# Relatório Técnico Avançado: Desenvolvimento de Arquitetura de Inteligência Artificial para Modelagem de Ameaças Automatizada via Análise de Diagramas (STRIDE)

## 1. Introdução e Contextualização Estratégica

A segurança de software, historicamente tratada como uma etapa final e obstrutiva no ciclo de desenvolvimento (o "gatekeeper"), enfrenta hoje um desafio existencial imposto pela velocidade das metodologias ágeis e DevOps. A prática de *Threat Modeling* (Modelagem de Ameaças), essencial para identificar vulnerabilidades na fase de design — onde a correção é ordens de magnitude mais barata do que na produção —, permanece um processo artesanal, dependente da escassa expertise de arquitetos de segurança sêniores. O desafio proposto pela FIAP Software Security 1 — o desenvolvimento de um sistema de Inteligência Artificial capaz de ingerir diagramas de arquitetura de sistemas e aplicar automaticamente a metodologia STRIDE — situa-se na vanguarda da engenharia de segurança moderna, conhecida como *Security by Design* automatizado.

Este relatório apresenta uma análise exaustiva, teórica e prática para a construção desta solução. O problema não é trivial: diagramas de arquitetura são representações semi-estruturadas de alta densidade semântica. Diferente do código-fonte, que segue gramáticas estritas passíveis de análise estática (SAST), um diagrama é uma imagem que carrega intenção, fluxo e fronteiras de confiança implícitas. A interpretação correta exige a convergência de múltiplas disciplinas da Ciência da Computação: Visão Computacional para reconhecimento de padrões visuais, Teoria dos Grafos para modelagem de relações e Engenharia de Segurança para a aplicação de heurísticas de risco.2

A relevância deste projeto transcende a automação; trata-se de democratizar a segurança. Ao fornecer feedback instantâneo sobre violações do STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) diretamente a partir de um esboço arquitetural, o sistema proposto capacita desenvolvedores a corrigir falhas estruturais antes que uma única linha de código seja escrita, alinhando-se aos princípios de *Shift-Left Security*.4 A solução aqui delineada propõe uma abordagem híbrida robusta, utilizando Redes Neurais Convolucionais (CNNs) de estado da arte, especificamente a arquitetura YOLO (You Only Look Once), para detecção de componentes, acoplada a algoritmos determinísticos de reconstrução de grafos e inferência lógica baseada em regras de segurança codificadas.

## 2. Fundamentação Teórica: Metodologia STRIDE e Arquitetura de Nuvem

Para desenvolver uma IA capaz de "pensar" como um especialista em segurança, é imperativo formalizar o conhecimento que este especialista utiliza. A metodologia STRIDE, desenvolvida pela Microsoft, fornece a taxonomia necessária para esta formalização.6 No entanto, a aplicação do STRIDE em arquiteturas de nuvem modernas (AWS, Azure) exige uma adaptação contextual que a IA deve aprender.

### 2.1 Decomposição do STRIDE para Análise Automatizada

A automação do STRIDE requer que cada categoria de ameaça seja traduzida em padrões visuais e topológicos detectáveis em um diagrama. A tabela a seguir detalha esta tradução, servindo como base para a engenharia de *features* do modelo de IA e para as regras do motor de inferência.7

| **Ameaça (STRIDE)** | **Propriedade de Segurança Violada** | **Definição Clássica** | **Padrão Arquitetural de Risco (Detectável Visualmente)** |
| --- | --- | --- | --- |
| **S**poofing | Autenticidade | Impersonificação de uma entidade. | Um fluxo de dados originado de uma zona não confiável (ex: Nuvem/Internet) conectando-se diretamente a um recurso interno (ex: Banco de Dados) sem passar por um componente de autenticação (ex: API Gateway, Cognito, Load Balancer). |
| **T**ampering | Integridade | Modificação não autorizada de dados. | Fluxos de dados atravessando redes públicas sem indicação de túneis seguros (VPN, TLS) ou componentes de escrita direta em armazenamento sem validação intermediária (ex: Lambda function). |
| **R**epudiation | Não-repúdio | Negação de autoria de uma ação. | Arquiteturas que carecem de componentes de log e auditoria (ex: CloudTrail, CloudWatch Logs) conectados aos fluxos de transação crítica ou bancos de dados. |
| **I**nformation Disclosure | Confidencialidade | Exposição de dados a não autorizados. | Buckets de armazenamento (S3, Blob Storage) ou Bancos de Dados situados fora de zonas privadas (VPCs), ou conexões de saída direta para a internet sem gateways NAT/Proxy. |
| **D**enial of Service | Disponibilidade | Degradação ou interrupção de serviço. | Servidores de aplicação expostos diretamente à internet sem camadas de proteção de borda (ex: WAF, Shield, CDNs) ou mecanismos de escalabilidade (Auto Scaling Groups). |
| **E**levation of Privilege | Autorização | Ganho de acesso não autorizado. | Componentes com permissões excessivas (inferidas) ou fluxos que saltam camadas de segurança, permitindo que um usuário anônimo acesse funções administrativas. |

