Static analysis for Ladder Logic Bombs (LLB) detection in Programmable Logic Controllers (PLC)

Diego Sousa de Miranda[1], **Victor Takashi Hayashi**[1]

[1]Laboratório de Arquitetura e Redes de Computadores – Escola Politécnica
Universidade de São Paulo (USP) – 05508-010 – São Paulo – SP – Brasil

{diego.miranda,victor.hayashi}@usp.br

Abstract. *PLCs are critical components because they translate control decisions into actions on sensors and actuators. In this context, LLB represent a relevant threat due to their ability to remain stealthy within the PLC control logic. This work evaluates the static detection of LLB based on attributes extracted from PLC programs, organized into structural and topological, logical, semantic, and sequential groups. Supervised and unsupervised techniques were applied to analyze the discriminative potential of these attributes. The models trained with logical and semantic attributes achieved the best F1-scores, both reaching 0.9692. The clustering of logical attributes also indicated separability, with a Silhouette score of 0.8529 and a Davies-Bouldin index of 0.2155.*

Resumo. *Os PLCs são elementos críticos por traduzirem decisões de controle em ações sobre sensores e atuadores. Nesse contexto, LLB representam uma ameaça relevante devido à sua capacidade furtiva na lógica de controle do PLC. Este trabalho avalia a detecção estática de LLB a partir de atributos extraídos de programas PLC, organizados em grupos estruturais e topológicos, lógicos, semânticos e sequenciais. Foram aplicadas técnicas supervisionadas e não supervisionadas para analisar o potencial discriminativo desses atributos. Os modelos treinados com atributos lógicos e semânticos alcançaram os melhores F1-scores, ambos iguais a 0,9692. A clusterização dos atributos lógicos indicou separabilidade, com Silhouette de 0,8529 e Davies-Bouldin de 0,2155.*

1. Introdução

As Infraestruturas Críticas (IC), cuja evolução tem sido impulsionada pela emergente Indústria 4.0, constituem a base operacional e estratégica das sociedades modernas. Esses ambientes tornaram-se mais integrados, automatizados e dependentes de sistemas computacionais. Os sistemas de controle industrial (*Industrial Control Systems* - ICS) representam o conjunto de tecnologias que integram sensores, atuadores e controladores com o objetivo de monitorar, automatizar e controlar processos industriais e físicos, viabilizando a operação contínua de setores como energia, transportes, abastecimento de água e saneamento e nuclear [Asghar et al. 2019].

Dentro das arquiteturas ICS, os controladores lógicos programáveis (*Programmable Logic Controllers* – PLC) exercem papel central, pois executam a lógica de controle e fazem a interface direta com sensores e atuadores de campo [Langmann and Stiller 2019]. Alertas recentes de segurança cibernética indicam que PLC tem sido alvo direto em ambientes de infraestrutura crítica. A *Cybersecurity and Infrastructure Security Agency*

(CISA) e agências parceiras relataram o comprometimento de PLC utilizados em múltiplos setores, incluindo sistemas de água e esgoto, energia, manufatura, alimentos e bebidas e saúde [Cybersecurity and Infrastructure Security Agency et al. 2023].

Como PLC podem ser fornecidos por diferentes fabricantes, a IEC 61131-3 padroniza as linguagens de programação empregadas nesses controladores, incluindo *Ladder Diagram* (LD), *Function Block Diagram* (FBD), *Sequential Function Chart* (SFC) ou *Structured Text* (ST), assim como os diferentes tipos de *Program Organization Units* (POU), definidos como *program*, *function block* ou *function* [International Electrotechnical Commission 2013].Complementarmente, o formato PLCopen XML, incorporado à IEC 61131-10, permite a representação estruturada de projetos PLC facilitando o intercâmbio de POU entre diferentes ambientes de desenvolvimento [International Electrotechnical Commission 2019].

O conceito de Ladder Logic Bombs (LLBs), inicialmente introduzido por [Govil et al. 2017], descreve esse tipo de ameaça como um código malicioso que pode ser inserido diretamente na linguagem IEC 61131-3, permanecer oculto entre blocos legítimos de códigos e ser ativado apenas sob condições específicas. Devido a complexidade da lógica de controle, múltiplas linguagens, diferentes blocos funcionais interconectados e a ausência de um mecanismo automatizado de inspeção, identificar LLB manualmente torna-se extremamente difícil. A modelagem do atacante aqui admitida, consiste de comprometimento da cadeia de suprimentos, na qual o atacante tenha acesso à lógica do controlador e conhecimento do processo industrial.

