007.4416867.

LASI, H. *et al.* Industry 4.0. *Business & Information Systems Engineering*, Springer, v. 6, n. 4, p. 239–242, 2014. DOI: 10.1007/s12599-014-0334-4.

LEZZI, M.; LAZOI, M.; CORALLO, A. Cybersecurity for Industry 4.0 in the current literature: A reference framework. *Computers in Industry*, v. 103, p. 97–110, 2018. DOI: 10.1016/j.compind.2018.09.004.

LI, H.; QIN, S. Optimization and Implementation of Industrial Control System Network Intrusion Detection by Telemetry Analysis. *In:* 2017 3rd IEEE International Conference on Computer and Communications (ICCC). [*S. l.: s. n.*], 2017. p. 1265–1269. DOI: 10.1109/CompComm.2017.8322676.

MAHMOOD, J. R.; ALI, R. S. Personal computer/programmable logic controller based variable frequency drive training platform using WxPython and PyModbus. *International Journal of Electrical and Computer Engineering (IJECE)*, v. 12, n. 4, p. 3564–3571, 2022. DOI: 10.11591/ijece.v12i4.pp3564-3571.

AL-MAITAH, M. A.; AL-KHAYER, A.; DARAWSHEH, F. Vulnerability Assessment and Mitigation of Modbus TCP in SCADA Systems. *International Journal of Advanced Computer Science and Applications (IJACSA)*, v. 11, n. 1, 2020. DOI: 10.14569/IJACSA.2020.0110101.

MEKALA, S. H. *et al.* Cybersecurity for Industrial IoT (IIoT): Threats, countermeasures, challenges and future directions. *Computer Communications*, v. 208, p. 294–320, 2023. DOI: 10.1016/j.comcom.2023.06.020.

MILLER, T. *et al.* Looking back to look forward: Lessons learnt from cyber-attacks on Industrial Control Systems. *International Journal of Critical Infrastructure Protection*, v. 35, p. 100464, 2021. DOI: 10.1016/j.ijcip.2021.100464.

MODBUS ORGANIZATION, INC. *MODBUS Application Protocol Specification V1.1b3*. [*S. l.*], abr. 2012. Disponível em: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.

MODBUS ORGANIZATION, INC. *Modbus Messaging on TCP/IP Implementation Guide V1.0b*. [*S. l.*], out. 2006. Disponível em: http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.

NGUYEN-HOANG, P.; VO-TAN, P. Development An Open-Source Industrial IoT Gateway. *In:* 2019 19th International Symposium on Communications and Information Technologies (ISCIT). [*S. l.: s. n.*], 2019. p. 201–204. DOI: 10.1109/ISCIT.2019.8905208.

PARIAN, C.; GULDIMANN, T.; BHATIA, S. Fooling the Master: Exploiting Weaknesses in the Modbus Protocol. *Procedia Computer Science*, v. 171, p. 2453–2458, 2020. DOI: 10.1016/j.procs.2020.02.261.

PLAMOWSKI, S. *et al.* Infrastructure and Tools for Testing the Vulnerability of Control Systems to Cyberattacks: A Coal Mine Industrial Facility Case. *Applied Sciences*, v. 14, n. 23, p. 11325, 2024. DOI: 10.3390/app142311325.

PLIATSIOS, D. *et al.* A Survey on SCADA Systems: Secure Communication, Real-Time Operation and Feasibily. *Procedia Computer Science*, v. 168, p. 234–241, 2020. DOI: 10.1016/j.procs.2020.02.261.

RADOGLOU-GRAMMATIKIS, P. *et al.* Implementation and Detection of Modbus Cyberattacks. *In:* 2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST). [*S. l.: s. n.*], 2020. DOI: 10.1109/MOCAST49295.2020.9200250.

RAHMAN, A. *et al.* Launch of denial of service attacks on the modbus/TCP protocol and development of its protection mechanisms. *International Journal of Critical Infrastructure Protection*, v. 39, p. 100568, 2022. DOI: 10.1016/j.ijcip.2022.100568.

