**RELATÓRIO TÉCNICO - PROJETO DE CAPTURA E REGISTRO DE PACOTES**

**1 ESTRUTURA DO PROJETO**

**1.1 Scripts Utilitários**

* **startup.bar / startup.sh**
  + Inicializam o programa (src/main.py) em uma janela dedicada chamada “PRINCIPAL”.
  + Oferece mecanismos de parada: Tecla T no Windows e Ctrl+C no Linux.
  + O startup.sh ainda valida se o usuário possui permissões de superusuário e dependências instaladas (pyftpdlib e scapy).
* **PEP8.bat / PEP8.sh**
  + Validam e formatam código conforme PEP8 (Um guia de estilo para código Python.).
  + Dependências:
    - isort: organiza imports.
    - black: formata o código.
    - flake8: verifica discrepâncias de estilo.

**1.2 Módulo src (código fonte)**

* **ip.py**
  + Função get\_local\_ip(): retorna e imprime o IP local da máquina.
  + Utiliza socket.gethostname() para obter o nome da máquina e socket.gethostbyname() para resolver esse nome em um endereço IP local.
  + Em caso de falha, encerra com erro.
* **netlog.py**
  + Classe NetLogger: responsável pela captura de pacotes e gravação em .csv.
  + Funcionalidades:
    - Captura pacotes via Scapy.
    - Armazena tráfego associado à IPs conectados nos servidores HTTP e FTP.
    - Gera registros contendo data/hora, protocolo, bytes enviados e recebidos.
    - Trata interrupções (SIGINT), que captura e gerencia sinais de interrupção, como Ctrl+C.
* **servers.py**
  + Classe Server: instancia servidores HTTP e FTP em threads distintas.
  + Utilizando:
    - http.server (HTTP).
    - pyftpdlib (FTP).
  + Mantém conjunto de IPs conectados, exposto para o NetLogger.
* **main.py**
  + Responsável por orquestrar a execução:
    - Inicializa Server e NetLogger em threads.
    - Configura logs e sinais de interrupção.
    - Sincroniza a execução até o término.

**1.3 Testes Automatizados (Pytest):**

* test\_ip.py
  + Testes de sucesso e falha em get\_local\_ip e execução do script.
  + Usa unittest.mock e pytest.raises para simulações.
* test\_netlog.py
  + Criação de pacotes falsos (MagicMock).
  + Testes de escrita CSV e controle de interrupções.

**2 DIFICULDADES ENCONTRADAS**

**2.1 Documentação da Biblioteca Scapy**

* A documentação oficial apresenta exemplos desatualizados e pouco detalhados.
* Isso dificultou a implementação de captura filtrada de pacotes.

**2.2 Exportação de Dados Para Análise**

* O Excel não permite importar em intervalos de captura inferiores a 1 minuto.
* Foi necessário ajustar o código para registrar dados em intervalos de 5 segundos no CSV, garantindo granularidade aceitável para análise.

**2.3 Problemas com testes automatizados**

* Dificuldades no uso de assert em testes que dependiam de exceções.
* Em alguns casos, falhas não eram corretamente capturadas pelo pytest.raises.
* Exigiu ajustes em mocks e tratamento explícito de erros.