A detecção de ameaças em PLCs pode ser realizada por abordagens dinâmicas, que observam o comportamento do controlador durante a execução, como comandos enviados, escritas em memória e variáveis de processo ou emissões eletromagnéticas [Boateng and Bruce 2022, Vaidyan and Tyagi 2022, Serhane et al. 2019]. Embora relevantes, essas técnicas dependem da cobertura das condições de ativação da LLB. A análise estática proposta neste trabalho busca identificar a presença de LLB antes da ocorrência do gatilho, além de apoiar a priorização de trechos suspeitos e a definição de cenários de teste mais direcionados.

O restante deste artigo está organizado da seguinte forma. A Seção II discute os trabalhos relacionados, abordando pesquisas voltadas à detecção de LLB, análise de código PLC e identificação de anomalias em lógicas de controle. A Seção III descreve a metodologia empregada para análise dos programas PLC. Em seguida, a Seção IV apresenta os resultados obtidos, incluindo a avaliação comparativa dos modelos, a análise da importância de atributos e dos grupos de atributos. A Seção V discute as limitações do trabalho, oportunidades de melhorias e possíveis direções futuras. Por fim, a Seção VI apresenta as conclusões do trabalho e os aspectos relevantes que ensejam a continuidade da pesquisa.

2. Trabalhos relacionados

A literatura apresenta diferentes perspectivas para tratar esse problema, variando desde a modelagem conceitual da ameaça até abordagens baseadas em verificação formal, análise dinâmica, detecção de alterações, mineração de processos e aprendizado de máquina.

Trabalhos mais recentes passaram a explorar a detecção de anomalias considerando a estrutura da lógica de controle. Algumas abordagens utilizam representações

sequenciais da lógica do PLC, como lista de instruções (*Instruction List* – IL), separando elementos como *opcode* e *operand* para alimentar modelos de aprendizado profundo, incluindo LSTM, *Autoencoders* e *Transformers*. Esse tipo de representação permite capturar padrões sequenciais da lógica e explorar relações sintáticas entre instruções. Contudo, tais métodos dependem da conversão da lógica para IL e tendem a apresentar menor interpretabilidade direta. Além disso, quando instruções individuais são tratadas como instâncias de treinamento, é necessário cuidado metodológico para evitar vazamento de dados entre programas derivados da mesma amostra original [Lee et al. 2025].

Outra linha de pesquisa relevante utiliza verificação formal para analisar programas PLC. Trabalhos como os de [Iacobelli et al. 2024] e [Rinieri et al. 2024] propõem arquiteturas voltadas à detecção de LLB por meio de propriedades formais, grafos de fluxo de controle e *model checking*. Essas abordagens apresentam rigor matemático e são adequadas para verificar comportamentos específicos da lógica de controle. Entretanto, podem exigir especificação formal do comportamento esperado, tradução da lógica para modelos verificáveis ou hipóteses específicas sobre o processo analisado. Isso pode limitar sua aplicação em ambientes com POU complexos, múltiplas linguagens IEC 61131-3 ou ausência de documentação completa da lógica esperada [Iacobelli et al. 2024, Rinieri et al. 2024].

Também existem trabalhos voltados à análise dinâmica e ao monitoramento em tempo de execução. Nesse grupo, incluem-se abordagens que observam eventos do processo, estados internos, variáveis de entrada e saída, comandos enviados ao controlador, endereços de memória ou comportamento operacional do sistema. Essas técnicas são úteis para detectar efeitos de ataques quando eles se manifestam durante a execução, aproximando-se do comportamento real do processo sob controle. Entretanto, no caso de ataques do tipo LLB, a lógica maliciosa pode permanecer inativa até que uma condição de disparo seja satisfeita. Assim, métodos puramente dinâmicos dependem da cobertura dos cenários testados e da capacidade de exercitar os gatilhos maliciosos durante a simulação ou operação monitorada [Boateng and Bruce 2022, Vaidyan and Tyagi 2022].

Há ainda estudos voltados à análise de vulnerabilidades em códigos LD e à segurança de sistemas baseados em PLC. Esses trabalhos apontam que vulnerabilidades no nível da lógica de controle podem decorrer tanto de más práticas de programação quanto da inserção intencional de trechos maliciosos. Eles contribuem para evidenciar que a segurança de PLC não deve se limitar a redes, *firmware* ou *hardware*, mas também considerar o código de controle carregado no controlador [Serhane et al. 2018, Serhane et al. 2019].

Outra vertente relacionada é a detecção forense de alterações na lógica de controle. Técnicas desse tipo são úteis para auditoria e identificação de mudanças não autorizadas em lógicas de controle de PLC. Porém, a simples detecção de modificação não implica necessariamente que a alteração seja maliciosa. Em ambientes industriais, mudanças legítimas podem ocorrer durante manutenção, atualização ou adaptação do processo. Portanto, identificar que a lógica foi modificada é diferente de determinar se a alteração introduz comportamento compatível com uma LLB [Yau and Chow 2015].

