FIAP

FIAP – FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA

MODELO AUTOMATIZADO DE THREAT MODELING – STRIDE

SÃO PAULO – 2025

RESUMO

Este trabalho apresenta um protótipo capaz de extrair automaticamente componentes de diagramas de arquitetura de software, classificar o tipo de solução e elaborar, de acordo com a metodologia STRIDE, um relatório de ameaças e contramedidas em formato PDF. Descrevem-se a metodologia empregada, os resultados obtidos e sugestões de melhorias futuras.

1 INTRODUÇÃO

A modelagem de ameaças constitui etapa fundamental do desenvolvimento seguro, mas exige tempo e conhecimento especializado. Com o uso de técnicas de Machine Learning (ML) e Visão Computacional, buscou-se automatizar parte desse processo, reduzindo o esforço das equipes e padronizando os entregáveis em ambientes acadêmicos e corporativos.

2 METODOLOGIA

2.1 Dataset

Para treinar o modelo de classificação, foi utilizado o arquivo dataset_augmented_v4.csv, que contém 70 instâncias. Cada linha representa uma arquitetura de sistema, composta por um conjunto de componentes e o respectivo rótulo de categoria.

Por exemplo:

```
tipo,componentes
serverless,"user,api_gateway,lambda,dynamodb"
```

Neste exemplo, a classe serverless é associada a um conjunto de quatro componentes que, juntos, caracterizam esse tipo de arquitetura. A coluna tipo é utilizada como variável de saída (target), enquanto a coluna componentes serve como entrada para o processo de treinamento.

O conjunto foi construído manualmente e expandido com variações típicas, simulando diferentes cenários reais de arquiteturas em nuvem. A diversidade dos dados visa garantir que o modelo reconheça padrões comuns de agrupamento entre os componentes.

2.2 Treinamento do Classificador

O processo de treinamento do modelo de classificação foi dividido em três etapas: vetorização das entradas, ajuste do modelo e persistência dos artefatos treinados.

Vetorização – CountVectorizer

Inicialmente, os dados textuais foram transformados em representações numéricas. Para isso, utilizou-se a técnica de bag-of-words através da ferramenta CountVectorizer. Essa abordagem analisa todos os componentes existentes no dataset e constrói um vocabulário global. Em seguida, cada instância é convertida em um vetor binário ou de contagem, onde cada posição indica se determinado componente está presente.

Exemplo:
A entrada user, api_gateway, lambda se transforma em um vetor como [1, 1, 1, 0, 0, ...], onde os valores indicam a presença ou ausência dos componentes no vetor de referência.

Ajuste do modelo – Multinomial Naïve Bayes

Após a vetorização, os dados foram utilizados para treinar um modelo do tipo Multinomial Naïve Bayes, uma técnica estatística simples e amplamente usada para problemas de classificação de texto. Esse algoritmo calcula a probabilidade de uma instância pertencer a cada uma das classes com base na frequência dos componentes.

A principal vantagem desse modelo está na sua eficiência: ele lida bem com entradas esparsas, é fácil de treinar e apresenta bons resultados em contextos com textos curtos e vocabulário delimitado — como é o caso deste projeto.

Durante o processo de predição, o modelo avalia os componentes fornecidos e retorna a classe arquitetural mais provável com base nos padrões aprendidos.

Persistência – Salvando o modelo

Ao final do treinamento, tanto o modelo quanto o vetorizador são salvos em arquivos (modelo_treinado.pkl e vectorizer.pkl) utilizando a biblioteca joblib. Isso permite que o pipeline de classificação seja reutilizado futuramente sem a necessidade de reprocessar os dados ou reexecutar o treinamento, tornando o sistema mais leve e eficiente.

3 RESULTADOS

Durante a fase de validação, foram testadas mais de **70 imagens de diagramas** por meio do serviço **Custom Vision**, utilizando uma variedade de arquiteturas reais extraídas de documentações oficiais da **AWS** e **Azure**. A partir desses testes, foi possível gerar um **dataset ampliado e robusto**, contendo os principais componentes de sistemas distribuídos, devidamente rotulados com suas respectivas categorias arquiteturais.

Após o treinamento, o modelo mostrou-se capaz de identificar com precisão **todos os componentes previamente catalogados no dataset**, permitindo a geração automatizada de relatórios com base na metodologia **STRIDE**. Esses relatórios incluíram ameaças potenciais, contramedidas e riscos associados a cada componente reconhecido.

No entanto, verificou-se uma limitação importante: **componentes que não estavam presentes no dataset de treinamento** ou que **não foram corretamente catalogados**, não foram reconhecidos pelo sistema durante o processo de OCR e classificação. Como consequência, esses elementos ausentes **não foram incluídos no relatório final**, o que pode comprometer a análise de ameaças em arquiteturas incompletas.

Esse comportamento reforça a importância de ampliar continuamente a base de dados de treinamento com **diagramas variados e atualizados**, a fim de garantir maior abrangência e cobertura do modelo em ambientes reais e complexos.

4 CONCLUSÃO

Os resultados indicam que a automação da modelagem STRIDE é viável e traz benefícios em cenários acadêmicos e corporativos. Como trabalhos futuros pretende-se substituir serviços proprietários por soluções open-source, adotar modelos baseados em transformers e incluir métricas de severidade gráfica.

REFERÊNCIAS

MICROSOFT. Threat Modeling: STRIDE Approach. 2022.

REPORTLAB. ReportLab User Guide. 2024.

FIAP. Hackaton 3IADT – 2025. Regulamento.