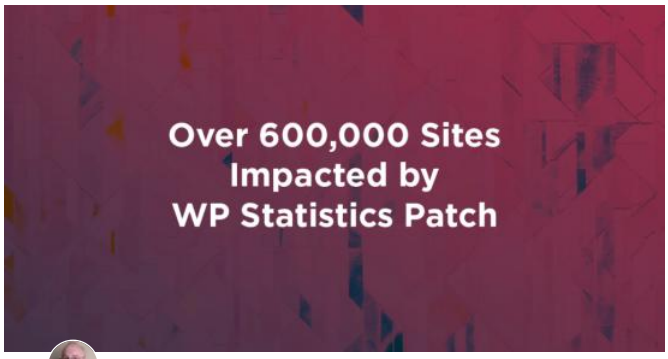

Ram Gall

May 18, 2021

Over 600,000 Sites Impacted by WP Statistics Patch

On March 13, 2021, the Wordfence Threat Intelligence team initiated responsible disclosure for a vulnerability in WP Statistics, a plugin installed on over 600,000 WordPress sites.

The vulnerability allowed any site visitor to extract sensitive information from a site's database via Time-Based Blind SQL Injection.

We received a response to our initial disclosure the same day, on March 13, 2021, and sent the full disclosure to the plugin's developers at VeronaLabs. A patch for this vulnerability was released on March 25, 2021.

While our original report indicated that an attacker needed to be authenticated to exploit this vulnerability, we have since discovered that it can be exploited by unauthenticated attackers as well.

Fortunately, all sites running Wordfence, including those using Wordfence Premium as well as the free version, are protected against this vulnerability by the Wordfence firewall's built-in SQL injection protection. This built-in protection blocks most SQL injection attempts even if a vulnerability is not yet known.

Description: Unauthenticated Time-Based Blind SQL Injection
Affected Plugin: WP Statistics
Plugin Slug: wp-statistics
Affected Versions: < 13.0.8
CVE ID: CVE-2021-24340
CVSS Score: 7.5 (High)
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Fully Patched Version: 13.0.8

WP Statistics is a WordPress plugin that allows site owners to see detailed statistics about visitors to their site, including which pages on the site they visit. As an administrator, accessing the WP Statistics "Pages" menu item generates a SQL query in order to display statistics on which pages have received the most traffic.

While the "Pages" page was intended for administrators only and would not display information to non-admin users, it was possible to start loading this page's constructor by sending a request to `wp-admin/admin.php` with the `page` parameter set to `wps_pages_page`. Since the SQL query ran in the constructor for the "Pages" page, this meant that any site visitor, even those without a login, could cause this SQL query to run. A malicious actor could then supply malicious values for the `ID` or `type` parameters.

```
class pages_page
{

    public function __construct()
    {
        global $wpdb;

        if (Menus::in_page('pages')) {

            // Disable Screen Option
            add_filter('screen_options_show_screen', '__return_false');

            // Check Exist Statistics For Custom Page
            if (self::is_custom_page()) {
                $page_count = $wpdb->get_var("SELECT COUNT(*) FROM " . DB::table('pages') . " WHERE `id` = " . esc_sql
                if ($page_count < 1) {
                    wp_die(__('Your request is not valid.', 'wp-statistics'));
                }
            }

            // Is Validate Date Request
            $DateRequest = Admin_Template::isValidDateRequest();
            if (!$DateRequest['status']) {
                wp_die($DateRequest['message']);
            }
        }
    }

    public static function is_custom_page()
    {
        return (isset($_GET['ID']) and isset($_GET['type']));
    }
```

Unfortunately, while this SQL query used `esc_sql` to attempt to escape the `ID` and `type` input parameters, it did not use a prepared statement. Since the `ID` input parameter was not quoted, it was trivial to bypass the `esc_sql` function and generate queries which could be used to extract sensitive information from the site.

As this was a Time-Based Blind SQL injection vulnerability, exfiltrating information would be a relatively slow process, and it would be impractical to use it to extract bulk records, but high-value information such as user emails, password hashes, and encryption keys and salts could be extracted in a matter of hours with the help of automated tools such as sqlmap. In a targeted attack, this vulnerability could be used to extract personally identifiable information from commerce sites containing customer information. This underscores the importance of having security protections with an endpoint firewall in place wherever sensitive data is stored.

We recently reported on another SQL injection vulnerability in which we were able to bypass a number of protections, and this is yet another example of why simply escaping input is insufficient to prevent SQL injection. The only reliable method of preventing SQL injection is to prepare all SQL statements before executing them, which can be performed using `$wpdb->prepare()`. While it might still be possible to construct a vulnerable query that uses a prepared statement, it is very difficult to do so unintentionally.

Timeline

March 13, 2021 – The Wordfence Threat Intelligence team finishes researching a vulnerability in the WP Statistics plugin and contacts VeronaLabs. VeronaLabs responds and we provide full disclosure.
March 15, 2021 – VeronaLabs replies with a fixed version for us to test and we verify that it corrects the issue.
March 25, 2021 – A patched version of the plugin, 13.0.8, is released.

Conclusion

used to extract sensitive data from any website with the plugin installed. We also covered why sanitizing and escaping input is not sufficient to protect against SQL Injection attacks.

All Wordfence users, including Wordfence Premium customers as well as those still using the free version, are protected against this vulnerability by our firewall's built-in SQL Injection protection.

If you have this plugin installed on your site, we urge you to update to the patched version, 13.0.8, as soon as possible. If you have a friend or colleague who is using this plugin on their site, we recommend forwarding this advisory to them to help keep their sites protected as this vulnerability allows attackers to access confidential data stored in a site's database.

Comments

No Comments