Além disso, há trabalhos que investigam anomalias em sistemas baseados em PLC por meio de métodos não supervisionados, como Isolation Forest, One-Class SVM ou *ensembles*. Contudo, em geral, elas se concentram em dados operacionais do processo,

variáveis de memória, sinais ou comportamento em tempo de execução, e não diretamente na análise da lógica de PLC [Boateng and Bruce 2022]. De forma semelhante, trabalhos baseados em mineração de processos ou logs de entrada e saída podem inferir modelos comportamentais do sistema, mas dependem de dados de operação e não identificam diretamente a presença de LLB no código exportado [Theis et al. 2019].

Diante dessa análise, observa-se uma oportunidade de investigação ainda pouco explorada: embora existam muitos trabalhos sobre segurança em PLC, ainda são limitadas as técnicas especificamente direcionadas à detecção de LLB. Este trabalho tem como objetivo contribuir para a detecção de LLB a partir de abordagens baseada na extração e análise de atributos de programas PLC. A abordagem proposta busca oferecer uma etapa preliminar de triagem baseada em atributos extraídos automaticamente da lógica PLC. Assim, o método é uma maneira complementar nos processos de revisão de código ao indicar lógicas de controle com características que merecem análise mais detalhada.

3. Metodologia

A metodologia deste trabalho parte da análise de programas PLC pertencentes a um *dataset* público, detalhado na Seção 3.4. Tais programas são analisados estaticamente e convertidos em uma representação vetorial de atributos, abrangendo características estruturais e topológicas, lógicas, semânticas e sequenciais. Em seguida, os atributos extraídos são avaliados por algoritmos de aprendizado de máquina supervisionados e por uma análise não supervisionada, buscando verificar quais grupos de atributos apresentam maior capacidade discriminativa entre programas PLC normais e programas contendo LLB.

3.1. Modelagem do atacante

Neste trabalho, considera-se um modelo de atacante voltado ao comprometimento da lógica de controle do PLC ao longo do ciclo de vida do processo controlado. Para isso, assume-se a inserção furtiva de LLB com condição de gatilho, utilizando a cadeia de suprimento do ambiente ICS como vetor de ataque. O ataque pode ocorrer em fases como desenvolvimento, instalação, comissionamento, manutenção e atualização da lógica de controle. O atacante possui capacidade de atuar tanto de forma externa, remota, quanto interna ao ambiente da ICS. Os objetivos do atacante podem incluir espionagem, sabotagem, degradação operacional, indisponibilidade de processos críticos, mantendo discrição quanto aos mecanismos de detecção [Plachkinova and Vo 2022].

3.2. Caracterização do programa PLC

O objetivo desta etapa do estudo foi extrair dados do programa PLC, originalmente representados em arquivos no padrão OpenPLC XML, e transformá-los numa representação vetorial para que fosse possível construir um modelo de aprendizagem de máquina.

A extração foi realizada por meio de um *parser* automatizado detalhado na próxima subseção. Por se tratar de uma análise experimental, buscou-se tornar o processo de obtenção de dados o mais amplo possível. O parser extraiu inicialmente 96 campos brutos dos arquivos PLCopen XML. Esses campos incluem atributos numéricos e booleanos, além de metadados e campos textuais, como nomes e caminhos de arquivos, listas de variáveis, nomes de POUs e outros elementos auxiliares. Os campos correspondentes a metadados e informações textuais foram descartados na etapa de preparação dos

dados para evitar que o modelo aprendesse padrões específicos do dataset, associados à identificação dos arquivos ou dos elementos da lógica de controle.

3.3. Parser

O *parser* é um *script* desenvolvido em *Python* responsável por processar arquivos PLCOpen XML e extrair dados do programa PLC. O algoritmo 1 apresenta o fluxo de execução do parser. O conjunto D representa os programas PLC analisados, enquanto cada x_i corresponde a um arquivo XML individual. Para cada arquivo, é gerada uma árvore estruturada T_i, da qual são extraídas informações sobre POUs (P_i), variáveis (V_i), corpos de programa (B_i) e tarefas (R_i). Quando a lógica contém elementos em LD, o algoritmo identifica contatos, bobinas, blocos, variáveis e conexões, construindo um grafo lógico $G_i = (N_i, E_i)$ para representar as relações entre os componentes. Nos trechos em ST, são analisadas instruções textuais, como atribuições, operadores, condicionais e laços. Ao final, todas as características estruturais, lógicas e sequenciais são reunidas em um vetor v_i, que passa a compor a matriz $F \in \mathbb{R}^{n \times d}$, em que n representa o número de programas analisados e d representa os diferentes atributos extraídos.

