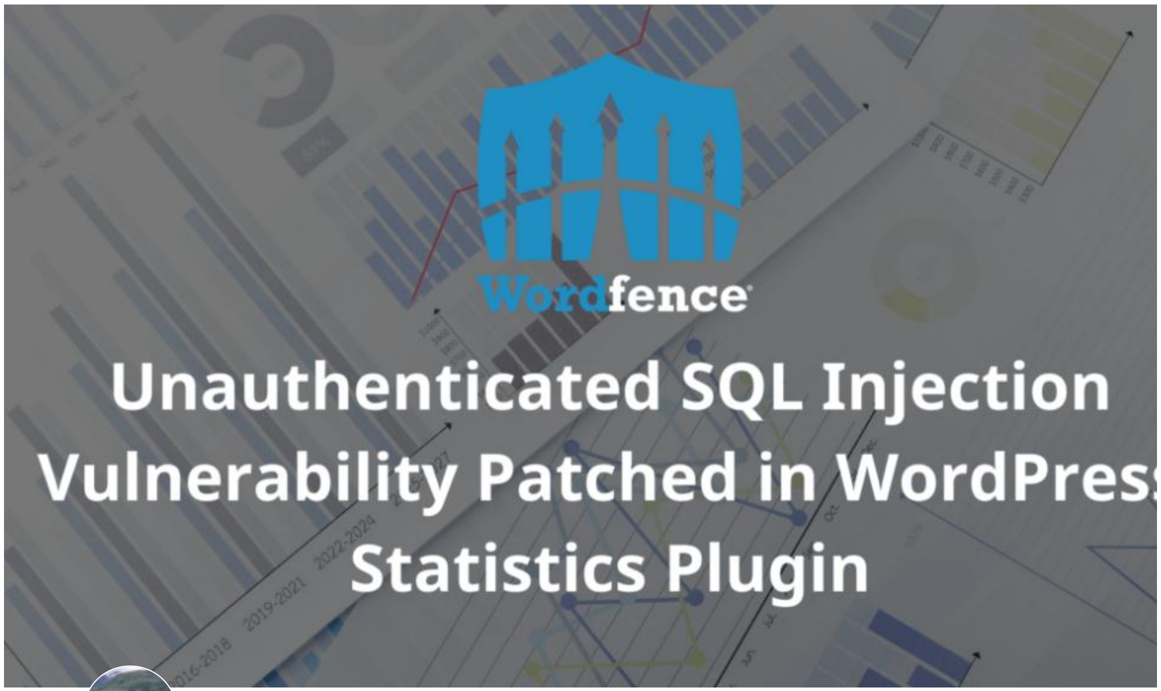
Wordfence
Unauthenticated SQL Injection
Vulnerability Patched in WordPres
Statistics Plugin

Chloe Chamberland

February 10, 2022

Unauthenticated SQL Injection Vulnerability Patched in WordPress Statistics Plugin

On February 7, 2022, Security Researcher Cyku Hong from DEVCORE reported a vulnerability to us that they discover
in WP Statistics, a WordPress plugin installed on over 600,000 sites. This vulnerability made it possible for
unauthenticated attackers to execute arbitrary SQL queries by appending them to an existing SQL query. This could b
used to extract sensitive information like password hashes and secret keys from the database. On request, we assig
them the vulnerability identifier: CVE-2022-0513.

All Wordfence users, including Free, Premium, Care, and Response, are protected from exploits targeting this vulnerability thanks to the Wordfence Firewall's built-in SQL Injection protection.

Even though Wordfence provides protection against this vulnerability, we strongly recommend ensuring that your site
has been updated to the latest patched version of "WP Statistics," which is version 13.1.5 at the time of this publicati

Description: Unauthenticated Blind SQL Injection
Affected Plugin: WP Statistics
Plugin Slug: wp-statistics
Plugin Developer: VeronaLabs
Affected Versions: <=13.1.4
CVE ID: CVE-2022-0513
CVSS Score: 9.8 (Critical)
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Researcher/s: Cyku Hong from DEVCORE
Fully Patched Version: 13.1.5

WP Statistics is a WordPress plugin designed to provide a centralized hub for all of a WordPress site's statistics, suc
visitor data, and it emphasizes storing this data locally to the WordPress site to preserve user privacy. As such, it is

database through the use of SQL queries. Unfortunately, the implementation of one of these queries was insecure, creating a SQL Injection vulnerability.

When the "Record Exclusions" feature was enabled, this vulnerability became exploitable. The "Record Exclusions" feature is designed to record when a visit, or a "hit", is excluded from the site's statistics, such as visits by users with specific roles, login page access, and anything else that a site owner may have explicitly selected to exclude. It recor that data to a separate database table so as not to contaminate the main statistical data the plugin collects.