A análise automatizada deve operar em duas dimensões principais:

1. **Análise por Elemento:** Avalia a configuração intrínseca do componente (ex: "Este ícone representa um banco de dados legado sem suporte a criptografia?").
2. **Análise por Interação:** Avalia o fluxo de dados entre componentes (ex: "O tráfego entre o Web Server e o DB está cruzando uma fronteira de confiança?").9

### 2.2 O Conceito de Fronteiras de Confiança (Trust Boundaries)

Um conceito crítico que a IA deve inferir é a **Fronteira de Confiança**. Em diagramas, estas são frequentemente representadas por caixas pontilhadas agrupando componentes (ex: uma VPC, uma Subnet, ou o perímetro corporativo). A violação de segurança ocorre frequentemente quando um fluxo de dados cruza uma fronteira de confiança sem a devida mediação.7

Para o MVP, a IA deve ser treinada não apenas para detectar ícones de servidores, mas também para detectar contêineres visuais (retângulos agrupadores) que denotam essas fronteiras. A lógica de detecção de ameaças deve então calcular a interseção entre as linhas de fluxo de dados (arestas do grafo) e esses limites de confiança. Se um fluxo Internet -> Database cruza a fronteira VPC sem passar por um nó Firewall ou WAF, uma ameaça de *Intrusão* e *Information Disclosure* é gerada automaticamente.

## 3. Engenharia de Dados: Construção de Bases de Treinamento Robustas

A eficácia de qualquer modelo de aprendizado supervisionado é diretamente proporcional à qualidade e quantidade dos dados de treinamento. No contexto de diagramas de arquitetura, enfrentamos um problema de escassez de dados: embora existam milhões de diagramas na internet, poucos estão anotados com metadados semânticos (bounding boxes e classes) necessários para treinar detectores de objetos como o YOLO.2

A estratégia recomendada para este projeto é a **Geração de Dados Sintéticos (Synthetic Data Generation)**, complementada por *Fine-Tuning* em datasets manuais. Esta abordagem resolve o problema do "Cold Start" e permite a criação de um dataset massivo, balanceado e perfeitamente rotulado.10

### 3.1 Pipeline de Geração de Dados Sintéticos com Python

Utilizando a biblioteca Python diagrams (que atua como uma interface de código para o Graphviz), podemos gerar diagramas de arquitetura aleatórios e, simultaneamente, extrair as coordenadas exatas de cada componente renderizado. Isso elimina a necessidade de anotação manual laboriosa.11

#### 3.1.1 Lógica de Geração Programática

O processo de geração sintética segue os seguintes passos:

1. **Definição do Espaço de Componentes:** Criar um dicionário de classes de interesse baseadas nos requisitos (Usuários, Servidores, Bancos de Dados, APIs, Firewalls) mapeados para os ícones das bibliotecas diagrams.aws, diagrams.azure, etc.
2. **Geração de Grafos Aleatórios:** Escrever um script que instancia um número aleatório de nós e cria conexões lógicas entre eles (ex: garantir que um DB esteja sempre conectado a um Compute, nunca diretamente à Internet, para criar exemplos positivos e negativos).
3. **Renderização e Extração de Metadados:** O Graphviz, motor por trás da biblioteca diagrams, pode exportar o layout no formato JSON ou DOT, que contém as coordenadas (pos), largura e altura de cada nó e aresta na imagem final.14
4. **Conversão para Formato YOLO:** As coordenadas do Graphviz (origem no canto inferior esquerdo, em *points*) devem ser convertidas para o formato YOLO (origem no canto superior esquerdo, normalizadas entre 0 e 1).