Para a reprodutibilidade dos experimentos, o código-fonte desenvolvido neste trabalho está disponível publicamente de forma anônima em https://github.com/dsmirandax/Projeto-LLB-detection.

3.4. *Dataset* utilizado

Para construção dos modelos, foi empregado o *dataset* correspondente aos programas PLC no formato OpenXML, proposto em [Iacobelli et al. 2024]. Esse *dataset* é composto por 60 amostras de programas PLC, balanceado entre programas legítimos e programas que contêm LLB, os quais modelam a lógica de controle de um sistema físico. Trata-se de uma das primeiras iniciativas voltadas à construção de um *dataset* público específico para análise de lógica de controle PLC, razão pela qual foi adotado como base experimental neste trabalho.

A lógica legítima modela um sistema simplificado de enchimento de tanque, no qual sensores e atuadores são utilizados para manter o nível de água dentro de limites operacionais definidos. Já as versões maliciosas foram construídas por meio da inserção de padrões de ataque do tipo LLB como efeito de negação de serviço (*Denied-of-Service* - DoS) na lógica de controle original. Ao todo, foram produzidas 30 variações de amostras maliciosas, explorando diferentes formas de alteração do programa, como manipulação de operações de comparação, modificação da leitura do nível de água e substituição ou alteração de componentes específicos da lógica de controle.

3.5. Preparação dos dados

Nesta etapa foi realizada a adequação dos dados brutos aos algoritmos de aprendizado de máquina, onde passam a ser chamados de atributos. Os dados extraídos foram convertidos em atributos numéricos e booleanos. A tabela 1 apresenta os atributos efetivamente extraídos da lógica de controle que são utilizados para treinamentos dos classificadores supervisionados e não-supervisionados.

Algorithm 1 Parser para extração de atributos de programas PLC

Require: Conjunto de arquivos PLCopen XML $D = \{x_1, x_2, \ldots, x_n\}$
Ensure: Matriz de atributos $F \in \mathbb{R}^{n \times d}$

$F \leftarrow \emptyset$
for all $x_i \in D$ **do**
  $T_i \leftarrow \text{PARSEXML}(x_i)$
  Inicializar o vetor de atributos $v_i \leftarrow \emptyset$
  $P_i \leftarrow$ localizar elementos `pou` em T_i
  $V_i \leftarrow$ extrair variáveis declaradas em T_i
  $B_i \leftarrow$ localizar corpos de programa em T_i
  $R_i \leftarrow$ extrair tarefas, intervalos e prioridades em T_i
  Calcular atributos estruturais a partir de P_i, V_i, B_i e R_i
  if T_i contém elementos `LD` **then**
    Extrair contatos, bobinas, blocos e variáveis
    Extrair conexões e identificadores locais
    Construir a representação lógica em grafo $G_i = (N_i, E_i)$
    Calcular atributos topológicos de G_i
  end if
  if T_i contém elementos `ST` **then**
    Extrair o texto dos nós `ST` e `xhtml:p`
    Identificar atribuições, comparações e operadores
    Identificar condicionais, laços e literais booleanos
    Calcular atributos lógicos e semânticos
  end if
  Gerar a representação sequencial intermediária
  Calcular atributos sequenciais
  $v_i \leftarrow$ concatenar os atributos extraídos de x_i
  Adicionar v_i como uma nova linha de F
end for
return F

Tabela 1. Atributos extraídos dos programas PLCs.

Grupo	Atributos efetivos	Qtd.
Estruturais e topológicos	`avg_degree`, `n_isolated_nodes`, `n_pous`, `n_function_blocks`, `n_inputs`, `n_outputs`, `n_local_vars`, `n_local_ids`, `n_inVariables`, `n_outVariables`, `n_int_inputs`, `n_int_outputs`, `n_outputs_written`	13
Lógicos	`n_conditionals_total`, `n_comparisons`, `n_if`, `n_end_if`, `n_while`, `n_end_while`, `has_loop_construct`, `n_true_literals`, `n_false_literals`	9
Semânticos	`st_text_length`, `n_constants`, `n_assignments`	3
Sequenciais	`seq_n_lines`, `seq_n_unique_opcodes`, `seq_n_unique_operands`, `seq_has_ld`, `seq_has_out`	5

3.6. Construção dos modelos supervisionados

Nesta etapa, foram utilizadas as configurações padrão dos classificadores disponibilizadas pelo PyCaret, com busca adicional de hiperparâmetros. Essa decisão foi adotada em razão do tamanho reduzido do *dataset* e do objetivo exploratório da comparação entre

diferentes famílias de classificadores. Assim, os resultados devem ser interpretados como uma avaliação inicial do potencial discriminativo dos atributos extraídos, e não como o desempenho máximo alcançável por cada algoritmo após otimização específica.

