

Ejercicio guiado — Uso desde la INTERFAZ (TUI + API local)

Andres Valdivieso Pinilla

Objetivo: recorrer todas las funcionalidades mediante la interfaz TUI `ui_web.py` (con API local FastAPI) sin usar EICAR ni ejecutar malware real. Requiere que el repositorio ya esté clonado y que `main.py`, `ui_web.py` y `modules/` estén presentes.

Índice

- Disclaimer
 - Arrancar la interfaz TUI + API local
 - Ejecutar TUI (local)
 - Interfaz: descripción de opciones (TUI)
 - 4) Ejercicio paso a paso (GUI) — usar todas las opciones sin EICAR
 - PREPARACIÓN (comprobaciones)
 - Opción 1 — Forensics (Hashes)
 - Opción 2 — Malware - Strings (extract_strings)
 - Opción 3 — Malware - ClamAV
 - Opción 4 — Malware - VT Upload (VirusTotal)
 - Opción 5 — OSINT
 - Opción 6 — Salir
 - Uso remoto: API local (FastAPI)
 - Reportes: nombres, ubicación y formato único
 - Dependencias (recomendadas) y requirements.txt
 - Troubleshooting rápido (problemas vistos y soluciones)
 - UnicodeEncodeError (símbolos emoji en Windows)
 - Invalid HTTP request received / Uvicorn warnings
 - FileNotFoundError al escribir reportes por nombre inválido
 - executable not found para herramientas del sistema
 - FastAPI no instalado
 - Checklist final de validación (usa antes de entregar el ejercicio)
 - Ejemplo completo — sesión de prueba rápida (resumen)
-

Disclaimer

1. El siguiente ejercicio y los scripts asociados se proporcionan únicamente con fines educativos y para comprender el funcionamiento y las capacidades de la herramienta.
 2. Las pruebas deben realizarse en entornos controlados (máquinas de laboratorio, contenedores o imágenes aisladas) y con copias de seguridad; ejecutar comandos en entornos productivos o sin autorización puede causar pérdida de datos o interrupciones.
 3. El autor/propietario del repositorio no se hace responsable de daños, pérdidas o perjuicios derivadas del uso indebido o de la ejecución de los comandos fuera de un entorno controlado. Cada usuario es responsable de obtener las autorizaciones necesarias y de aplicar las precauciones apropiadas antes de ejecutar las pruebas.
-

Arrancar la interfaz TUI + API local

Ejecutar TUI (local)

Dentro del `venv` :
`python ui_web.py`

Salida esperada (resumen): verás logo ASCII, menú y una línea indicando la API local, p.e.
`🌐 API local disponible en http://127.0.0.1:8000 [logo ASCII] Menú Principal 1 Forensics`

Interfaz: descripción de opciones (TUI)

El menú TUI estándar muestra las opciones (ID / descripción). En tu interfaz actual las entradas son:

ID	Módulo / Acción	Descripción
1	Forensics	Generar hashes de archivos
2	Malware - Strings	Extraer strings de archivo (implementación Python)
3	Malware - ClamAV	Escaneo antivirus (clamscan)
4	Malware - VT Upload	Subir archivo a VirusTotal (requiere VT_API_KEY)
5	OSINT	Consultar dominio o IP (subdominios, IP info, email leaks)
6	Salir	Cierra la interfaz

Atajos de teclado (sugeridos y ya implementados por Prompt): puedes introducir `1..6`.

Si quieres que la UI no pida confirmación para algunas acciones, se puede ajustar; las acciones ya guardan automáticamente los outputs en `reports/` por diseño.

4) Ejercicio paso a paso (GUI) — usar todas las opciones sin EICAR

Antes de comenzar: **usa rutas relativas** en los campos de la UI, p. ej. `./test_data/notepad_sample.exe` para evitar errores de nombre de archivo en Windows (ver sección 7).

PREPARACIÓN (comprobaciones)

- `ls test_data / Get-ChildItem test_data` — confirmar `notepad_sample.exe`, `random.bin`, `wordlist.txt`.
- `ls reports` — confirmar carpeta vacía o que existe.
- Variables de entorno (opcional): si vas a probar VirusTotal o FileScan, añade sus keys (ver sección 6).