Abaixo, apresenta-se um exemplo estruturado de código para este pipeline, fundamental para o sucesso do MVP:

Python

# Exemplo Didático: Gerador de Dataset Sintético para Treinamento YOLO  
# Autor: Especialista em IA  
# Descrição: Gera diagramas aleatórios e arquivos de anotação correspondentes.  
  
import os  
import random  
from diagrams import Diagram, Cluster  
from diagrams.aws.compute import EC2, Lambda  
from diagrams.aws.database import RDS, DynamoDB  
from diagrams.aws.network import ELB, Route53  
from diagrams.onprem.client import User, Users  
import graphviz  
import shutil  
  
# Configurações do Dataset  
OUTPUT\_DIR = "./synthetic\_dataset"  
NUM\_IMAGES = 1000 # Quantidade de imagens a gerar  
CLASSES = # Classes mapeadas  
  
def gerar\_diagrama\_sintetico(id\_imagem):  
 filename = f"diag\_{id\_imagem}"  
   
 # Inicia o contexto do Diagrama (sem mostrar visualização imediata)  
 with Diagram("Architecture", show=False, filename=os.path.join(OUTPUT\_DIR, "images", filename), outformat="png"):  
 # Lógica de geração aleatória  
 num\_servers = random.randint(1, 3)  
 has\_db = random.choice()  
   
 # Componentes  
 lb = ELB(f"lb\_{id\_imagem}")  
 servers = [EC2(f"web\_{i}") for i in range(num\_servers)]  
   
 # Conexões (Definem a topologia)  
 lb >> servers  
   
 if has\_db:  
 db = RDS(f"db\_{id\_imagem}")  
 servers >> db  
  
 # O truque: A biblioteca 'diagrams' gera um arquivo.dot temporário.  
 # Precisamos interceptar esse.dot para extrair as coordenadas.  
 dot\_file = os.path.join(OUTPUT\_DIR, "images", filename) # A lib diagrams pode adicionar extensão  
   
 # Processar o arquivo DOT para extrair bounding boxes (Lógica Complexa omitida para brevidade)  
 # A função extrairia 'pos', 'width', 'height' do arquivo DOT gerado pelo Graphviz  
 # e salvaria um arquivo.txt correspondente na pasta labels/  
   
 # Exemplo de formato de saída esperado no.txt (YOLO format):  
 # <class\_id> <x\_center> <y\_center> <width> <height>  
 # 0 0.5 0.5 0.1 0.1

*Insight Técnico:* A geração sintética permite criar "casos de borda" difíceis de encontrar em datasets reais, como diagramas extremamente densos ou com sobreposição de setas, forçando o modelo a aprender representações mais robustas. Além disso, permite o *Domain Randomization*, variando cores de fundo e estilos de linha para evitar overfitting ao estilo padrão do Graphviz.15

### 3.2 Fontes de Dados Reais e Fine-Tuning

Para garantir que o modelo generalize para diagramas desenhados por humanos ou outras ferramentas (Draw.io, Lucidchart), o dataset sintético deve ser misturado com dados reais.

* **Roboflow Universe:** Existem datasets públicos de alta qualidade prontos para uso.
  + *AWS Icon Detector:* Contém milhares de ícones AWS anotados. Essencial para ensinar a IA a distinguir entre um S3 Bucket e um RDS Instance.16
  + *Azure Icons:* Similarmente, para a nuvem Microsoft.18
  + *Diagrams & Flowcharts:* Datasets de diagramas de fluxo genéricos ajudam no reconhecimento de setas e formas geométricas básicas.20

O processo de **Fine-Tuning** consiste em pegar um modelo pré-treinado em um dataset vasto (como o COCO, que conhece "pessoas" e "carros") e re-treinar suas camadas finais com nosso dataset híbrido (Sintético + Real) para que ele aprenda a detectar "Load Balancers" e "Firewalls". Isso economiza tempo computacional e melhora drasticamente a precisão com menos dados.21

## 4. Arquitetura da Solução MVP: Visão Computacional e Lógica

A solução proposta é um pipeline modular composto por três estágios principais: **Detecção Visual**, **Reconstrução de Grafo** e **Análise de Segurança**. Esta separação de responsabilidades segue os princípios de engenharia de software robusta, permitindo a evolução independente de cada módulo.

