Nombre Taller: Lab 16

Insecure File Upload 0x02

Detalles Prácticos:

Nivel(Básico/Intermedio o avanzado)
Instructores:Rieradipe

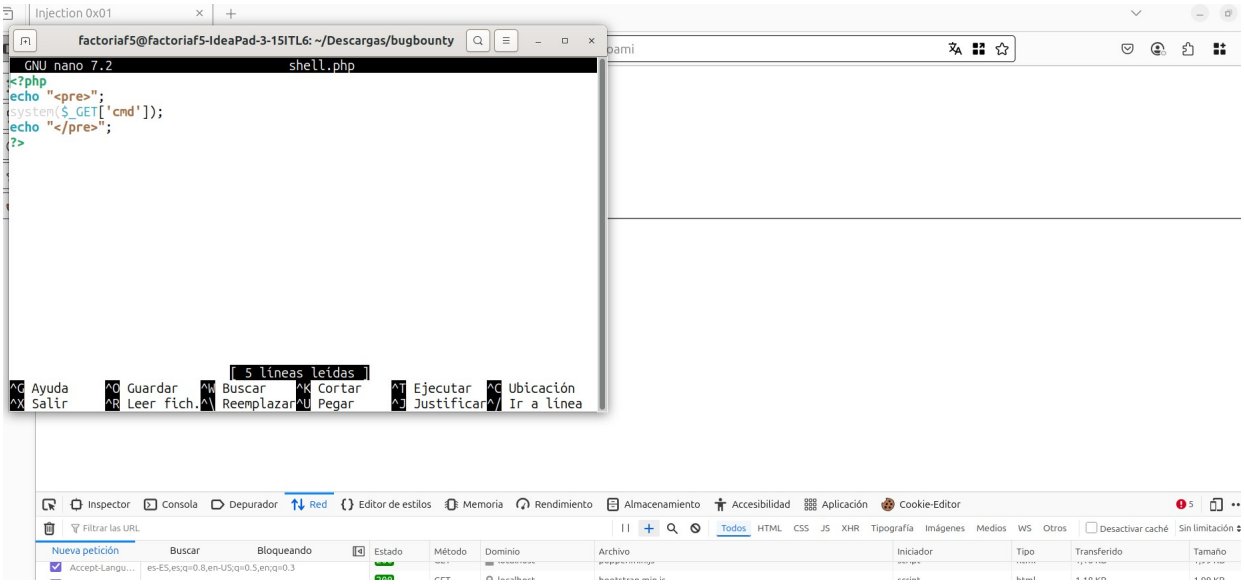
Tabla de contenidos:

Objetivo:

Demostrar cómo un atacante puede evadir restricciones de subida de archivos para ejecutar código malicioso en el servidor, incluso cuando la aplicación aparenta válidar tipos y extensiones

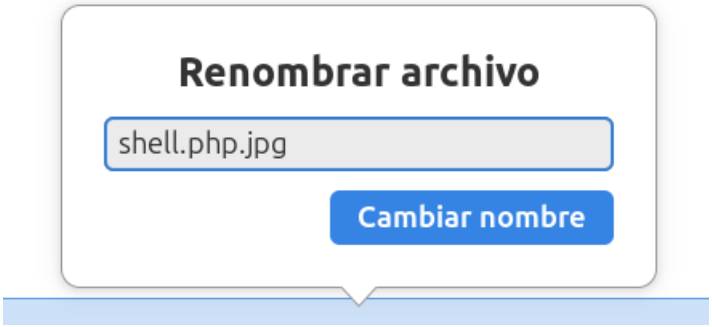
Paso a paso:

1. Crear el archivo malicioso

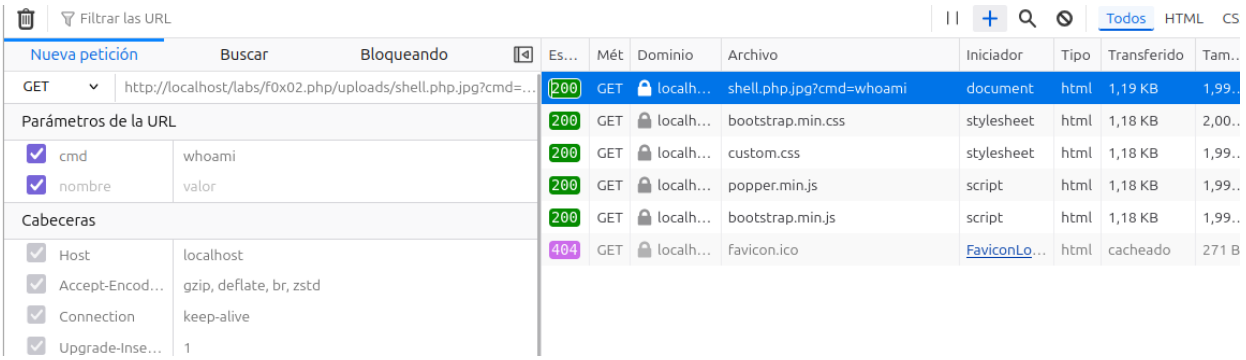


Creamos un archivo shell.php con código malicioso en PHP

2. Cambiar extensión a .jpg



3. Subir el archivo desde el formulario

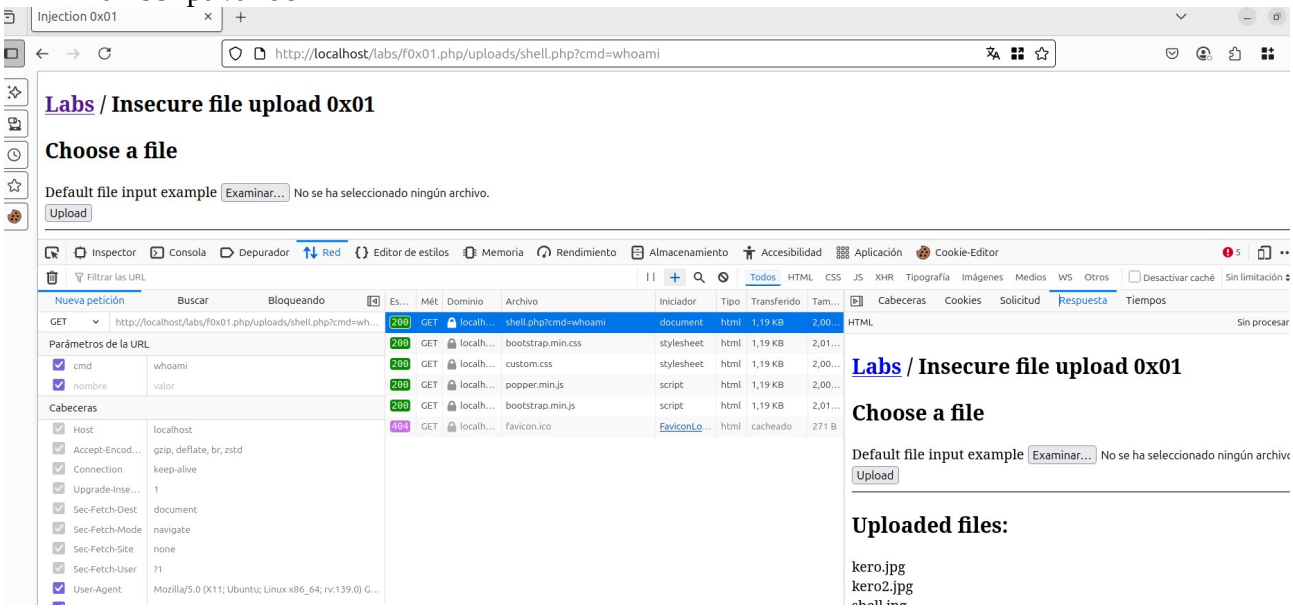


4. Ejecutar el payload malicioso

   Accedemos al archivo por URL con un parámetro cmd para ejecutar comandos:

http://localhost/labs/f0x02.php/uploads/shell.php.jpg?cmd=whoami

   El servidor ejecuta el código PHP aunque la extensión termine en .jpg ya que lo trata como un script válido



Conclusión:

Este laboratorio demuestra que incluso con validaciones de tipo archivo por extensión, es posible:

- Insertar código malicioso PHP
- Subirlo disfrazado como imagen (.php.jpg)
- Ejecutarlo si el servidor intercepta el archivo como PHP

Esto evidencia la importancia de:no

- Validar contenido del archivo(no solo su extensión)
- Nunca permitir que archivos subidos se ejecuten directamente
- Almacenar archivos en rutas sin permisos de ejecución

Recomendaciones para la mitigación:

Para prevenir este tipo de vulnerabilidades, Debe aplicar **múltiples capas de defensa:**

1. **Validación del lado servidor**

- Verificar que la extensión y el contenido (por magic numbers o MIME type real) corresponde a archivos válidos(ej: kero.jpg)
- No confiar solo en vaidaciones del lado del cliente o extensiones

2. **Deshabilitar la ejecución en el directorio de subida**

- Configurar el servidor para que no ejecute archivos(PHP, ASP…) en la carpeta donde se almacenan los archivos subidos
  RemoveHandler .php
  RemoveType .php
  php_flag engine off

3. **Renombrar los archivos al subirlos**
- No mantener el nombre original del archivo subido. Usar nombres aleatorios o UUIDs
4. **Almacenarlos fuera de public root**
- Si es posible, guarda los archivos subidos, fuera del directorio publico (/var/www/html) y accede a ellos mediante scripts controlados
5. **Revisar extensiones peligrosas**
- Bloquear las extensiones peligrosas como .php .phtml .asp .jsp...
6. **Aplicar Content Security Policy(CSP)**
- Limita la ejecución de scripts y carga de recursos externos para reducir el riesgo de ejecución si un archivo malicioso se accede desde el navegador
7. **Auditorias periodicas**
- Realizar pruebas de seguridad en la funcionalidad de subida de archivos y analiza logs de accesos para detectar comportamientos anómalos