Opción 1 — Forensics (Hashes)

- Seleccionar `1` en TUI.
- Input: `Archivo para hashes` → por defecto `./test_data/notepad_sample.exe`.
- Acción: la UI ejecuta `python main.py forensics --file-hash <path>` y guarda output en `reports/`.
- Salida esperada:** MD5 / SHA1 / SHA256 impresos en la consola TUI y guardados en `reports/report_forensics_...txt` o `forensics-<file>.json` según implementación.
- Comprobar:** `cat reports/<nombre>` → confirmas que los hashes coinciden con `sha256sum` / `Get-FileHash`.

Opción 2 — Malware - Strings (extract_strings)

- Seleccionar `2`.
- Input: `Archivo para analizar con strings` → `./test_data/notepad_sample.exe` o `./test_data/random.bin`.
- Acción: ejecuta `python main.py malware --tool strings --target <path>` (o `--strings`).
- Salida esperada:** listado (muestra) de strings extraídos, count, y un reporte guardado en `reports/` (ej. `malware_tool_sysstrings.txt` o `report_malware--tool_strings_...txt`).

- **Comprobación adicional:** abrir el fichero de salida y ver strings (asegúrate de que el archivo es legible y no contiene malware real).

Opción 3 — Malware - ClamAV

- Seleccionar 3.
- Input: Archivo para ClamAV → ./test_data/notepad_sample.exe.
- Acción: ejecuta python main.py malware --tool clamscan --target <path> o --scan.
- **Requisito:** clamscan debe estar instalado en el sistema.
- **Salida esperada:** el resultado de clamscan (limpio o detección de prueba) impreso y guardado en reports/.
- **En caso de no tener clamscan:** el modulo devolverá executable not found — instala clamav o salta la prueba.

Opción 4 — Malware - VT Upload (VirusTotal)

- Seleccionar 4.
- Input: Archivo a subir a VirusTotal → ./test_data/notepad_sample.exe
- **Requisito:** configurar variable de entorno VT_API_KEY (ver sección 6).
- Acción: python main.py malware --vt-upload <path> -> sube y espera el análisis (si la API permite).
- **Salida esperada:** reporte amistoso por consola y archivo en reports/ con prefijo CASE-xxx_malware_vt_upload...json o report...txt.
- **Si no tiene key o falló:** verás VirusTotal helper missing o ERROR: upload failed. No uses EICAR; usa solo tu notepad_sample.exe.

Opción 5 — OSINT

- Seleccionar 5.
- Input: Dominio o IP para OSINT → example.com o 8.8.8.8.
- Acción: ejecuta python main.py osint --subdomains example.com o --ip-info 8.8.8.8.
- **Salida esperada:** lista de subdominios (placeholder) o información de IP. Guardado en reports/ como osint_subdomains_example.json o similar.

Opción 6 — Salir

Finalizar la interfaz. Todos los outputs ya guardados en reports/.

Uso remoto: API local (FastAPI)

La UI también expone una API local (por defecto http://127.0.0.1:8000) con un endpoint clave:

- GET /run/{module} — ejecuta un módulo:
 - Ejemplo: consultar forensics (usa target por query):
curl "http://127.0.0.1:8000/run/forensics?target=./test_data/notepad_sample.exe"
Respuesta JSON: {"stdout": "...", "stderr": "..."}
 - Malware strings:
curl "http://127.0.0.1:8000/run/malware?tool=strings&target=./test_data/notepad_sample.exe"

Nota: la API ejecuta python main.py ... internamente y devuelve stdout/stderr. Para uso remoto asegúrate de correrla solo en red segura (por defecto 127.0.0.1).