### 4.1 Estágio 1: Detecção de Objetos com YOLOv8/v11

Para a tarefa de identificar os componentes no diagrama, a arquitetura **YOLO (You Only Look Once)** é a escolha ideal devido ao seu equilíbrio entre velocidade e precisão (mAP - mean Average Precision). As versões mais recentes, como YOLOv8 ou v11, oferecem APIs Python extremamente amigáveis e suporte nativo a treinamento customizado.23

Implementação Técnica:

O modelo deve ser configurado para detectar classes que mapeiam diretamente para os elementos do STRIDE. Em vez de detectar 500 tipos de serviços AWS específicos, recomenda-se agrupar em super-classes funcionais para o MVP, reduzindo a confusão do modelo:

* Compute (EC2, Lambda, VM)
* Database (RDS, DynamoDB, SQL)
* Storage (S3, Blob)
* Gateway (API Gateway, Load Balancer)
* User (Mobile, Laptop, Person)
* External (Internet, 3rd Party API)

O output deste estágio é uma lista de objetos detectados, cada um com sua classe, confiança e caixa delimitadora (Bounding Box - [x1, y1, x2, y2]).

### 4.2 Estágio 2: Detecção de Conexões e Reconstrução de Grafo

Este é o componente mais complexo e onde a maioria das soluções falha. Saber que existem um "Usuário" e um "Banco de Dados" na imagem é inútil se não soubermos se eles estão conectados. A detecção de setas e linhas em diagramas é desafiadora devido à sua magreza e variabilidade (linhas curvas, tracejadas, cruzamentos).2

#### 4.2.1 Abordagem Heurística com Processamento de Imagem

Para um MVP robusto sem a necessidade de treinar um segundo modelo complexo apenas para setas, recomenda-se uma abordagem baseada em **Processamento Digital de Imagens (PDI)** clássico utilizando a biblioteca **OpenCV**.27

**Algoritmo de Reconstrução de Arestas:**

1. **Subtração de Nós:** Utilize as *bounding boxes* detectadas pelo YOLO para "apagar" os ícones da imagem (pintando-os de branco/fundo). Isso deixa apenas as linhas de conexão e textos soltos na imagem.
2. **Binarização e Esqueletização:** Converta a imagem residual para preto e branco e aplique operações morfológicas (esqueletização) para reduzir as linhas a 1 pixel de largura.
3. **Detecção de Contornos/Linhas:** Utilize cv2.findContours ou a Transformada de Hough Probabilística (cv2.HoughLinesP) para identificar os segmentos de linha.
4. **Associação de Proximidade:** Para cada linha detectada, verifique os pontos finais (endpoints). Se um ponto final estiver dentro ou muito próximo da *bounding box* expandida de um Nó A e o outro ponto próximo de um Nó B, infere-se uma conexão A <-> B.
5. **Inferência de Direção:** A detecção da "ponta da seta" pode ser feita analisando a densidade de pixels nas extremidades da linha ou treinando uma classe YOLO específica para "ArrowHead".29

Estrutura de Dados (NetworkX):

O resultado deste estágio é a construção de um grafo direcionado (networkx.DiGraph), onde:

* **Nós:** Representam os componentes (com atributos de metadados).
* **Arestas:** Representam os fluxos de dados (Data Flows).

### 4.3 Estágio 3: Motor de Análise STRIDE

Com o grafo da arquitetura em memória, a análise de segurança torna-se um problema de travessia de grafo e verificação de padrões. Não é necessário IA para esta etapa; regras lógicas determinísticas são mais precisas e explicáveis.31

A lógica pode ser implementada inspirando-se no framework **OWASP pytm**.31 O sistema itera sobre cada nó e aresta do grafo e verifica condições de risco.

