Ram Gall

April 23, 2020

Critical Vulnerabilities Patched in MapPress Maps Plugin

On April 1, 2020, the Wordfence Threat Intelligence Team discovered two vulnerabilities in MapPress Maps for WordPress, a WordPress plugin with over 80,000 installations. One vulnerability that allowed stored Cross-Site Scripting (XSS) was present in both the free and pro versions of the plugin, while a far more critical vulnerability that allowed Remote Code Execution (RCE) was present in the pro version.

We reached out to the plugin's author the next day, April 2, 2020 and received a response within a few hours. A patched version of both MapPress Free and MapPress Pro were released within hours. We strongly recommend updating both the free and pro versions to the latest version, 2.54.2, as soon as possible.

Wordfence Premium users received a new firewall rule on April 2, 2020 to protect against exploits targeting these vulnerabilities. Wordfence users still using the free version will receive the rule after thirty days on May 2, 2020.

Description: Authenticated Map Creation/Deletion Leading to Stored Cross-Site Scripting (XSS)
Affected Plugin: MapPress Maps for WordPress
Plugin Slug: mappress-google-maps-for-wordpress
Affected Versions: <=2.53.8 Free and Pro
CVE ID: CVE-2020-12077
CVSS Score: 6.5(Medium)
CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:L
Fully Patched Version: 2.53.9

MapPress Maps for WordPress allows site owners to add custom maps on their site, using both the Google Maps API and the open-source Leaflet engine. Both the free and pro versions of this plugin registered AJAX actions that called functions lacking capability checks and nonce checks. The affected AJAX hooks:

```
54 | add_action('wp_ajax_mapp_delete', array(__CLASS__, 'ajax_delete'));
```

```
57 | add_action('wp_ajax_mapp_save', array(__CLASS__, 'ajax_save'));
```

As such, it was possible for a logged-in attacker with minimal permissions, such as a subscriber, to add a map containing malicious JavaScript to an arbitrary post or page by sending a `$_POST` request to `wp-admin/admin-ajax.php` with the `action` parameter set to `mapp_save`, the `postid` parameter set to the post to add the map to, and the `map` parameter containing JSON data representing the map to be added. Malicious JavaScript could be added to the `title` and `body` parameters of a "Point of Interest" in the saved map, which would be executed whenever a visitor to the site clicked on the mapping pin denoting that Point of Interest (POI). Alternatively, if the global setting for "Show a list of POIs with each map" was enabled, the JavaScript would be executed immediately upon visiting a page with an impacted map.

This vulnerability could redirect a site visitor to a malicious site, or even use an administrator's session to take over the site by adding a malicious administrative user.

The vulnerable function:

```
232 | static function ajax_save() {
233 |     ob_start();
234 |
235 |     $mapdata = (isset($_POST['map'])) ? json_decode(stripslashes($_POST['map']), true) : null;
236 |     $postid = (isset($_POST['postid'])) ? $_POST['postid'] : null;
237 |
238 |     if (!$mapdata)
239 |         Mappress::ajax_response('Internal error, your data has not been saved!');
240 |
241 |     $map = new Mappress_Map($mapdata);
242 |     $mapid = $map->save($postid);
243 |
244 |     if ($mapid === false)
245 |         Mappress::ajax_response('Internal error, your data has not been saved!');
246 |
247 |     do_action('mappress_map_save', $mapid);     // Use for your own developments
248 |
249 |     // Return saved mapid
250 |     Mappress::ajax_response('OK', array('mapid' => $mapid));
251 | }
```

Alternatively, an attacker could delete an existing map by sending a `$_POST` request to `wp-admin/admin-ajax.php` with the `action` parameter set to `mapp_delete` and the `mapid` parameter of the map to delete. While this would be significantly less impactful, it could still be used by an attacker to remove any of the original maps on the site, potentially prompting an administrator to investigate and trigger a script inserted into a malicious map.

The vulnerable function:

```
276 | static function ajax_delete() {
277 |     ob_start();
278 |
279 |     $mapid = (isset($_POST['mapid'])) ? $_POST['mapid'] : null;
280 |     $result = Mappress_Map::delete($mapid);
281 |
282 |     if (!$result)
283 |         Mappress::ajax_response("Internal error when deleting map ID '$mapid'!");
284 |
285 |     do_action('mappress_map_delete', $mapid);    // Use for your own developments
286 |     Mappress::ajax_response('OK', array('mapid' => $mapid));
287 | }
```

Description: Authenticated File Upload, Deletion, and Disclosure Leading to RCE or Site Reset
Affected Pluain: MapPress Maps for WordPress

CVSS Score: 9.9(Critical)
CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H
Fully Patched Version: 2.53.9

The pro version of the MapPress plugin offers the ability to create templates controlling how maps are displayed. It saves these templates as custom .php files. Unfortunately, the plugin registered several AJAX actions that called functions without capability checks or nonce checks:

```
add_action('wp_ajax_mapp_tpl_get', array(__CLASS__, 'ajax_get'));
add_action('wp_ajax_mapp_tpl_save', array(__CLASS__, 'ajax_save'));
add_action('wp_ajax_mapp_tpl_delete', array(__CLASS__, 'ajax_delete'));
```

Note that while the names of some of these functions are identical to functions in the previous vulnerability, they are located in another file and belong to a different class. The file containing this vulnerable code was present in both the free and pro versions of the MapPress plugin, but was only active in the pro version.