RAKAS, S. V. B.; STOJANOVIĆ, M. D.; MARKOVIĆ-PETROVIĆ, J. D. A Review of Research Work on Network-Based SCADA Intrusion Detection Systems. *IEEE Access*, v. 8, p. 93083–93108, 2020. DOI: 10.1109/ACCESS.2020.2994961.

RODRÍGUEZ, R. J. *et al.* MOSTO: A toolkit to facilitate security auditing of ICS devices using Modbus/TCP. *Computers & Security*, v. 132, p. 103373, 2023. DOI: 10.1016/j.cose.2023.103373.

SENGUPTA, J.; RUJ, S.; BIT, S. D. A Secure Fog-Based Architecture for Industrial Internet of Things and Industry 4.0. *IEEE Transactions on Industrial Informatics*, v. 17, n. 4, p. 2316–2324, abr. 2021. DOI: 10.1109/TII.2020.2998105.

SUCHORAB, J.; PLAMOWSKI, S.; ŁAWRYŃCZUK, M. Anomaly detection system for Modbus data based on an open source tool. *Computers & Security*, v. 157, p. 104572, 2025. DOI: 10.1016/j.cose.2025.104572.

TIETZ, F.; BRANDÃO, D.; ALVES, L. F. Development of an Internet of Things Gateway Applied to a Multitask Industrial Plant. *In:* 2018 13th IEEE International Conference on Industry Applications (INDUSCON). [*S. l.: s. n.*], 2018. p. 917–923. DOI: 10.1109/INDUSCON.2018.8627252.

TRUNGADI, F. *et al.* Securing Modbus in legacy industrial control systems: A decentralized approach using proxies, Post-Quantum Cryptography and Self-Sovereign Identity. *Journal of Information Security and Applications*, v. 94, p. 104199, 2025. ISSN 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2025.104199. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214212625002364.

USTUNDAG, A.; CEVIKCAN, E. (ed.). *Industry 4.0: Managing The Digital Transformation*. Cham: Springer International Publishing, 2018. (Springer Series in Advanced Manufacturing). ISBN 978-3-319-57870-5. DOI: 10.1007/978-3-319-57870-5.

Apêndices

Apêndice A - DOCUMENTAÇÃO DO REPOSITÓRIO DE CÓDIGO

Este apêndice apresenta a documentação e passos necessários para reprodução dos códigos disponibilizados no repositório do projeto (`https://github.com/Therocyn/ModbusTCP-IDS`). A finalidade desta documentação é orientar engenheiros e pesquisadores da área de automação no processo de montagem, configuração e reprodução dos ensaios de segurança cibernética validados ao longo deste trabalho.

Descrição Geral da Solução

O repositório armazena os algoritmos e arquivos de configuração desenvolvidos como parte integrante deste Trabalho de Conclusão de Curso em Engenharia de Controle e Automação pela Universidade Federal do Rio Grande do Sul (UFRGS).

A solução consiste em um protótipo de Sistema de Detecção de Intrusão (IDS) não-invasivo focado na proteção do protocolo Modbus TCP em redes de Tecnologia de Operação (OT). Operando por meio de análise passiva do tráfego através de escuta de interface (*sniffing*), o sistema atua de forma transparente na infraestrutura de rede, sem adicionar latência ou atrasos ao ciclo de controle nominal de 80 ms da planta, salvaguardando os requisitos de disponibilidade e segurança física (*Safety*) recomendados pelas diretrizes da norma ISA/IEC 62443.

Arquitetura e Estrutura de Arquivos

Os arquivos do repositório estão organizados para cobrir tanto o ambiente físico-simulado de controle industrial (*testbed* OT) quanto a camada de segurança e monitoramento (IT):

- **`sniffer_ids_v7.py`**: Núcleo algorítmico do IDS escrito em Python. Faz o uso da biblioteca Scapy para captura assíncrona de pacotes em nível de *kernel* e executa rotinas de Inspeção de Carga Útil Modbus na camada de aplicação.
- **`pentest_ics.py`**: Script de auditoria e testes ofensivos parametrizável. É capaz de simular três vetores distintos de ataque para homologar a eficácia das regras contidas no sniffer.
- **`flowsTCC_v4.json`**: Arquivo de fluxo estruturado para importação no Node-RED (Dashboard do Centro de Operações de Segurança - SOC). Consolida telas de alarmes históricos e medidores de vazão alimentados via protocolo MQTT.

