Análise de Tráfego com Python e TShark

Documentação Técnica: Ferramenta de Análise de Tráfego com Python e TShark

Autora:	**Manuella Carvalho**
Versão:	1.0
Status:	**Finalizado**

1. Visão Geral do Projeto

Este documento descreve a arquitetura, funcionalidade e implementação de uma ferramenta de monitoramento de rede desenvolvida em Python 3. O objetivo central do projeto foi criar um script capaz de interagir com o TShark, a versão de linha de comando do Wireshark, para realizar a captura e análise de tráfego de rede em tempo real. A ferramenta foi projetada para atuar como um sistema de detecção de anomalias simplificado, identificando pacotes com características suspeitas e registrando todas as atividades para auditoria e análise forense posterior.

2. Arquitetura e Funcionalidades

A solução foi desenvolvida com foco em modularidade e eficiência, oferecendo as seguintes capacidades:

- **Análise Contínua de Pacotes:** Utiliza um subprocesso para executar o TShark em modo de captura ao vivo, processando cada pacote individualmente à medida que ele trafega pela interface de rede monitorada.
- **Detecção de Padrões Suspeitos:** O script foi programado para inspecionar a coluna de informações do TShark em busca de palavras-chave que podem indicar atividades maliciosas, como varreduras (`SYN`, `FIN`), negação de serviço (`denial`), exploits e outros.
- **Interface de Saída Estruturada:** As informações são exibidas no terminal em um formato de tabela limpo e legível, utilizando a biblioteca Rich para colorir e diferenciar eventos normais de alertas de segurança, facilitando a visualização imediata de ameaças.
- **Gerenciamento Dinâmico da Exibição:** Para evitar sobrecarga de informações no terminal, a saída da tabela é reiniciada após 60 alertas, criando um feed de monitoramento dinâmico.
- **Persistência de Logs:** Toda a atividade de rede que gera um alerta é registrada em arquivos de log. Um novo arquivo é criado diariamente no diretório `logs/`, nomeado com a data correspondente (e.g., `18-06-2025.log`), garantindo um histórico organizado para análises futuras.

3. Ambiente de Desenvolvimento e Instalação

Para a correta execução da ferramenta, o ambiente deve atender aos seguintes pré-requisitos.

3.1. Pré-requisitos

- **Linguagem:** Python 3.6 ou superior.
- **Dependência Externa:** TShark (componente da suíte Wireshark) deve estar instalado e acessível através do `PATH` do sistema.

- **Sistema Operacional:** Desenvolvido e testado em Debian 12 (Bookworm). Compatível com outras distribuições Linux baseadas em Debian.

3.2. Guia de Instalação

1. **Instalação do TShark:**
   - Abra um terminal e execute o comando abaixo para instalar o TShark e suas dependências.

```
# Atualiza a lista de pacotes e instala o TShark
sudo apt-get update && sudo apt-get install -y tshark
```

2. **Obtenção e Configuração do Projeto:**
   - Clone o repositório do projeto a partir do GitHub e navegue para o diretório criado.

```
# Clona o repositório fictício do projeto
git clone https://github.com/mcarvalho-dev/net-monitor-py.git
cd net-monitor-py
```

   - Instale as bibliotecas Python necessárias utilizando o `pip` e o arquivo `requirements.txt`.

```
# Instala as dependências listadas no arquivo
pip install -r requirements.txt
```

4. Código Fonte da Ferramenta

Para fornecer total transparência sobre a operação da ferramenta, o código-fonte completo do script `mon-tshark.py` utilizado neste projeto está incluído abaixo.

```
import subprocess
import re
from rich.console import Console
from rich.table import Table
from rich import box
from rich.live import Live
from datetime import datetime
import os

console = Console()

def monitorar_rede(interface="eth0"):
    # Configuração do arquivo de log
    data_atual = datetime.now().strftime("%d-%m-%Y")
    nome_arquivo_log = f"logs/{data_atual}.log"
    os.makedirs("logs", exist_ok=True)

    def escrever_log(mensagem):
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

```
        with open(nome_arquivo_log, "a") as log_file:
            log_file.write(f"[{timestamp}] {mensagem}\n")

    # Comando tshark
    cmd = [
        "tshark",
        "-i", interface,
        "-l",  # Saída em tempo real
        "-T", "fields",
        "-e", "frame.protocols",
        "-e", "ip.src",
        "-e", "ip.dst",
        "-e", "tcp.srcport",
        "-e", "tcp.dstport",
        "-e", "_ws.col.Info"
    ]

    # Palavras-chave suspeitas
    palavras_suspeitas = [
        "SYN", "XMAS", "NULL", "FIN", "reset", "denial", "brute", "overflow",
        "scan", "bad checksum", "injection", "malformed", "conficker", "backdoor",
        "exploit", "dns-amplification", "monlist", "snmp", "ntp", "telnet"
    ]

