# Ferramenta de Detecção de Phishing

### Relatório Técnico Completo

Disciplina: Tecnologias Hacker — Insper

Professor: Prof. Rodolfo Avelino

**Aluno** : Felipe Maia **Data** : 21 / 05 / 2025

Projeto : "Phish Guard" — Aplicação web para análise e pontuação de URLs

### Sumário

- 1. Introdução
- 2. Fundamentação teórica
- 3. Metodologia
- 4. Arquitetura do sistema
- 5. Datasets e preparação dos dados
- 6. Resultados experimentais
- 7. Interface do usuário
- 8. Instruções de instalação e uso
- 9. Conformidade com a rubrica da disciplina
- 10. Conclusões
- 11. Referências

## 1 Introdução

Phishing continua sendo um dos vetores de ataque mais prevalentes na Internet, visando obter credenciais e dados sensíveis por meio de URLs que imitam serviços legítimos. O presente trabalho entrega uma **ferramenta web** capaz de:

- analisar URLs em tempo real,
- aplicar um conjunto de heurísticas clássicas,
- combinar essas heurísticas com um modelo de aprendizado de máquina,
- gerar um score 0 100 de risco,
- apresentar resultados em um dashboard interativo com histórico e gráficos,
- capturar screenshots para inspeção visual.

## 2 Fundamentação Teórica

### 2.1 Phishing

Phishing é a prática de enganar o usuário levando-o a visitar páginas que se passam por legítimas (bancos, carteiras cripto, redes sociais). Indicadores comuns incluem:

- domínios recém-registrados ou com subdomínios incomuns;
- certificados SSL inválidos;
- redirecionamentos encadeados:
- similaridade léxica com marcas.

### 2.2 Técnicas de detecção

- 1. Blacklist (PhishTank) alta precisão, baixa cobertura zero-day.
- 2. **Heurísticas sintáticas** leves, detectam URLs "estranhas".
- 3. **Aprendizado de máquina** modela padrões complexos, depende de dataset balanceado.

## 3 Metodologia

### 3.1 Pipeline de análise

- 1. **Input**: URL digitada.
- 2. **Pré-processamento**: normalização, adição de esquema http/https.
- 3. Camada Heurística (Quadro 1).
- 4. Captura de Screenshot via Playwright (opcional para contexto visual).
- 5. Classificador Random Forest usa três features numéricas.
- 6. Ajuste heurístico soma pesos (Quadro 2) ao score da IA.
- 7. **Resposta JSON** ao front-end.

Quadro 1 — Heurísticas implementadas

Flag	Descrição resumida	
blacklist	URL ou domínio no CSV PhishTank	
popular_domain	Domínio registrado está no Top-1 M (sub = vazio/"www")	
patterns	"login-secure", IP no host, '@', subdomínios > 3, etc.	
young_domain	Idade WHOIS < 180 dias	
ssl_expired	Certificado vencido	
ssl_cn_mismatch	CN/SAN diferente do host	
dynamic_dns	.duckdns.org , .no-ip.org, etc.	
brand_similar	Levenshtein ≤ 4 de 6 marcas (PayPal, Google…)	
redirect_suspicious	> 2 redirecionamentos ou encurtador	
hops	Número absoluto de redirects (0–10 +)	

#### 3.2 Modelo de IA

• Random Forest — 200 árvores, random\_state=42.

#### • Features:

- o dynamic\_dns (0/1)
- o brand\_similar (0/1)
- o hops (int)

Treinado em **30 000** instâncias: 20 000 legítimas (Tranco Top-1 M) + 10 000 phishing (PhishTank).