Para a construção dos modelos supervisionados, utilizou-se um *pipeline* automatizado de pré-processamento e treinamento por meio da biblioteca PyCaret [Ali 2020] em suas configurações padrão. A avaliação foi conduzida por meio de *Stratified K-Fold Cross-Validation* com $k = 5$ *folds*. Foram utilizados os algoritmos *Logistic Regression* (LR), *Random Forest* (RF), *Extra Trees* (ET), *Multilayer Perceptron* (MLP), *Gradient Boosting* (GB), *LightGBM* e *Support Vector Machine* (SVM).

3.7. Construção dos modelos não supervisionados

De maneira complementar a avaliação por classificadores supervisionados, realizou-se uma análise exploratória para buscar responder a seguinte pergunta: "*As características extraídas dos programas PLC apresentam algum agrupamento natural no espaço de atributos, mesmo utilizando dados sem rótulos de classe?*"

Para responder a essa pergunta, inicialmente, foi aplicada a Análise de Componentes Principais (*Principal Component Analysis* - PCA), reduzindo a dimensionalidade dos dados para duas componentes. Essa projeção permitiu visualizar, de forma simplificada, a proximidade entre programas normais e maliciosos.

Em seguida, foi aplicado o algoritmo *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) sobre os dados projetados. O DBSCAN foi utilizado por sua capacidade de identificar *clusters* com base na densidade das amostras e marcar pontos isolados como possíveis anomalias.

Os rótulos reais dos dados não foram utilizados na formação dos *clusters*, e sim, posteriormente, para comparação visual e interpretação dos *clusters* obtidos.

4. Resultados obtidos

Os resultados obtidos a partir dos experimentos descritos na metodologia são apresentados em duas etapas. A primeira corresponde à análise supervisionada, composta pela comparação de desempenho entre classificadores e pela avaliação da importância de atributos. A segunda corresponde à análise não supervisionada, realizada de forma complementar por meio de PCA e DBSCAN, com o objetivo de observar a distribuição das amostras e os agrupamentos formados no espaço de características.

4.1. Análise supervisionada

A Tabela 2 apresenta o desempenho dos classificadores. Embora não seja o foco deste trabalho, o RF apresentou o melhor desempenho geral, com acurácia de 0,9214 e *F1-score* de 0,9333, seguido por RL e pelo ET. Esse resultado sugere que modelos baseados em árvores foram capazes de explorar de forma eficiente as características extraídas dos arquivos PLCopen XML. Nota-se também que o LightGBM apresentou desempenho destoante, situação imposta pelo reduzido tamanho do *dataset*.

Adicionalmente, é analisado o impacto dos grupos de atributos no desempenho do classificador, bem como a importância individual dos atributos para o modelo que obteve

Tabela 2. Comparação de desempenho dos classificadores avaliados.

Classificador	**Acurácia**	***Recall***	**Precisão**	***F1-score***
Random Forest	0,9214	1,0000	0,8800	0,9333
Regressão Logística	0,9179	0,9333	0,9200	0,9156
Extra Trees	0,8929	0,9333	0,8933	0,8978
SVM Linear	0,8929	0,9333	0,8800	0,8933
Gradient Boosting	0,8679	0,9333	0,8533	0,8756
MLP	0,8679	0,8833	0,8700	0,8656
LightGBM	0,4714	0,2000	0,0857	0,1200

o melhor desempenho segundo a métrica *F1-score*. Essa métrica foi adotada por representar o equilíbrio entre precisão e *recall*, aspecto relevante em um cenário de triagem de lógicas maliciosas. Os resultados apresentados na Tabela 3 indicam que os grupos de atributos lógicos e semânticos apresentaram a maior contribuição para a detecção de LLB no *dataset* utilizado.

Considerando a distribuição dos atributos apresentada na Tabela 1 e a análise de importância do classificador RF, observa-se que atributos pertencentes aos grupos lógico e semântico estiveram entre os mais relevantes para a classificação. O comprimento do texto em ST (*st_text_length*) aparece entre os principais fatores, possivelmente devido ao incremento textual decorrente da inserção da LLB. Além disso, atributos como o total de condicionais (*n_conditionals_total*) e o número de comparações (*n_comparisons*) também apresentaram elevada importância. Esse comportamento sugere que o modelo explora alterações na estrutura lógica e semântica dos programas analisados. Tais resultados também podem indicar que, em razão do tamanho reduzido do *dataset*, a possibilidade de que parte do desempenho esteja associada a padrões específicos de construção das LLBs.

Tabela 3. Desempenho médio dos classificadores por grupo de características.