In order to record these hits when a caching plugin was enabled, the plugin registered a REST route `/wp-json/wp-statistics/v2/hit` that would call the `hit_callback()` function. This function would then call the `record()` functi from the 'Hits' class which checks to see if the request should be excluded and determines what exclusion the reque correlates to, prior to calling the next appropriate `record()` function.

```
public static function record()
{

    # Check Exclusion This Hits
    $exclusion = Exclusion::check();

    # Record Hits Exclusion
    if ($exclusion['exclusion_match'] === true) {
        Exclusion::record($exclusion);
    }

    # Record User Visits
    if (Visit::active() and $exclusion['exclusion_match'] === false) {
        Visit::record();
    }

    # Record Visitor Detail
    if (Visitor::active()) {
        $visitor_id = Visitor::record($exclusion);
    }
```

When the `exclusion_match` parameter is set to true in a request, the data is then passed to the `record()` function fr the 'Exclusion' class where the plugin attempts to update the count of an exclusion reason for the day if it is present i the database. If the exclusion reason isn't present in the database for the current date the initial query will return false and trigger the next query to add a new record count to the table for the reason.

```
public static function record($exclusion = array())
{
    global $wpdb;

    // If we're not storing exclusions, just return.
    if (self::record_active() != true) {
        return;
    }

    // Check Exist this Exclusion in this day
    $result = $wpdb->query("UPDATE " . DB::table('exclusions') . " SET `count` = `count` + 1 WHERE `date` = '" .
    if (!$result) {
        $insert = $wpdb->insert(
            DB::table('exclusions'),
            array(
                'date'   => TimeZone::getCurrentDate('Y-m-d'),
                'reason' => $exclusion['exclusion_reason'],
                'count'  => 1,
            )
        );
        if (!$insert) {
            if (!empty($wpdb->last_error)) {
                \WP_Statistics::log($wpdb->last_error);
            }
        }
```

The `$wpdb->query()` function was used for the initial UPDATE query and used the user-supplied '`exclusion_reason` value as part of the query. Due to the fact that there was no escaping on the user supplied value, or parameterization the query, attackers could easily append additional SQL queries to the existing query via the '`exclusion_reason`' and extract sensitive information from the database.

Since no data from the SQL query was returned in the response, and the response did not indicate a boolean answer, attacker would need to use a Time-Based blind approach to extract information from the database. This means that they would need to use SQL CASE statements along with the `SLEEP()` command while observing the response time each request to steal information from the database. This is an intricate, yet frequently successful method to obtain

the `exclusion_reason` parameter set to the SQLi payload, and the `wp_statistics_hit_rest` parameter set to true, along with passing the string `wp-json/` in the request URI to trigger the same `record()` function from the 'Exclusion class. This method did not require a caching plugin to be enabled to obtain a valid nonce to trigger the REST endpoin This is due to the `is_rest_request()` function returning true when the `$_SERVER['REQUEST_URI']` contains the RES prefix, `wp-json/`, even if the request isn't a genuine REST request. This ultimately triggers the entire record process.

Conclusion

In today's post, we detailed a flaw in the "WP Statistics" plugin that made it possible for unauthenticated attackers to inject arbitrary SQL queries to steal sensitive information from a database. This flaw has been fully patched in versio 13.1.5.

We recommend that WordPress site owners immediately verify that their site has been updated to the latest patchec version available, which is version 13.1.5 at the time of this publication.

All Wordfence users, including Free, Premium, Care, and Response, are protected from exploits targeting this vulnerability thanks to the Wordfence Firewall's built-in SQL Injection protection.

If you believe your site has been compromised as a result of this vulnerability or any other vulnerability, we offer Incid Response services via Wordfence Care. If you need your site cleaned immediately, Wordfence Response offers the same service with 24/7/365 availability and a 1-hour response time. Both these products include hands-on support i case you need further assistance.

If you know a friend or colleague who is using this plugin on their site, we highly recommend forwarding this advisory them to help keep their sites protected as this is a serious vulnerability that can lead to complete site takeover.

Congratulations to Cyku Hong from DEVCORE for discovering and responsibly disclosing this vulnerability to the plugin developers. As a reminder, Wordfence is a CVE Numbering Authority (CNA) and we can assign CVE IDs to your vulnerak discoveries in WordPress Plugins, Themes, and Core. If you need a CVE for one of your WordPress finds, please fill out form here. Your vulnerability discovery may be featured on our blog with your permission!

Did you enjoy this post? Share it!

Comments

No Comments