As such, it was possible for an authenticated attacker with minimal permissions to upload an executable PHP file such as a backdoor or webshell. This could easily lead to complete site takeover, as an attacker with backdoor access could then modify any file on the site, upload additional files, or connect to the database and insert an administrative user.

An attacker could exploit this vulnerability by sending a `$_POST` request to `wp-admin/admin-ajax.php` with the `action` parameter set to `mapp_tpl_save`, the `name` parameter set to the base name of the file they wanted to create, and the `content` parameter set to executable PHP code. This file would then be created and could be executed from the directory of the currently active theme, resulting in Remote Code Execution.

The vulnerable function:

```
static function ajax_save() {
    $name = (isset($_POST['name'])) ? $_POST['name'] : null;
    $content = (isset($_POST['content'])) ? stripslashes($_POST['content']) : null;
    $filepath = get_stylesheet_directory() . '/' . $name . '.php';

    $result = @file_put_contents($filepath, $content);
    if ($result === false)
        Mappress::ajax_response('Unable to save');

    // Return filepath after save
    Mappress::ajax_response('OK', $filepath);
}
```

An authenticated attacker could also delete any existing PHP file on the site by sending a `$_POST` request to `wp-admin/admin-ajax.php` with the `action` parameter set to `mapp_tpl_delete`, and the `name` parameter set to the basename of the file to delete.

For example, to delete `wp-config.php` a directory traversal attack could be used, with the `name` parameter set to `../../../wp-config`. This would cause the site to be reset, at which point an attacker could gain control of the site by setting it up from scratch and connecting it to a remotely hosted malicious database.

The vulnerable function:

```
static function ajax_delete() {
    $name = (isset($_POST['name'])) ? $_POST['name'] : null;
    $filepath = get_stylesheet_directory() . '/' . $name . '.php';

    $result = @unlink($filepath);
    if ($result === false)
        Mappress::ajax_response('Unable to delete');

    Mappress::ajax_response('OK');
}
```

Finally, an authenticated attacker could view the contents of any existing PHP file on the site by sending a `$_GET` request to `wp-admin/admin-ajax.php` with the `action` parameter set to `mapp_tpl_get`, and the `name` parameter of the file to disclose. For example, to view the contents of `wp-config.php`, an attacker could set the `name` parameter to `../../../../wp-config`. This could allow an attacker to determine the site's database credentials. If the site's database allowed remote access, the attacker could then use this to add an administrative user or make any other desired changes to the database, up to and including deleting nearly all of the site's content.

The vulnerable function:

```
static function ajax_get() {
    $name = (isset($_GET['name'])) ? $_GET['name'] : null;

    $filename = $name . '.php';
    $filepath = get_stylesheet_directory() . '/' . $filename;

    $html = @file_get_contents($filepath);
    $standard = @file_get_contents(Mappress::$basedir . "/templates/$filename");

    if (!$standard)
        Mappress::ajax_response('Invalid template');

    $template = new Mappress_Template(array(
        'name' => $name,
        'content' => ($html) ? $html : $standard,
        'path' => $filepath,
        'standard' => $standard,
        'exists' => ($html) ? true : false
    ));

    Mappress::ajax_response('OK', $template);
}
```

Disclosure Timeline

April 1, 2020 – Wordfence Threat Intelligence discovers and analyzes vulnerabilities.
April 2, 2020 – Firewall rule released for Wordfence Premium users. Initial contact with plugin developer. Developer releases patch before the end of day.
May 2, 2020 – Firewall rule becomes available to Wordfence free users.

Conclusion

In today's post, we detailed two vulnerabilities in the MapPress Maps for WordPress plugin, including a stored Cross-Site Scripting (XSS) and a more critical Remote Code Execution (RCE), File Disclosure, and File deletion vulnerability. These vulnerabilities have been fully patched in version 2.53.9 and we strongly recommend that all users of this plugin upgrade to the latest version available immediately. Sites running Wordfence Premium have been protected against these vulnerabilities since April 2, 2020. Sites still running the free version of Wordfence will receive the firewall rule update on May 1, 2020.

Special thanks to Chris Richardson, the developer of the MapPress Maps for WordPress plugin, for his extremely rapid and professional handling of our disclosure. Patching a vulnerability on the same day the vulnerability disclosure is received by the developer is an exemplary response.

Breaking WordPress Security Research in your inbox as it happens.

you@example.com

By checking this box I agree to the terms of service and privacy policy.*

SIGN UP

Our business hours are 9am-8pm ET, 6am-5pm PT and 2pm-1am UTC/GMT excluding weekends and holidays.
Response customers receive 24-hour support, 365 days a year, with a 1-hour response time.

Terms of Service Privacy Policy

CCPA Privacy Notice

Products

Wordfence Free
Wordfence Premium
Wordfence Care
Wordfence Response
Wordfence Central

Support

Documentation
Learning Center
Free Support
Premium Support

News

Blog
In The News
Vulnerability Advisories

About

About Wordfence
Careers
Contact
Security
CVE Request Form

Stay Updated

Sign up for news and updates from our panel of experienced security professionals.

you@example.com

By checking this box I agree to the terms of service and privacy policy.*

SIGN UP