Grupo	**Acurácia**	**Precisão**	***Recall***	***F1-score***
Estruturais	0,8000	0,7810	0,8333	0,7966
Lógicas	0,9667	0,9429	1,0000	0,9692
Semânticas	0,9667	0,9429	1,0000	0,9692
Sequenciais	0,2167	0,2221	0,2667	0,2408

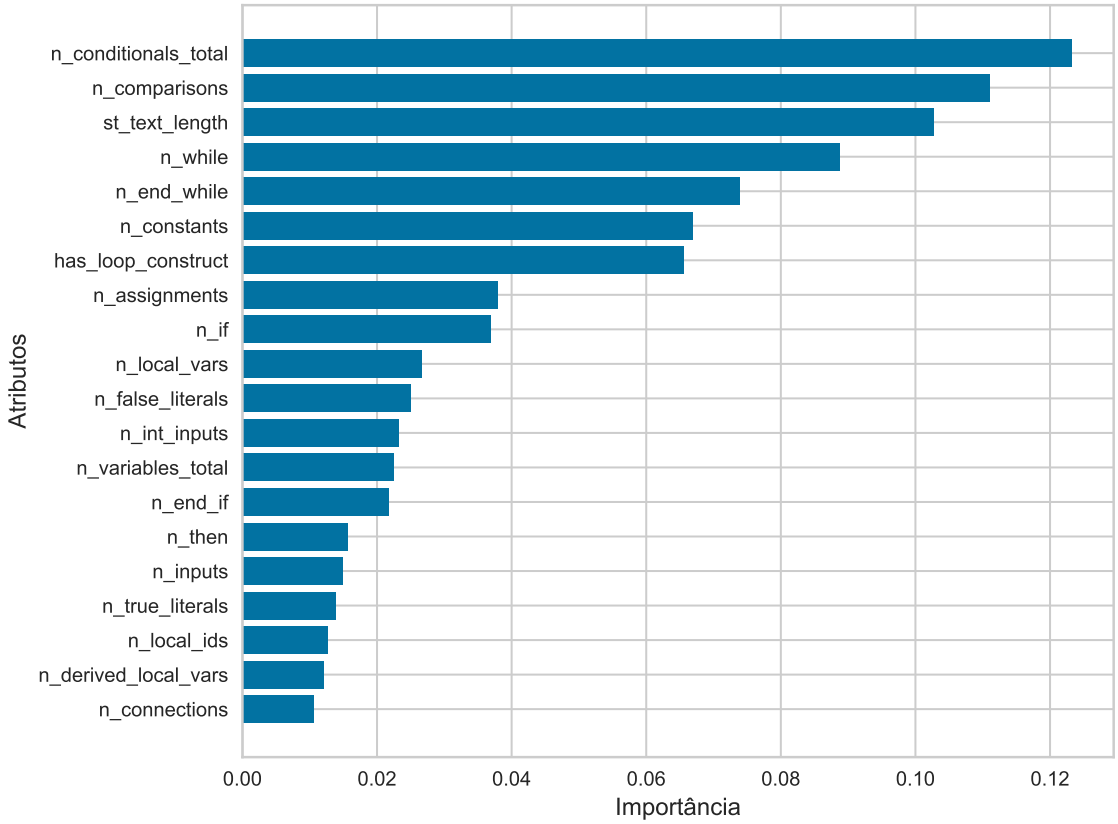


Figura 1. Importância dos 20 atributos mais relevantes do melhor modelo.

4.2. Análise não supervisionada

A Tabela 4 resume os resultados da análise exploratória com PCA e DBSCAN para os diferentes grupos de atributos. O DBSCAN foi aplicado sobre a projeção bidimensional obtida por PCA, utilizando os parâmetros `eps = 0.8` e `min_samples = 5`. Esses valores foram mantidos fixos para todos os grupos de atributos, a fim de permitir uma comparação uniforme entre as projeções avaliadas. De forma semelhante ao observado na análise supervisionada, o grupo de atributos lógicos apresentou resultado de destaque, com o maior valor de *Silhouette* entre os grupos avaliados, igual a 0,8529, e o menor índice *Davies-Bouldin*, igual a 0,2155. Além disso, a projeção por PCA preservou 88,91% da variância dos atributos lógicos, e o DBSCAN identificou cinco *clusters*, com 16,67% das amostras classificadas como ruído.

A Figura 2(a) apresenta a projeção bidimensional dos atributos lógicos por PCA, com as amostras de dados de acordo com seus rótulos reais. Observa-se que parte das amostras normais e maliciosas ocupa regiões distintas no espaço projetado, indicando que esses atributos mantêm padrões de separação em função da classe do programa.

