

# Tráfico HTTP inseguro

Para que la apertura de una imagen genere paquete(s) capturables y un registro en python -m http.server debes asegurarte de:

- Que el navegador obtenga la imagen vía **HTTP** (`http://127.0.0.1:8080/tu.jpg`) y **no** `file:///`.
- Que la captura de paquetes esté hecha en la **interfaz correcta** (loopback / Npcap Loopback Adapter si estás en Windows).
- Que la petición no esté servida desde la **caché** del navegador (forzar recarga o desactivar caché en DevTools).
- Opcional: usar curl para forzar un GET y comprobar la captura rápidamente.

## Pasos detallados (Windows)

### 1) Instala Npcap con soporte loopback (si no lo tienes)

- Descarga e instala **Npcap** (marca la opción "Support loopback traffic ('Npcap Loopback Adapter')").
- Reinicia Wireshark si estaba abierto.

### 2) Selecciona la interfaz de loopback en Wireshark

- Abre Wireshark → *Capture Options*.
- Selecciona:

**"Npcap Loopback Adapter"** (o interfaz que muestre tráfico localhost).

- (Opcional) en *Capture Filter* pon:

host 127.0.0.1 and port 8080

Eso reduce el ruido (BPF capture filter).

### 3) Desactiva o evita caché en el navegador

- Abre DevTools (F12) → pestaña **Network** → marca **Disable cache** (mientras DevTools esté abierto).
- O abre la imagen en una pestaña nueva y pulsa **Ctrl+F5**.

#### 4) Inicia la captura y luego abre la imagen por HTTP

1. En Wireshark, inicia la captura en la interfaz loopback.
2. Desde el navegador navega a:

`http://127.0.0.1:8080/lpic.jpg`

o la ruta que corresponda (usa 127.0.0.1 mejor que localhost si quieres evitar problemas IPv6).

3. Observa en el servidor python `-m http.server` si aparece un `GET /kl.jpg 200 -.` Si aparece, ya llegó la petición al servidor.

#### 5) Filtros útiles en Wireshark (display filters)

- Ver solo GET a puerto 8080:

`http.request.method == "GET" && tcp.port == 8080`

- Ver solo respuestas 200:

`http.response.code == 200 && tcp.port == 8080`

- Ver peticiones que contienen ".jpg" (URI):

`http.request && http.request.uri contains ".jpg"`

## Alternativa: usar línea de comandos (tshark / tcpdump)

Si prefieres línea:

- Captura en vivo con tshark (selecciona el índice de interfaz apropiado):

`tshark -i <iface_number> -f "host 127.0.0.1 and port 8080" -w capture.pcapng`

- Leer un pcap y mostrar GETs:

`tshark -r capture.pcapng -Y "http.request and tcp.port==8080" -T fields -e frame.number -e ip.src -e ip.dst -e http.request.method -e http.request.uri`

- Buscar POSTs:

`tshark -r capture.pcapng -Y 'http.request.method == "POST"'`

# Si estás en una VM / host distinto

- Si sirves desde la VM y el navegador está en el host (o viceversa), captura en la **máquina que origina** la petición o en la red entre ambas.
- Si el servicio corre en la VM y navegas dentro de la VM → captura dentro de la VM (o en la interfaz virtual del host que conecta con la VM).

## Puntos importantes / errores típicos

- Si abres la imagen desde el sistema de archivos (file:///...), no habrá petición HTTP ni registro en http.server.
- Si la petición se sirve desde la caché, no verás paquetes; por eso **desactivar caché** o forzar recarga es clave.
- Si en tus logs aparece ::ffff:127.0.0.1 (IPv4-mapped IPv6) – Wireshark puede mostrarlo como IPv6; usar tcp.port == 8080 es la forma más robusta de filtrar.
- python -m http.server **sí** registra GET; si no ves nada en el log, la petición no llegó al servidor.

## RECOMPONER/EXTRAER LA IMAGEN

### 1) Método fácil – Wireshark (GUI)

Para una imagen servida en la respuesta (GET → response contiene la jpg):

1. Abre el pcap en Wireshark.
2. Usa este filtro de visualización:

```
http && http.content_type contains "jpeg"
```

ó para búsquedas más amplias:

`http.request || http.response`

3. En la lista de paquetes localiza la respuesta HTTP que tiene `Content-Type: image/jpeg`.
4. Menú: **File** → **Export Objects** → **HTTP**.
  - Aparecerá una lista de objetos HTTP (archivos). Selecciona el `.jpg` y pulsa **Save**.
  - Esto te exporta el JPG ya reconstruido.

**Si no lo ves como objeto HTTP**, selecciona el paquete de la respuesta → botón derecho → **Follow** → **TCP Stream**. En la ventana que abre:

- En *Show data as* elige **Raw**.
- Guarda el contenido (Save As) con extensión `.jpg`. (Wireshark ya reensambla automáticamente si la opción está activada – ver punto 4 abajo.)

## 2) Método rápido – tshark (línea de comandos)

tshark tiene una opción muy útil para volcar objetos HTTP:

```
tshark -r captura.pcapng --export-objects "http,./export_http"
```

- Esto crea la carpeta `./export_http` con todos los objetos HTTP (incluidas las `.jpg`).
- Si sólo quieres listar las respuestas con `content-type`:

```
tshark -r captura.pcapng -Y 'http.response and http.content_type contains "jpeg"' -T fields -e frame.number -e http.content_type -e http.content_length -e http.response.code -e http.response.line
```

## 3) Si la imagen fue subida (POST multipart/form-data)

Wireshark no siempre lista las partes subidas en **Export Objects** (porque Export Objects muestra objetos *servidos*, no

necesariamente el body del request). Para extraer un fichero que viajó dentro de un POST:

1. Localiza el **paquete** donde empieza el POST (o cualquier paquete perteneciente a ese TCP stream).
2. Botón derecho → **Follow** → **TCP Stream**.
3. En la ventana "Follow TCP Stream":
  - Cambia "Show data as" a **Raw** (o Hex Dump si deseas ver en hex).
  - Guarda el contenido como upload\_raw.bin.
4. El upload\_raw.bin contiene todo el stream TCP (cabeceras HTTP + body multipart). Ahora necesitas **extraer la parte binaria** entre los boundary del multipart. Los boundary aparecen en las cabeceras Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryXXXX.

## 4) Asegúrate de que Wireshark reensamble TCP/HTTP

En Wireshark: **Edit** → **Preferences** → **Protocols** → **TCP** → activa "Allow subdissector to reassemble TCP streams".

Y en **Protocols** → **HTTP** marca "Reassemble HTTP bodies spanning multiple TCP segments". Esto garantiza que HTTP bodies largos se muestren completos y Export Objects/Follow TCP Stream te den el archivo entero.