# Programa de Pentest Automatizado e Investigação OSINT

## 1. Introdução

Este projeto visa desenvolver uma ferramenta multifuncional para auxiliar em testes de penetração (pentest) e investigações de inteligência de código aberto (OSINT). O programa está sendo construído em fases, agregando funcionalidades progressivamente.

* **Fase 1:** Implementação de um scanner de rede básico via CLI, capaz de identificar portas abertas e serviços, e simular explorações simples (SSH, HTTP).
* **Fase 2:** Desenvolvimento de uma Interface Gráfica do Usuário (GUI) com PyQt5 para visualização de processos, animações simuladas e uma lógica de decisão colaborativa, onde o programa sugere ações e aguarda confirmação do usuário.
* **Fase 3 (Em Andamento):** Integração de módulos OSINT para coleta extensiva de dados sobre alvos, utilizando ferramentas como theHarvester, e preparação para um módulo de investigação com visualização de dados em grafo.
* **Fase 4 (Planejada):** Desenvolvimento de uma IA auxiliar para guiar o operador, com capacidades de autoaprendizagem e autodesenvolvimento (com autorização do usuário).

## 2. Estrutura de Diretórios (Principal)

/home/ubuntu/pentest\_project/

├── main.py # Ponto de entrada para a CLI da Fase 1

├── gui.py # Ponto de entrada para a GUI da Fase 2

├── decision\_logic.py # Lógica de decisão para a GUI da Fase 2

├── modules/

│ ├── network\_scanner.py # Módulo de varredura de rede (Nmap)

│ └── harvester\_collector.py # Coletor OSINT para theHarvester (Fase 3)

├── exploit\_modules/ # Módulos de "exploração" simulados (Fase 1)

│ ├── \_\_init\_\_.py

│ ├── http\_exploiter.py

│ └── ssh\_exploiter.py

├── osint\_module/ # Módulos para coleta OSINT (Fase 3)

│ ├── \_\_init\_\_.py

│ ├── base\_collector.py # Classe base para coletores OSINT

│ └── collectors/ # Coletores OSINT específicos (ex: spiderfoot\_collector.py - a ser criado)

├── investigation\_module/ # Módulo de investigação (Fase 3 - a ser desenvolvido)

├── utils/ # Utilitários (Fase 3 - a ser desenvolvido)

├── harvester\_results/ # Diretório para resultados do theHarvester

├── README.md # Este arquivo

├── arquitetura\_gui\_logica.md # Documento de arquitetura da GUI (Fase 2)

├── osint\_tools\_evaluation.md # Avaliação de ferramentas OSINT (Fase 3)

└── architecture\_plan.md # Plano de arquitetura para OSINT e Investigação (Fase 3)

Documentos de planejamento (todo.md, todo\_fase2.md, todo\_fase3.md) também estão no diretório /home/ubuntu/.

## 3. Requisitos e Instalação de Dependências

O projeto é desenvolvido em Python 3.

**Dependências Gerais:**

1. **Python 3.11+**
2. **Nmap:** Necessário para o network\_scanner.py.

sudo apt-get update

sudo apt-get install -y nmap

1. **theHarvester:** Necessário para o harvester\_collector.py.
   * Siga as instruções de instalação do repositório oficial: <https://github.com/laramies/theHarvester>
   * Geralmente envolve clonar o repositório e instalar dependências Python:

git clone https://github.com/laramies/theHarvester.git

cd theHarvester

python3 -m pip install -r requirements.txt

# Certifique-se de que theHarvester está no seu PATH ou chame-o com o caminho completo.

**Dependências Python (instalar via pip3):**

python3 -m pip install python-nmap requests beautifulsoup4 PyQt5

* python-nmap: Para interagir com o Nmap via Python.
* requests: Para requisições HTTP (usado no http\_exploiter.py).
* beautifulsoup4: Para parsing HTML (usado no http\_exploiter.py).
* PyQt5: Para a Interface Gráfica do Usuário (GUI da Fase 2).

## 4. Como Executar

Navegue até o diretório do projeto: cd /home/ubuntu/pentest\_project

### a. Módulo CLI (Fase 1)

O script main.py orquestra a varredura de rede e a simulação de exploração.

python3 main.py

Por padrão, ele está configurado para escanear localhost. Você pode modificar o alvo diretamente no arquivo main.py.

### b. Módulo GUI (Fase 2)

O script gui.py inicia a interface gráfica.

python3 gui.py

**Importante (Sobre a GUI):** A execução da GUI com PyQt5 pode falhar no ambiente sandbox atual devido à ausência do plugin Qt xcb. Para visualizar e interagir com a GUI, é recomendado executar o gui.py em um ambiente desktop Linux com um servidor X funcional, ou em Windows/macOS com as dependências PyQt5 corretamente instaladas.

### c. Módulos OSINT (Fase 3 - Exemplo com HarvesterCollector)

O harvester\_collector.py pode ser executado individualmente para testes. Ele tentará usar a ferramenta theHarvester instalada no sistema.

python3 ./modules/harvester\_collector.py

Este script de exemplo tentará coletar dados para example.com e salvará os resultados em /home/ubuntu/pentest\_project/harvester\_results/example.com\_harvester\_results.json.

## 5. Descrição dos Módulos Principais

* **main.py**: Orquestrador da funcionalidade CLI da Fase 1. Realiza varredura de rede e aciona módulos de exploração simulados.
* **modules/network\_scanner.py**: Utiliza python-nmap para realizar varreduras de portas e identificar serviços.
* **exploit\_modules/**: Contém scripts que simulam tentativas de exploração para serviços como SSH e HTTP.
* **gui.py**: Implementa a interface gráfica da Fase 2 usando PyQt5. Apresenta informações, simula animações de progresso e interage com uma lógica de decisão colaborativa.
* **decision\_logic.py**: Contém a lógica de decisão usada pela GUI para sugerir próximos passos ao usuário.
* **osint\_module/base\_collector.py**: Classe abstrata base para futuros coletores OSINT, definindo uma interface comum.
* **modules/harvester\_collector.py**: Implementação de um coletor OSINT que utiliza a ferramenta theHarvester para reunir informações sobre um domínio.

## 6. Próximos Passos e Evolução

* **Fase 3 (Continuação):**
  + Implementar mais coletores OSINT (SpiderFoot, Shodan API, Photon, etc.).
  + Desenvolver o módulo de investigação para processar, correlacionar e armazenar os dados OSINT coletados.
  + Integrar a visualização de dados em grafo (ex: com NetworkX e Graphviz/Matplotlib) na GUI para exibir as relações descobertas.
  + Expandir a lógica de decisão para utilizar os dados OSINT e sugerir ações investigativas ou de exploração mais complexas.
* **Fase 4 (Planejada):**
  + Desenvolver um módulo de Inteligência Artificial (IA) para auxiliar o operador.
  + Capacitar a IA com aprendizado para sugerir estratégias de pentest/investigação.
  + Implementar mecanismos para que a IA possa se autoaprimorar (com supervisão e autorização do usuário).

Este documento serve como um guia de referência para o estado atual do projeto e seus planos futuros.