CMSsite

<https://github.com/VictorAlagwu/CMSsite>

Install:

Do the following change the code in the **db.php**

* + $DB\_host = "";
  + $DB\_user = "";
  + $DB\_pass = "";
  + $DB\_name = "";
* Import the database file named "php\_cms.sql"

**SQL Injection**

<https://www.exploit-db.com/exploits/48485>

Retrieve Version e Database name

<https://www.exploit-db.com/exploits/48490>

<https://www.exploit-db.com/exploits/48484>

<https://www.exploit-db.com/exploits/48451>

SQL-Injection

WordPress plugin Chained-Quiz SQL-Injection

<https://www.exploit-db.com/exploits/45221>

Vulnerability description

The Chained-Quiz WordPress plugin lets the user create quiz where the next question depends on the answer of the previous one. This plugin, in the versions prior to 1.0.8 is vulnerable to time-based SQL injection.

Causes of vulnerability

This version of the plugin allows unauthorized users to execute SQL queries via the *answer* parameter. The problem lies on the *answer* backend variable.

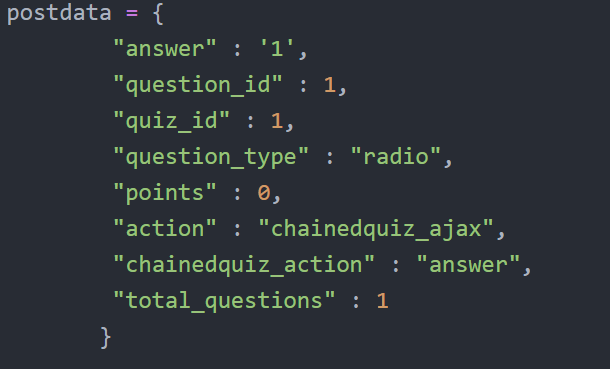
Implementation

The provided version of the plugin is v0.8.7. In order for the exploit to work, the database on which WordPress runs must be mySQL.

When we submit the answer of a question via a POST request, we can inject SQL commands to the parameter *answer* to check which database the web-application is using.

We are going to inject the command SLEEP(15), that is a mySQL command. This means that if the application by reading the request will sleep 15 seconds, the underlying database will be a mySQL database.

The parameters that are being passed in the POST request are the following:



The payload will be inserted in the answer parameter and it will be the following:



The POST request is executed with the python library *requests*:

Cross-Site Scripting

<https://www.exploit-db.com/exploits/37521>

CodeIgniter XSS scripting attack

Vulnerability description

CodeIgniter is a PHP framework to create web applications. The version prior to v2.1.2 are vulnerable to XSS scripting attacks.

Causes of vulnerability

The cause of the vulnerability lies on the function xss\_clean(), that is a filter protection used to sanitize user inputs. The filter is not working correctly, therefore it is possible to execute a malicious script.

Implementation