**Exemplos de Regras Lógicas:**

* IF (Node.type == 'Database') AND (Edge.source.type == 'User') AND (Edge.protocol!= 'JDBC/ODBC via VPN') -> RAISE Spoofing/Elevation of Privilege (Acesso direto do usuário ao DB).
* IF (Node.type == 'Storage') AND (Node.attributes.encryption == False) -> RAISE Information Disclosure.
* IF (Edge.crosses\_trust\_boundary == True) AND (Edge.is\_encrypted == False) -> RAISE Tampering.

## 5. Guia Prático de Implementação (Passo a Passo Didático)

Abaixo, apresenta-se um roteiro técnico para a implementação do MVP, incluindo snippets de código essenciais.

### Passo 1: Preparação do Ambiente

As bibliotecas selecionadas compõem a stack padrão de Data Science e Visão Computacional em Python.

Bash

# Instalação das dependências principais  
pip install ultralytics # Framework YOLO (State-of-the-art detection)  
pip install opencv-python-headless # Processamento de imagem (Core CV)  
pip install networkx # Modelagem de Grafos e Algoritmos  
pip install matplotlib # Visualização  
pip install diagrams # Geração de dados sintéticos (Opcional mas recomendado)

### Passo 2: O Cérebro da Detecção (Inferência YOLO)

Este script assume que você já treinou o modelo (conforme descrito na seção 3) ou está usando um modelo genérico para testes iniciais.

Python

import cv2  
from ultralytics import YOLO  
  
def detectar\_componentes(caminho\_imagem):  
 # Carrega o modelo treinado (best.pt é o resultado do seu fine-tuning)  
 model = YOLO('models/best.pt')   
   
 # Realiza a inferência com um limiar de confiança ajustável  
 results = model.predict(caminho\_imagem, conf=0.5)  
   
 componentes =  
 for result in results:  
 for box in result.boxes:  
 # Extrai coordenadas normalizadas e classe  
 x1, y1, x2, y2 = box.xyxy.tolist()  
 cls\_id = int(box.cls)  
 label = model.names[cls\_id]  
   
 componentes.append({  
 "id": f"{label}\_{len(componentes)}", # ID único temporário  
 "tipo": label,  
 "bbox": (int(x1), int(y1), int(x2), int(y2)),  
 "centro": (int((x1+x2)/2), int((y1+y2)/2))  
 })  
 return componentes

### Passo 3: Reconstrução Lógica (Grafo)

Aqui, simplificamos a heurística de conexão para o MVP: conectamos componentes baseados na distância euclidiana mínima, assumindo que componentes muito próximos ou alinhados têm alta probabilidade de interação. Em uma versão final, a detecção de linhas do OpenCV seria integrada.

Python

import networkx as nx  
import math  
  
def construir\_grafo(componentes):  
 G = nx.DiGraph()  
   
 # Adiciona nós ao grafo  
 for comp in componentes:  
 G.add\_node(comp['id'], tipo=comp['tipo'], pos=comp['centro'])  
   
 # Heurística simplificada de conexão (Proximidade)  
 # Na versão avançada, substituir por detecção de setas via OpenCV  
 for i, c1 in enumerate(componentes):  
 for c2 in componentes[i+1:]:  
 dist = math.dist(c1['centro'], c2['centro'])  
 # Se a distância for menor que um threshold (ex: 200px), assume conexão  
 # Isso é uma simplificação para o MVP.  
 if dist < 200:   
 # Adiciona aresta bidirecional por padrão na dúvida  
 G.add\_edge(c1['id'], c2['id'])  
   
 return G

### Passo 4: O Avaliador de Segurança (Motor STRIDE)

Este módulo percorre o grafo gerado e aplica as regras de segurança.

Python

def avaliar\_stride(grafo):  
 relatorio =  
   
 # Mapeamento de Riscos por Tipo de Componente  
 riscos\_bd = {  
 "STRIDE": "Information Disclosure",  
 "Descricao": "Banco de Dados potencialmente exposto ou sem criptografia.",  
 "Recomendacao": "Garantir criptografia em repouso (KMS) e restringir Security Groups."  
 }  
   
 riscos\_publicos = {  
 "STRIDE": "Denial of Service (DoS)",  
 "Descricao": "Ponto de entrada público sem menção a WAF ou CDN.",  
 "Recomendacao": "Implementar AWS WAF e Shield Advanced."  
 }  
  
 for node, dados in grafo.nodes(data=True):  
 tipo = dados.get('tipo')  
   