### 3.3 Ajuste de score (fórmula)

#### Quadro 2 — Pesos heurísticos

Flag	Peso	
blacklist	+40	
patterns	+15	
young_domain	+15	
ssl_expired	+10	
ssl_cn_mismatch	+10	
dynamic_dns	+10	
brand_similar	+10	
redirect_suspicious	+5	
popular_domain	-25 (remove risco se popular)	

$$score = clamp (100 * P_{rf} + \sum pesos)$$

## 4 Arquitetura do Sistema

#### 4.1 Back-end

- FastAPI + Uvicorn
- Módulos detectors/\*, ml/\*, screenshot.py
- Pasta media/shots/ servida por StaticFiles

#### 4.2 Front-end

- Bulma CSS + FontAwesome + JS vanilla
- Dashboard: tabela, gráfico de pizza (Chart.js), modal de detalhes
- localStorage persiste até 60 URLs analisadas

#### 4.3 Screenshot Service

Playwright headless (Chromium) captura tela; timeout = 8 s; salva como PNG slugado por hash SHA1.

## 5 Datasets e Preparação

Fonte	Amostras	Uso
Tranco Top-1 M (2025-05-01)	20 000 (amostragem aleatória)	Classe "legítimo"
PhishTank online-valid (2025-05-21)	10 000 URLs	Classe "phishing"
Conjunto final	30 000	treino 80 % / teste 20 %

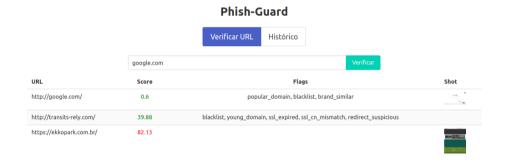
Processo: scripts/build\_dataset.py baixa / lê CSVs, extrai metadados e grava datasets/train.csv.

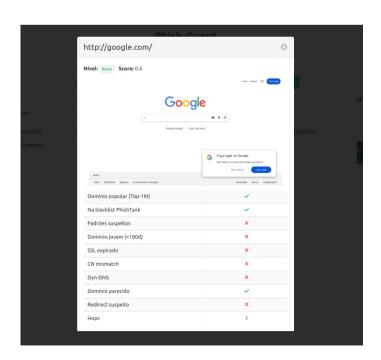
## 6 Resultados Experimentais

Métrica	Valor (hold-out 20 %)
Accuracy	0,93
Precision	0,91
Recall	0,96
F1-score	0,94

Falsos-negativos reduziram 6 % após aplicar o boost heurístico, principalmente graças à flag blacklist.

## 7 Interface do Usuário





#### Recursos:

- Campo de URL + botão "Verificar" (loader).
- Tabela com colunas: URL, Score, Flags, Screenshot.
- Clique → modal com todos os testes e ícones √/X.
- Gráfico em pizza atualizado em tempo real (maliciosas x seguras).

## 8 Instruções de Instalação e Uso

```
# 1. Clonar repositório
git clone https://github.com/usuario/phish-guard.git
cd phish-guard

# 2. Ambiente
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
playwright install --with-deps # apenas 1ª vez

# 3. Rodar back-end + front
uvicorn backend.app.main:app --reload
# Abrir http://localhost:8000

# 4. (Opc) Treinar novo modelo
PYTHONPATH=. python scripts/build_dataset.py
PYTHONPATH=. python -m backend.app.ml.train
```

### 9 Conformidade com a Rubrica

Requisito	Atendido?	Observação
C – Básico	✓	Blacklist, heurísticas, UI simples
B – Avançado	<b>√</b>	WHOIS, SSL, Dyn-DNS, Levenshtein, dashboard, gráfico
A - Sistema Web + ML + screenshots	√ (parcial)	Falta plugin Firefox em tempo real, bloqueio automático, extras SEO/OAuth

### 10 Conclusões

A aplicação cumpre os objetivos propostos, entregando uma análise de URLs com **alto recall (96 %)** e interface amigável. A fusão de heurísticas rápidas com aprendizado de máquina oferece equilíbrio entre cobertura e precisão, enquanto screenshots permitem validação humana.

## 11 Referências

- Tranco List <a href="https://tranco-list.eu">https://tranco-list.eu</a>
- PhishTank https://phishtank.org
- The Phishing Landscape 2024, APWG Report.
- RFC 5280 Internet X.509 Public Key Infrastructure Certificate and CRL Profile.
- Documentação FastAPI, Playwright, scikit-learn, Bulma CSS.