Wordfence
Stored Cross-Site Scripting Vulnerability Patched in a WordPress Photo Gallery Plugin

Chloe Chamberland February 24, 2022

# Stored Cross-Site Scripting Vulnerability Patched in a WordPress Photo Gallery Plugin

On November 11, 2021 the Wordfence Threat Intelligence team initiated the responsible disclosure process for a vulnerability we discovered in "Photoswipe Masonry Gallery", a WordPress plugin that is installed on over 10,000 sites This flaw makes it possible for an authenticated attacker to inject malicious JavaScript that executes whenever a site administrator accesses the PhotoSwipe Options page or a user accesses a page with a gallery created by the plugin.

All Wordfence users, including users of our Free, Premium, Care, and Response products are protected from exploits targeting this vulnerability thanks to the Wordfence Firewall's built-in Cross-Site Scripting (XSS) protection.

We attempted to reach out to the developer day on November 11, 2021, the same day we discovered the vulnerability We never received a response after a couple of follow-ups so we sent the full details to the WordPres.org plugins tea on November 20, 2021. The plugin was fully patched on January 14, 2022.

We strongly recommend ensuring that your site has been updated to the latest patched version of "Photoswipe Mas Gallery", which is version 1.2.18 at the time of this publication.

**Description:** Authenticated Stored Cross-Site Scripting
**Affected Plugin:** Photoswipe Masonry Gallery
**Plugin Slug:** photoswipe-masonry
**Plugin Developer:** Web Design Gold Coast
**Affected Versions:** <= 1.2.14
**CVE ID:** CVE-2022-0750
**CVSS Score:** 6.4 (Medium)
**CVSS Vector:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N
**Researcher/s:** Chloe Chamberland
**Fully Patched Version:** 1.2.15

Photoswipe Masonry Gallery is a plugin designed to enhance gallery creation using the default WordPress gallery bui
which can be added to WordPress pages and posts. As with many other plugins available in the WordPress repositor

owner chooses to create and includes things like thumbnail width and height for images along with many other setti
Unfortunately, this plugin had a vulnerability that made it possible for attackers to modify these settings.

Diving deeper, the plugin registered an `admin_menu` action that hooked to the update function that controlled saving t
plugins settings.

```
add_action('admin_menu', array('photoswipe_plugin_options', 'update'));
```

As with several other admin style hooks in WordPress, like `wp_ajax`, `admin_post`, and `admin_init`, the `admin_menu` h
checks to see if a user is accessing the administrative area of a site prior to loading the hooked function. It does not, however, validate that the user accessing the administrative area is an admin user. The `admin_menu` action does this
add additional menu pages to the administrative area of a WordPress site. This means that an authenticated user accessing the /wp-admin area of a vulnerable site will trigger the hook and ultimately execute the function associate
with the hook. In this case that is the update function.

Due to the fact that the update function had no capability checks or nonce checks of it's own, any authenticated user
accessing the /wp-admin area of a vulnerable site could send a POST request with `photoswipe_save` set to true and
update the settings of the plugin.

```
public static function update() {

    if(isset($_POST['photoswipe_save'])) {

        $options = photoswipe_plugin_options::pSwipe_getOptions();

        $options['thumbnail_width'] = stripslashes($_POST['thumbnail_width']);
        $options['thumbnail_height'] = stripslashes($_POST['thumbnail_height']);

        $options['max_image_width'] = stripslashes($_POST['max_image_width']);
        $options['max_image_height'] = stripslashes($_POST['max_image_height']);
```

Considering the `thumbnail_width`, `thumbnail_height`, `max_image_width`, and `max_image_height` parameters had r
sanitization or validation, attackers could inject malicious JavaScript into the plugin's settings which would result in t
malicious JavaScript executing anytime an administrator accessed the plugins setting page or a user accessed a ga
that was created with the plugin. This malicious JavaScript could be used to redirect site visitors accessing a gallery
malicious domains for further infection or inject new administrative user accounts if an administrator accessed a pa
containing the malicious payload. As such, it is important to verify that your site has been updated to the latest versic
as soon as possible.

## Timeline

**November 11, 2021** – Conclusion of the plugin analysis that led to the discovery of a Stored Cross-Site Scripting Vulnerability in the "Photoswipe Masonry Gallery" plugin. We verify that the Wordfence Firewall provides sufficient protection. We attempt to initiate contact with the developer.
**November 30, 2021** – After no response from the developer, we send the full disclosure details to the WordPress plu
team. They acknowledge the report and make contact with the developer.
**January 4, 2022** – We follow-up with the plugins team asking about a missing capability check on the vulnerable function. They inform us that they have let the developer know and they will have a release out by the end of the mon
**January 14, 2022** – A fully patched version of the plugin is released as version 1.2.15.

## Conclusion

In today's post, we detailed a flaw in the "Photoswipe Masonry Gallery" plugin that made it possible for authenticated attackers to inject malicious web scripts that will execute whenever a site owner accesses the PhotoSwipe Options page or a gallery created with the plugin, which could lead to complete site compromise. This flaw has been fully patched in version 1.2.15.

We recommend that WordPress site owners immediately verify that their site has been updated to the latest patchec

targeting this vulnerability thanks to the Wordfence Firewall's built-in Cross-Site Scripting (XSS) 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 ii case you need further assistance.

Did you enjoy this post? Share it!

## Comments

2 Comments

Kadigan *
February 24, 2022
12:08 pm

I would think that it should be easily-visible in all sorts of admin-area functions and actions that they don't check for permissions/capabilities at all. If it comes at a cost of having "__no_perm_check" tacked onto each and every one, I can live with that

But this situation really brings attention to what I like to call Rule #1 of dealing with user-provided input:

-- Assume all user input is malicious until proven otherwise. Act accordingly. If sanitising isn't straightforward, fail and do it early. --

The above is especially true for things like admin panel actions and similar, since - if everything is above-board - the values should be exactly what you expect them to be. Failure at this stage shouldn't really happen for legit admin actions, but should happen immediately for potentially malicious ones.

This particular case is straightforward in the extreme as well, because it's quite obvious that all four inputs are intended to be numeric. That is very easily checked against - and could stand aditional "idiot checks", like testing against values below 1 and so on.

PHP, due to its weak typing, makes it very easy to accidentally allow more freedom than you'd want - something that clearly happene here. Clamping that freedom down with checks (to be very specific about what sort of value you want/expect) should be "par for the course" on the developer's part. You're the developer; you should know EXACTLY what you want at any point in your code. Adding in check is, in my view, simply being more precise in my demand. The fact that it also eliminates potential bugs is an added bonus. "Warnings are errors waiting to happen", after all (at least - more likely than not; if you know exactly what you're doing, you can simpl be more explicit about it).

I am honestly appalled that there were absolutely no checks (apart from 'stripslashes', which looks to me like cargo cult tendencies i this case?) involved while dealing with user-provided input here.

It's not paranoia if they really ARE out to get you.

And we all know they are.
It's why Wordfence EXISTS.

Fabian *
March 1, 2022
11:54 pm

So the problem is with admin_menu being called even for unauthorized users. Would a nonce check be enough? Let's say I generate nonce when outputting the form on a page which can only be viewed with manage_options capability. Then checking for that nonce admin_menu would mean that I can be sure the user had the capabilities to view the form, right?
Which would be the correct hook to save post data in admin? I had a look and WooCommerce does it on wp_loaded with only a non check (no capability check).

Breaking WordPress Security Research in your inbox as it happens.

you@example.com

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.

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