 # Análise baseada em regras de negócio  
 if tipo in:  
 relatorio.append(f" {node}: {riscos\_bd} ({riscos\_bd})")  
   
 if tipo in ['Internet', 'User', 'Client']:  
 # Verifica o que o usuário acessa diretamente  
 for vizinho in grafo.successors(node):  
 tipo\_vizinho = grafo.nodes[vizinho]['tipo']  
 if tipo\_vizinho in:  
 relatorio.append(f" Acesso direto {node} -> {vizinho}. Risco de Spoofing/Tampering. Use uma API intermediária.")  
  
 return relatorio

## 6. Insights Avançados e Caminhos para Evolução

### 6.1 Integração com LLMs Multimodais (A Segunda Opinião)

Embora a abordagem baseada em YOLO e regras seja determinística e robusta, ela carece de "senso comum". Uma inovação poderosa é utilizar um LLM Multimodal (como GPT-4o Vision ou LLaVA rodando localmente) como um validador.

* **Workflow:** O sistema clássico gera o JSON do grafo. Enviamos a imagem original + o JSON para o LLM com o prompt: *"Analise este diagrama. O grafo JSON anexo descreve corretamente as conexões visuais? Identifique arestas faltantes ou erros de classificação."*
* Isso permite corrigir erros de OCR ou interpretação visual complexa que o YOLO possa ter perdido.34

### 6.2 Threat Modeling as Code (TaaC)

A saída do sistema não deve ser apenas um PDF estático. O ideal para uma integração DevSecOps é gerar um arquivo de "Threat Model as Code", compatível com ferramentas como o **OWASP pytm** ou **Threagile**.

* Ao exportar o grafo detectado para um script Python pytm, permitimos que o arquiteto de segurança refine o modelo gerado pela IA, adicione detalhes invisíveis (como "porta 443", "protocolo HTTPS") e rode simulações mais profundas. Isso transforma a ferramenta de um simples scanner para um assistente de produtividade.37

## 7. Conclusão e Recomendação para o Hackathon

A solução apresentada atende integralmente aos requisitos do desafio, oferecendo uma base técnica sólida que vai além do básico.

**Pontos Chave para o Sucesso do MVP:**

1. **Foco nos Dados:** Não perca tempo rotulando dados manualmente no início. Use o script de geração sintética com a biblioteca diagrams para criar um modelo YOLO inicial em horas.
2. **Robustez na Detecção de Arestas:** A maior falha potencial é conectar tudo a tudo ou não conectar nada. Invista tempo na lógica de reconstrução do grafo (etapa 4.2). Se a detecção automática de linhas falhar, permita que o usuário corrija as conexões manualmente na interface (Human-in-the-loop).
3. **Contextualização do STRIDE:** Evite alertas genéricos. Use a lógica de grafos para identificar padrões específicos de nuvem (ex: S3 público, DB sem WAF) que demonstram entendimento real da arquitetura, não apenas repetição de teoria.

Este relatório fornece o mapa completo: da teoria de segurança à implementação de redes neurais, passando pela engenharia de dados. A execução deste plano resultará em uma ferramenta de alto impacto, alinhada com as necessidades da indústria de segurança cibernética moderna.

### Tabela Resumo de Tecnologias Recomendadas

| **Componente da Solução** | **Tecnologia Recomendada** | **Justificativa Técnica** | **Fonte de Referência** |
| --- | --- | --- | --- |
| **Detecção de Objetos** | **YOLOv8** (Ultralytics) | SOTA em velocidade/precisão, fácil fine-tuning. | 22 |
| **Geração de Dados** | **Diagrams** (Python) + Graphviz | Geração massiva de dados rotulados automaticamente. | 11 |
| **Modelagem de Grafos** | **NetworkX** | Padrão da indústria para manipulação de estruturas de rede. | 40 |
| **Processamento de Imagem** | **OpenCV** (cv2) | Detecção heurística de linhas, setas e pré-processamento. | 27 |
| **Lógica de Ameaças** | **OWASP pytm** (Conceito) | Framework base para regras "Threat Modeling as Code". | 31 |
| **Dataset Base** | **AWS Icon Detector** | Base sólida de imagens reais para complementar sintéticas. | 16 |