A Figura 2(b) apresenta os agrupamentos identificados pelo DBSCAN sobre a mesma projeção. A composição dos agrupamentos foi comparada com as classes reais das amostras, conforme resultado apresentado na Tabela 5. Observa-se que os *clusters* 0 e 2 foram compostos exclusivamente por amostras normais, enquanto os *clusters* 3 e 4 foram compostos exclusivamente por amostras maliciosas. O *cluster* 1 apresentou predominância de amostras maliciosas, com 4 das 5 amostras, e apenas o grupo classificado

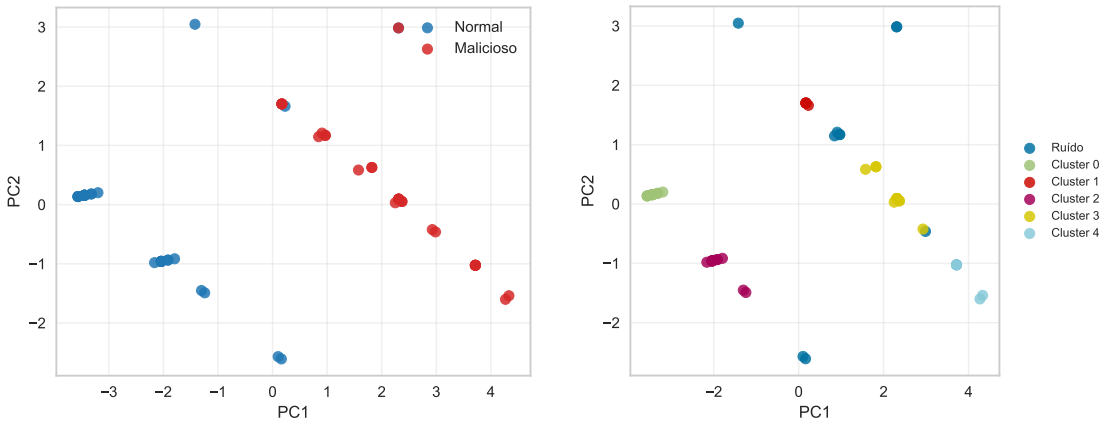
como ruído apresentou mistura entre as classes. Esse resultado também indica que, no conjunto de dados analisado, os atributos lógicos preservam uma estrutura de agrupamento compatível com a separação entre lógicas de controle normais e contendo LLB.

Tabela 4. Resumo da análise exploratória com PCA e DBSCAN.

Grupo	Nº atributos	PCA	Clust.	Ruído	Silh.	DBI
Estruturais	13	0,6249	7	21,67%	0,8055	0,3230
Lógicas	9	0,8891	5	16,67%	0,8529	0,2155
Semânticas	3	0,9919	2	1,67%	0,6963	0,4047
Sequenciais	5	0,7661	2	20,00%	0,6489	0,3853

Tabela 5. Composição dos clusters DBSCAN para atributos lógicos.

Cluster	Normal	Malicioso
-1	4	6
0	13	0
1	1	4
2	12	0
3	0	13
4	0	7



(a) Projeção PCA por rótulos reais. (b) Agrupamentos identificados pelo DBSCAN.

Figura 2. Análise exploratória dos atributos lógicos por PCA e DBSCAN.

5. Discussões e Trabalhos Futuros

Apesar dos resultados destacados na seção anterior, ao longo do desenvolvimento deste trabalho foi observada uma limitação importante relacionada aos *datasets* disponíveis publicamente, o que resulta em superajuste dos classificadores aos dados. A literatura referente análise de comportamento e detecção de LLB ainda carece de *datasets* mais amplos e padronizados, especialmente, contendo diferentes tipos de ataques e variações de processos industriais. Essa limitação, dificulta a avaliação da capacidade de modelos

de aprendizado de máquina e inviabiliza a generalização de conclusões. Assim, os resultados obtidos devem ser interpretados como uma análise experimental e exploratória sobre características estáticas de lógica de controle de PLC que são potencialmente úteis para detecção de LLB, e não uma solução geral para diferentes ambientes de automação industrial.

Como continuidade do trabalho, pretende-se investigar estratégias para ampliação dos *datasets* existentes, assegurando a funcionalidade do programa conforme o processo sob controle. Essa expansão pode envolver a geração de novas lógicas de controle normais e maliciosas, a geração de variações dos programas existentes e a aplicação de técnicas de *data augmentation*. Além da análise estática, pretende-se avançar para uma análise dinâmica do comportamento de LLBs e os efeitos no funcionamento do controlador. Nesse contexto, poderão ser obtidas métricas como duração e *jitter* de *scan cycle*, latência de resposta e consumo de memória. Adicionalmente, a análise dinâmica pode ser combinada com a análise estática, permitindo identificar a presença de código malicioso, bem como os efeitos observáveis durante a execução.

Por fim, estudos posteriores podem explorar representações alternativas da lógica PLC, como grafos e modelos baseados em aprendizado de máquina explicável. Essas abordagens podem contribuir para melhorar a interpretabilidade dos resultados, permitindo não apenas classificar um programa como normal ou malicioso, mas também indicar trechos da lógica de controle com a presença de código malicioso.

