Guía de Uso: `sniff_and_fingerprint.py`

Este documento detalla el funcionamiento del script `sniff_and_fingerprint.py`, describe su propósito, explicación técnica y ofrece una guía paso a paso para su ejecución y análisis de resultados. Está pensado para que los integrantes del equipo suban este archivo al repositorio en formato Word.

---

1. ¿Qué es la herramienta?

`sniff_and_fingerprint.py` es un script en Python que combina dos funcionalidades clave para pruebas de penetración en redes:

- **Captura de tráfico**: Utiliza la librería `pyshark` (CLI de Wireshark en Python) para capturar paquetes en tiempo real de una interfaz de red, filtrando aquellos que puedan contener información sensible.
- **Fingerprinting de sistema operativo**: Emplea Nmap con detección de SO (`-O`) para identificar el sistema operativo y su versión en el equipo objetivo.

Ambas tareas pueden ejecutarse simultáneamente, generando archivos PCAP y reportes en JSON con hallazgos y recomendaciones.

---

2. Propósito de la herramienta

1. **Identificar vulnerabilidades por exposición de datos**: Detectar credenciales en texto claro (HTTP Basic Auth), cookies, tokens, handshakes WPA y cabeceras sensibles.
2. **Determinar el sistema operativo**: Realizar un escaneo de fingerprinting que ayude a planificar ataques basados en exploits específicos.
3. **Automatizar flujos**: Unificar en un único comando la captura de tráfico y el escaneo de fingerprinting, optimizando tiempo y reduciendo errores manuales.

---

3. Funcionamiento a nivel técnico

Estructura y componentes principales

Componente	Descripción
ArgumentParser	Define la interfaz CLI con parámetros `--iface`, `--target` y `--duration`.
Logging	Configura registro en consola y archivo `sniff_and_fingerprint.log` con nivel `INFO` y `WARNING`.

Componente	Descripción
TrafficAnalyzer	Clase que: 1) inicia captura `LiveCapture` de Pyshark, 2) aplica filtros Wireshark, 3) procesa cada paquete con callback `_packet_handler`, 4) guarda credenciales y handshakes en JSON.
NmapScanner	Clase que: 1) ejecuta Nmap con `subprocess.run`, 2) recibe salida XML, 3) parsea con `xml.etree.ElementTree`, 4) extrae `osmatch` y guarda en JSON.
Threading	Permite correr el fingerprinting de Nmap en paralelo con la captura de tráfico.
Directorios	Crea carpetas `captures/` y `reports/` para almacenar: PCAPs, JSON de credenciales y JSON de fingerprinting.

Filtros de captura

El análisis de tráfico aplica un filtro compuesto que incluye:

```
tcp.port == 80 or tcp.port == 8080 or tcp.port == 21 or tcp.port == 23 or
tcp.port == 143 or tcp.port == 110 or tcp.port == 25 or eapol or
http.authorization or http.cookie or http.request.uri contains "token" or
http.request.uri contains "session"
```

Este filtro selecciona paquetes HTTP, FTP, Telnet, IMAP, POP3, SMTP, WPA handshakes, authorization headers, cookies y posibles tokens en URLs.

---

4. Guía paso a paso de cada script

A continuación, se describe la ejecución completa de `sniff_and_fingerprint.py`, con ejemplos y explicación detallada.

1. **Preparación del entorno**

2. Asegurarse de tener **Python 3.8+** instalado en Kali Linux.

3. Instalar dependencias:

   ```
   pip install pyshark xmltodict
   ```

4. Verificar que `nmap` y `dumpcap` (o `tcpdump`) estén disponibles en el sistema.

5. **Descarga del script**

6. Colocar `sniff_and_fingerprint.py` dentro de `scripts/python/` en el repositorio.

7. **Permisos de ejecución**

8. Aunque se invoca con `python3`, se recomienda dar permisos de lectura y escritura:

```
chmod +x scripts/python/sniff_and_fingerprint.py
```

9. **Ejecución básica**

10. Ejecutar con parámetros obligatorios:

```
python3 scripts/python/sniff_and_fingerprint.py \
  --iface eth0 \
  --target 10.0.0.5 \
  --duration 60
```

11. **Explicación**:

    - `--iface eth0`: interfaz de red donde se hará la captura.
    - `--target 10.0.0.5`: IP o rango de la máquina a fingerprintear.
    - `--duration 60`: tiempo (en segundos) de captura activa.

12. **Procesos en segundo plano**

13. Al iniciar, el script crea dos hilos:

    - **Hilo Nmap**: ejecuta fingerprinting y guarda JSON en `reports/fingerprint_<target>_<timestamp>.json`.
    - **Hilo Pyshark**: captura paquetes, aplica filtros y guarda PCAP en `captures/sniff_<iface>_<timestamp>.pcap`, además de JSON con credenciales en `reports/credentials_<iface>_<timestamp>.json`.

14. **Salida de resultados**

15. Revisar `captures/` para descargar el archivo `.pcap` y analizar con Wireshark si se desea.

16. Abrir en `reports/`:
    - `credentials_<iface>_<timestamp>.json`: listado de credenciales, handshakes y cabeceras sensibles encontradas.
    - `fingerprint_<target>_<timestamp>.json`: sistema operativo detectado, precisión y marca de tiempo.

17. Consultar `sniff_and_fingerprint.log` para detalles de mensajes y posibles errores.

18. **Interpretación de hallazgos**

19. Credenciales encontradas indican vulnerabilidad de transmisión de datos en texto claro.

20. Handshakes WPA pueden servir para ataques de fuerza bruta a la red inalámbrica.
21. Sistema operativo y versión facilitan la selección de exploits específicos de Nmap o Metasploit.

Fin de la guía

Este documento está listo para convertirse en un archivo Word y mantenerse en el repositorio como guía de uso para el equipo.