- `code_tcc_rev3.ACD`: Arquivo de projeto compilado para o Rockwell Studio 5000. Contém a lógica em linguagem *Ladder* e a temporização do cliente Modbus TCP que roda nativamente no CLP CompactLogix.
- `flexfact_code_tcc_rev3.ffs`: Malha gráfica e layout de processos da planta industrial virtual configurada dentro do simulador FlexFact (atuando como servidor Modbus TCP).

Mecanismos e Regras de Detecção Implementadas

O motor de análise processa os pacotes capturados com base em três camadas complementares de defesa:

1. **Regra 1 - Anomalia Volumétrica (DoS):** Monitora a taxa de requisições por segundo por meio de um algoritmo de Janela Deslizante (*Sliding Window*) com tempo amostral de 1 segundo. Fluxos contínuos que atinjam ou ultrapassem o patamar de 40,0 pkt/s acionam alertas imediatos de alta prioridade para ataques de Negação de Serviço.
2. **Regra 2 - Controle de Acesso (IP Whitelist):** Implementa uma lista estrita de permissões contendo os endereços IP autorizados a emitir pacotes de controle para alteração de estados da planta.
3. **Regra 3 - Inspeção de Carga Útil Modbus:** Realiza a análise dos campos internos do cabeçalho Modbus TCP (Camada 7). Restringe o tráfego aos *Function Codes* (FC) intrínsecos à operação (como FC 15 para escritas de coils e FC 02 para leituras). Tentativas de injeção de comandos com códigos anômalos ou pacotes forjados com falsificação de identidade (*IP Spoofing*) são identificadas e reportadas imediatamente.

Instruções de Execução

1. Requisitos de Sistema

O ambiente operacional requer a instalação do Python na versão 3.8 ou superior. As dependências lógicas de bibliotecas podem ser instaladas diretamente por meio do terminal:

```
pip install scapy paho-mqtt pymodbus
```

- **Restrição Windows:** É mandatória a instalação prévia do driver de captura **Npcap** com a flag de compatibilidade com WinPcap habilitada. Caso contrário, o Scapy não terá permissões para funcionamento completo.
- **Camada de Telemetria:** É necessária uma instância do Node-RED rodando localmente ou em servidor e acesso à internet para conexão ao Broker MQTT público configurado (`broker.hivemq.com:1883`).

2. Inicialização do Testbed Industrial

Para colocar o laboratório de testes em funcionamento, siga a sequência operacional descrita abaixo:

1. Inicie o Node-RED, faça a importação do arquivo `flowsTCC_v4.json` e clique em *Deploy* para subir o painel na rota `/dashboard`.
2. Abra o software FlexFact, carregue o layout `flexfact_code_tcc_rev3.ffs` e coloque a simulação em modo ativo.
3. Conecte-se ao CLP via Studio 5000 e descarregue a lógica contida no arquivo `code_tcc_rev3.ACD`, iniciando o ciclo de controle nominal.
4. Abra um terminal de comandos do sistema com privilégios de **Administrador** e execute o script do sniffer:

```
python sniffer_ids_v7.py
```

3. Homologação das Regras (Auditoria Cibernética)

Para validar se o IDS capturará as intrusões com sucesso, utilize o script de testes ofensivos `pentest_ics.py`. Antes de rodar o programa, edite a variável global `TIPO_ATAQUE` no início do código-fonte para selecionar o cenário desejado:

- **Parâmetro igual a 1**: Dispara um ataque volumétrico (inundação de requisições) para testar o estouro do limiar de DoS.
- **Parâmetro igual a 2**: Efetua uma injeção direta de comandos de escrita a partir de uma estação de trabalho sem permissões.
- **Parâmetro igual a 3**: Executa a montagem artesanal de *frames* via Scapy aplicando *IP Spoofing* para testar a robustez do motor de DPI contra falsificação de identidade na Camada 3.

Com a variável definida, abra um segundo terminal de administrador e execute a ferramenta:

```
python pentest_ics.py
```