Log Me In Again

In this challenge, I was supposed to find a flag in a database server. To do this, I followed the following steps:

1. First, I leaked database names through information_schema.

I found that the two databases in the database server were 'information_schema' and 'logmein'. To leak the database names, I programmed a python script to send the following SQL query to the website (please refer to logMeInAgain.py for more details):

```
string = "admin' UNION SELECT IF(SUBSTR((SELECT SCHEMA_NAME FROM
information_schema.SCHEMATA LIMIT 1 OFFSET x), {}, 1) = '{}' , SLEEP(5), 0), 2,
3; -- ".format(i, characters[index])
```

This code leaks the ith letter of the database's name. The offset is the xth database name that the programmer is trying to leak (starting at offset 0).

This part of the code is used for all steps of the exploit, so I will explain it here:

a. First, I make a list (please call this list 'characters') of all possible characters that could be in the string I am trying to leak (in this case, that is the database's name). Then, I inspect the first character of the string that I am trying to leak (which would be what SUBSTR() returns) and iterate through my list 'characters' (please assume that the character I am currently at is characters[index]). If the first letter of the string I am trying to leak matches characters[index], the script sleeps for 5 seconds. If the script sleeps, I have found a letter of the string I am trying to leak and can move on to the next letter (by adding 1 to i) and try to leak it. If the script does not sleep, I keep on iterating through 'characters' (by adding 1 to index) to find a character that matches the current letter of the string that I am trying to leak.

2. Next, I leaked table names through information_schema.

I found that the two tables in 'logmein' were 'secrets' and 'users'. To leak the table names, I programmed my script to send the following SQL query to the website:

```
string = "admin' UNION SELECT IF(SUBSTR((SELECT TABLE_NAME FROM
information_schema.TABLES WHERE TABLE_SCHEMA = 'logmein' AND TABLE_TYPE = 'BASE
TABLE' LIMIT 1 OFFSET x), {}, 1) = '{}' , SLEEP(5), 0), 2, 3; -- ".format(i,
characters[index])
```

This code leaks the ith letter of the table's name. The offset is the xth table that the programmer is trying to leak (starting at offset 0). The logic for why this code works was explained in section (a.).

3. After that, I leaked column names of the table 'secrets' through information_schema

I found that the two columns in 'secrets' were 'ID' and 'value'. To leak the column names, I programmed my script to send the following SQL query to the website:

```
string = "admin' UNION SELECT IF(SUBSTR((SELECT COLUMN_NAME FROM
information_schema.COLUMNS WHERE TABLE_SCHEMA = 'logmein' AND TABLE_NAME =
'secrets' LIMIT 1 OFFSET x), {}, 1) = '{}' , SLEEP(5), 0), 2, 3; -- ".format(i,
characters[index])
```

This code leaks the ith letter of the column's name. The offset is the xth column that the programmer is trying to leak (starting at offset 0). The logic for why this code works was explained in section (a.).

4. Finally, I leaked the flag by getting the string in the 'value' column of the table 'secrets'

To leak the flag, I programmed my script to send the following SQL query to the website:

```
string = "admin' UNION SELECT IF(SUBSTR((SELECT VALUE FROM secrets LIMIT 1 OFFSET
0), {}, 1) = BINARY '{}' , SLEEP(5), 0), 2, 3; -- ".format(i, characters[index])
```

This code leaks the ith letter of the flag's name. The logic for why this code works was explained in section (a.).