#### Referências citadas

1. Hackaton IADT.pdf
2. ArchCAD-400K: An Open Large-Scale Architectural CAD Dataset and New Baseline for Panoptic Symbol Spotting - arXiv, acessado em janeiro 6, 2026, <https://arxiv.org/html/2503.22346v1>
3. Software Architecture Meets LLMs: A Systematic Literature Review - arXiv, acessado em janeiro 6, 2026, <https://arxiv.org/html/2505.16697v1>
4. STRIDE Threat Model: A Complete Guide - Jit.io, acessado em janeiro 6, 2026, <https://www.jit.io/resources/app-security/stride-threat-model-a-complete-guide>
5. Threat modeling STRIDE methodology - IriusRisk, acessado em janeiro 6, 2026, <https://www.iriusrisk.com/resources-blog/threat-modeling-methodology-stride>
6. STRIDE model - Wikipedia, acessado em janeiro 6, 2026, <https://en.wikipedia.org/wiki/STRIDE_model>
7. Uncover Security Design Flaws Using The STRIDE Approach | Microsoft Learn, acessado em janeiro 6, 2026, <https://learn.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach>
8. Using the STRIDE Threat Model: Tutorial & Best Practices - Drata, acessado em janeiro 6, 2026, <https://drata.com/grc-central/risk/guide-stride-threat-model>
9. Threat Modeling Tool 2016 User Guide.docx - Microsoft Download Center, acessado em janeiro 6, 2026, <https://download.microsoft.com/download/4/F/D/4FDDEA98-4ABD-47A7-AA0E-815CE8660A76/Threat%20Modeling%20Tool%202016%20User%20Guide.docx>
10. Training YOLOv8 with Synthetic Data from Falcon - Duality Robotics, acessado em janeiro 6, 2026, <https://www.duality.ai/blog/training-yolov8-with-synthetic-data>
11. Synthetic Data Generation: A Hands-On Guide in Python - DataCamp, acessado em janeiro 6, 2026, <https://www.datacamp.com/tutorial/synthetic-data-generation>
12. mingrammer/diagrams: :art: Diagram as Code for prototyping cloud system architectures - GitHub, acessado em janeiro 6, 2026, <https://github.com/mingrammer/diagrams>
13. Diagrams · Diagram as Code, acessado em janeiro 6, 2026, <https://diagrams.mingrammer.com/>
14. How to get nodes coordinates in graph rendered with graphviz - Stack Overflow, acessado em janeiro 6, 2026, <https://stackoverflow.com/questions/57289762/how-to-get-nodes-coordinates-in-graph-rendered-with-graphviz>
15. Synthetic dataset generation for object-to-model deep learning in industrial applications, acessado em janeiro 6, 2026, <https://pmc.ncbi.nlm.nih.gov/articles/PMC7924434/>
16. AWS Icon Detector Object Detection Dataset by AWS Icons - Roboflow Universe, acessado em janeiro 6, 2026, <https://universe.roboflow.com/aws-icons/aws-icon-detector>
17. AWS Icon Detection Object Detection Model by Steven Kuo - Roboflow Universe, acessado em janeiro 6, 2026, <https://universe.roboflow.com/steven-kuo-5s6aq/aws-icon-detection>
18. azure icons Object Detection Model by Saugat - Roboflow Universe, acessado em janeiro 6, 2026, <https://universe.roboflow.com/saugat/azure-icons>
19. How to Use the azure icons Detection API - Roboflow Universe, acessado em janeiro 6, 2026, <https://universe.roboflow.com/saugat/azure-icons/model/1>
20. Activity Diagrams Object Detection Dataset by Public1 - Roboflow Universe, acessado em janeiro 6, 2026, <https://universe.roboflow.com/public1/activity-diagrams-s7sxv>
21. Fine-tune large multimodal models using Amazon SageMaker | Artificial Intelligence - AWS, acessado em janeiro 6, 2026, <https://aws.amazon.com/blogs/machine-learning/fine-tune-large-multimodal-models-using-amazon-sagemaker/>
22. Train YOLOv8 on Custom Dataset – A Complete Tutorial, acessado em janeiro 6, 2026, <https://learnopencv.com/train-yolov8-on-custom-dataset/>