Reportes: nombres, ubicación y formato único

- **Ubicación por defecto:** `reports/` (creado automáticamente por la UI).
- **Convención de nombres que la UI usa** (ejemplos):
 - `report_forensics_--file-hash_<sanitized_target>_YYYYmmdd_HHMMSS.txt`
 - `CASE-001_malware_vt_upload_YYYY-mm-ddTHH:MM:SS.ssssss.json` (si `reporting.present_vt_report` lo guarda)
 - `malware_deepscan_<file>.json` (para deep-scan battery)
 - `osint_subdomains_<target>.json`

IMPORTANTE (Windows): evita caracteres ilegales en nombres de archivos (dos puntos `:`, barras `\` dentro del nombre).

- **Recomendación:** usar siempre rutas relativas en campos de la UI (ej. `./test_data/notepad_sample.exe`) para que el generador de nombres de report no incluya `C:\...` con `:` y `\`.
- Si aparece error `Invalid argument` al crear el nombre de fichero, revisa `reports/` y renombra manualmente o modifica el argumento `--target` para usar nombre base (p.e. `--target notepad_sample.exe` y que la UI lo construya relativo).

Dependencias (recomendadas) y `requirements.txt`

Ejemplo de `requirements.txt` mínimo (para venv pip):

```
rich fastapi uvicorn requests pefile yara-python shodan python-whois
```


Nota: `yara-python` y `pefile` pueden requerir compiladores o headers. Las utilidades como `clamscan`, `nmap`, `hydra` no se instalan con pip; instálalas via apt/brew/choco según SO.

Instalación:

```
pip install -r requirements.txt
```

Troubleshooting rápido (problemas vistos y soluciones)

`UnicodeEncodeError` (símbolos emoji en Windows)

Si ves `UnicodeEncodeError` al imprimir caracteres como , en PowerShell:

```
chcp 65001 $OutputEncoding = [Console]::OutputEncoding = [Text.UTF8Encoding]::UTF8
```

`Invalid HTTP request received` / `UVicorn warnings`

- Sucede cuando la TUI y el servidor uvicorn se mezclan en la misma consola (peticiones por stdin). Si ves muchas `Invalid HTTP request received`, ignóralas si la UI funciona; de lo contrario asegúrate de que `uvicorn` se lance en un hilo aparte como hace `ui_web.py`.

`FileNotFoundError` al escribir reportes por nombre inválido

- Evita pasar rutas absolutas con `:` y `\` como parte del nombre. Usa `./test_data/notepad_sample.exe` o solo el `basename`.
- Si ya ocurrió, crea `reports/` manualmente y renombra.

`executable not found` para herramientas del sistema

- Instala la herramienta en PATH (ej. `apt install clamav` o en Windows agrega Sysinternals a PATH).

FastAPI no instalado

```
pip install fastapi uvicorn
```

Checklist final de validación (usa antes de entregar el ejercicio)

- `venv` creado y activado.
 - `pip install -r requirements.txt` completado sin errores críticos.
 - `reports/` y `test_data/` existen y contienen archivos de prueba (no maliciosos).
 - Variables de entorno (`VT_API_KEY` / `FILESCAN_API_KEY`) definidas si se probará VT/FileScan.
 - Ejecutar `python ui_web.py` → aparece logo y `API local disponible`.
 - Ejecutar cada opción 1..5: confirmar output en pantalla y archivo guardado en `reports/`.
 - Probar endpoint HTTP `GET /run/<module>` (curl) y confirmar respuesta JSON con `stdout` y `stderr`.
 - Verificar que no aparezcan errores de codificación en consola (si aparecen, ejecutar `chcp 65001` y set `OutputEncoding`).
 - Confirmar que no se subieron archivos sensibles a servicios externos sin consentimiento.
-

Ejemplo completo — sesión de prueba rápida (resumen)

1. Activar `venv`
2. `python ui_web.py`
3. Pulsar `1` → path `./test_data/notepad_sample.exe` → comprobar hashes y `reports/`
4. Pulsar `2` → strings sobre `./test_data/notepad_sample.exe` → ver salida y `reports/`
5. Pulsar `3` → clamscan (si instalado) → ver salida
6. Pulsar `4` → vt-upload (si `VT_API_KEY` está configurada)
7. Pulsar `5` → `example.com` → ver output y `reports/`
8. Revisar `reports/` y validar que los ficheros contienen la información esperada