6. Conclusões

Este trabalho investigou a detecção estática de LLBs em lógicas de controle de PLCs, por meio da extração e análise de atributos estruturais e topológicos, lógicos, semânticos e sequenciais. Nos experimentos supervisionados, o classificador RF apresentou o melhor desempenho geral, com acurácia de 0,9214, *recall* de 1,0000, precisão de 0,8800 e *F1-score* de 0,9333. Na avaliação por grupos de atributos, os grupos lógico e semântico obtiveram os melhores resultados, ambos com acurácia de 0,9667 e *F1-score* de 0,9692, indicando maior poder discriminativo para características associadas a comparações, condicionais, atribuições e trechos em ST. Na análise não supervisionada, o grupo lógico também apresentou indícios de separabilidade, com *Silhouette* de 0,8529 e *Davies-Bouldin* de 0,2155 na clusterização com DBSCAN.

Ainda que os classificadores tenham apresentado desempenho satisfatório em termos relativos, os resultados não permitem conclusões definitivas quanto à generalização dos classificadores. Essa limitação decorre, principalmente, das características do *dataset* utilizado, que possui tamanho reduzido e diversidade limitada de amostras.

7. Agradecimento

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Ali, M. (2020). *PyCaret: An open-source, low-code machine learning library in Python*. PyCaret version 1.0.0.

Asghar, M. R., Hu, Q., and Zeadally, S. (2019). Cybersecurity in industrial control systems: Issues, technologies, and challenges. *Computer Networks*, 165:106946.

Boateng, E. A. and Bruce, J. W. (2022). Unsupervised machine learning techniques for detecting plc process control anomalies. *Journal of Cybersecurity and Privacy 2022, Vol. 2, Pages 220-244*, 2:220–244.

Cybersecurity and Infrastructure Security Agency, Federal Bureau of Investigation, National Security Agency, Environmental Protection Agency, and Israel National Cyber Directorate (2023). IRGC-Affiliated Cyber Actors Exploit PLCs in Multiple Sectors, Including U.S. Water and Wastewater Systems Facilities. Technical Report AA23-335A, Cybersecurity and Infrastructure Security Agency. Joint Cybersecurity Advisory.

Govil, N., Agrawal, A., and Tippenhauer, N. O. (2017). On ladder logic bombs in industrial control systems. In *Proceedings of the Workshop on the Security of Industrial Control Systems and of Cyber-Physical Systems (CyberICPS), co-located with ESORICS*.

Iacobelli, A., Rinieri, L., Melis, A., Sadi, A. A., Prandini, M., and Callegati, F. (2024). Detection of ladder logic bombs in plc control programs: an architecture based on formal verification. *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems, ICPS 2024*.

International Electrotechnical Commission (2013). IEC 61131-3: Programmable controllers – Part 3: Programming languages.

International Electrotechnical Commission (2019). IEC 61131-10: Programmable controllers – Part 10: PLCopen XML exchange format.

Langmann, R. and Stiller, M. (2019). The plc as a smart service in industry 4.0 production systems. *Applied Sciences*, 9(18):3815.

Lee, J. H. et al. (2025). Anomaly detection method considering plc control logic structure for ics cyber threat detection. *Applied Sciences*, 15(7):3507.

Plachkinova, M. and Vo, A. (2022). A taxonomy of cyberattacks against critical infrastructure. *Journal of Cybersecurity Education, Research and Practice*, 2021.

Rinieri, L., Iacobelli, A., Melis, A., Prandini, M., and Callegati, F. (2024). Plc-defuser: Detecting hidden ladder logic bombs in plcs via control flow graph and model checking.

Serhane, A., Raad, M., Raad, R., and Susilo, W. (2018). Plc code-level vulnerabilities. *2018 International Conference on Computer and Applications, ICCA 2018*, pages 348–352.

Serhane, A., Raad, M., Raad, R., and Susilo, W. (2019). Programmable logic controllers based systems (plc-bs): vulnerabilities and threats. *SN Applied Sciences*, 1:924–.

Theis, J., Mokhtarian, I., and Darabi, H. (2019). Process mining of programmable logic controllers: Input/output event logs. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 216–221.

Vaidyan, V. M. and Tyagi, A. (2022). Towards quantum artificial intelligence electromagnetic prediction models for ladder logic bombs and faults in programmable logic

controllers. *Proceedings of the 2022 International Conference on Electronic Systems and Intelligent Computing, ICESIC 2022*, pages 1–6.

Yau, K. and Chow, K.-P. (2015). Plc forensics based on control program logic change detection. *Journal of Digital Forensics, Security and Law*, 10(4).