23. Object Detection: Models, Architectures & Tutorial [2024] - V7 Go, acessado em janeiro 6, 2026, <https://www.v7labs.com/blog/object-detection-guide>
24. Training YOLOv8 on Custom Data - DigitalOcean, acessado em janeiro 6, 2026, <https://www.digitalocean.com/community/tutorials/yolov8>
25. YOLO Algorithm for Object Detection Explained [+Examples] - V7 Go, acessado em janeiro 6, 2026, <https://www.v7labs.com/blog/yolo-object-detection>
26. How I Built a Computer Vision Pipeline to Extract Structure from flowchart Diagrams | by Wasif Ullah | Medium, acessado em janeiro 6, 2026, <https://medium.com/@wasifullahdev/how-i-built-a-computer-vision-pipeline-to-extract-structure-from-flowchart-diagrams-55173df64871>
27. Analyzing a hand-drawn flowchart diagram - opencv - Stack Overflow, acessado em janeiro 6, 2026, <https://stackoverflow.com/questions/17947587/analyzing-a-hand-drawn-flowchart-diagram>
28. How to detect different types of arrows in image? - Stack Overflow, acessado em janeiro 6, 2026, <https://stackoverflow.com/questions/66718462/how-to-detect-different-types-of-arrows-in-image>
29. Arrow R-CNN for handwritten diagram recognition - MADOC - Uni Mannheim, acessado em janeiro 6, 2026, <https://madoc.bib.uni-mannheim.de/58430/1/Sch%C3%A4fer2021_Article_ArrowR-CNNForHandwrittenDiagra.pdf>
30. [2505.19944] Can Visual Encoder Learn to See Arrows? - arXiv, acessado em janeiro 6, 2026, <https://arxiv.org/abs/2505.19944>
31. OWASP pytm, acessado em janeiro 6, 2026, <https://owasp.org/www-project-pytm/>
32. pytm - OWASP Developer Guide, acessado em janeiro 6, 2026, <https://devguide.owasp.org/en/04-design/01-threat-modeling/02-pytm/>
33. OWASP/pytm: A Pythonic framework for threat modeling - GitHub, acessado em janeiro 6, 2026, <https://github.com/OWASP/pytm>
34. Understanding Architecture Diagrams using Vision LLMs: A Comparative Study of Four Leading Visual LLMs | by Shubham Karwa | Medium, acessado em janeiro 6, 2026, <https://medium.com/@cout.shubham/understanding-architecture-diagrams-using-vision-llms-a-comparative-study-of-four-leading-visual-5de6f8906cc3>
35. mrwadams/stride-gpt: An AI-powered threat modeling tool that leverages OpenAI's GPT models to generate threat models for a given application based on the STRIDE methodology. - GitHub, acessado em janeiro 6, 2026, <https://github.com/mrwadams/stride-gpt>
36. Experiments with gpt-4o vision and architecture diagrams - Dhaval Singh's Blog, acessado em janeiro 6, 2026, <https://www.dsdev.in/experiments-with-gpt-4o-vision-and-architecture-diagrams>
37. Threat Modeling as Code: Implementing STRIDE in DevSecOps | by Urshila - Medium, acessado em janeiro 6, 2026, <https://medium.com/@urshilaravindran/threat-modeling-as-code-implementing-stride-in-devsecops-0dd9f9ff06c1>
38. From Manual to Automated: A Deep Dive into Threat Modeling as Code with Python, acessado em janeiro 6, 2026, <https://dev.to/ellipse2v/from-manual-to-automated-a-deep-dive-into-threat-modeling-as-code-with-python-1j1f>
39. Continuous Threat Modeling and Threat Modeling as Code (TMasC) - DEV Community, acessado em janeiro 6, 2026, <https://dev.to/vaib/continuous-threat-modeling-and-threat-modeling-as-code-tmasc-43c8>
40. Introduction — NetworkX 3.6.1 documentation, acessado em janeiro 6, 2026, <https://networkx.org/documentation/stable/reference/introduction.html>
41. Morphological Transformations - OpenCV, acessado em janeiro 6, 2026, <https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html>