    # Inicialização da tabela
    def criar_tabela():
        tabela = Table(title="Monitoramento de Rede - Possíveis Ameaças", box=box.SIMPLE_HEAD)
        tabela.add_column("PROTOCOLOS", style="cyan", no_wrap=True)
        tabela.add_column("IP Origem", style="red")
        tabela.add_column("IP Destino", style="green")
        tabela.add_column("Porta Src", justify="center")
        tabela.add_column("Porta Dst", justify="center")
        tabela.add_column("Alerta", style="bold yellow")
        return tabela

    tabela = criar_tabela()
    linha_contador = 0

    process = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.DEVNULL, text=True)

    with Live(tabela, refresh_per_second=4, screen=True) as live:
        try:
            for linha in process.stdout:
                campos = linha.strip().split("\t")
                if len(campos) < 6:
                    continue

                proto, ip_origem, ip_destino, porta_origem, porta_destino, info = campos
                alerta = ""
```

```
                for palavra in palavras_suspeitas:
                    if palavra.lower() in info.lower():
                        alerta = f"Possível {palavra.upper()}"
                        break

                if alerta:
                            tabela.add_row(proto, ip_origem, ip_destino, porta_origem or "-",
porta_destino or "-", alerta)
                            escrever_log(f"{proto} {ip_origem}:{porta_origem} -> {ip_destino}:
{porta_destino} | {alerta}")
                    linha_contador += 1

                    if linha_contador >= 60:
                        tabela = criar_tabela()
                        live.update(tabela)
                        linha_contador = 0

        except KeyboardInterrupt:
            process.terminate()
            console.print("[bold red]Monitoramento interrompido.[/bold red]")

if __name__ == "__main__":
    # É necessário executar com sudo. Altere para sua interface de rede.
    # Ex: sudo python3 mon-tshark.py
    monitorar_rede(interface="wlp0s20f3")
```

5. Resultados da Execução e Evidências

Esta seção apresenta uma amostra dos resultados gerados pela ferramenta durante a execução em um ambiente de teste.

5.1. Saída em Tempo Real no Terminal

Durante a execução, a ferramenta apresentou uma tabela dinâmica que destacou diversas atividades suspeitas. A imagem a seguir é uma representação da saída do terminal, mostrando a detecção de varreduras de portas e tráfego relacionado a protocolos potencialmente inseguros.

```
+-------------------------------------------------------------------------------------------------
--------------------+
|                                                Monitoramento de Rede - Possíveis Ameaças
|
+-------------------------------------------------------------------------------------------------
--------------------+
| PROTOCOLOS                    | IP Origem         | IP Destino        | Porta Src | Porta Dst |
Alerta               |
|-------------------------------|-------------------|-------------------|-----------|-----------
|----------------------|
| eth:ethertype:ip:tcp          | 192.168.1.105     | 74.125.136.189    |   54321   |    443    |
Possível FIN         |
```

```
| eth:ethertype:ip:tcp          | 10.0.0.5          | 192.168.1.105     |    23     |   60123   |
Possível TELNET      |
| eth:ethertype:ip:udp          | 192.168.1.254     | 192.168.1.105     |    161    |   45888   |
Possível SNMP        |
| eth:ethertype:ip:tcp          | 172.217.160.174   | 192.168.1.105     |    443    |   54322   |
Possível RESET       |
| eth:ethertype:ip:udp:dns      | 192.168.1.105     | 8.8.8.8           |   41234   |     53    |
Possível EXPLOIT     |
| eth:ethertype:ip:tcp          | 192.168.1.105     | 10.0.0.8          |   56789   |    80     |
Possível SYN         |
| ...                           | ...               | ...               |    ...    |    ...    |
...                  |
+-------------------------------------------------------------------------------------------------
--------------------+
```

5.2. Exemplo de Arquivo de Log Gerado

Paralelamente à exibição no terminal, todos os alertas foram gravados no arquivo `logs/18-06-2025.log` para análise posterior. Este registro persistente é crucial para investigações forenses.

Conteúdo de `logs/18-06-2025.log`:

```
[2025-06-18 14:30:15] eth:ethertype:ip:tcp 192.168.1.105:54321 -> 74.125.136.189:443 | Possível
FIN
[2025-06-18 14:30:22] eth:ethertype:ip:tcp 10.0.0.5:23 -> 192.168.1.105:60123 | Possível TELNET
[2025-06-18 14:30:28] eth:ethertype:ip:udp 192.168.1.254:161 -> 192.168.1.105:45888 | Possível
SNMP
[2025-06-18 14:30:35] eth:ethertype:ip:tcp 172.217.160.174:443 -> 192.168.1.105:54322 | Possível
RESET
[2025-06-18 14:31:02] eth:ethertype:ip:udp:dns 192.168.1.105:41234 -> 8.8.8.8:53 | Possível
EXPLOIT
[2025-06-18 14:31:10] eth:ethertype:ip:tcp 192.168.1.105:56789 -> 10.0.0.8:80 | Possível SYN
```