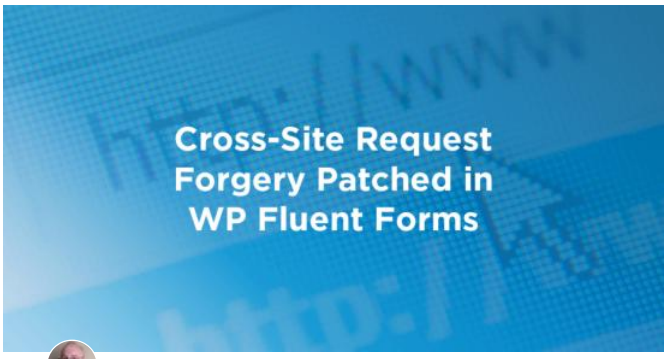

Ram Gall

June 16, 2021

Cross-Site Request Forgery Patched in WP Fluent Forms

On March 2, 2021, the Wordfence Threat Intelligence team responsibly disclosed a Cross-Site Request Forgery(CSRF) vulnerability in WP Fluent Forms, a WordPress plugin installed on over 80,000 sites. This vulnerability also allowed a stored Cross-Site Scripting(XSS) attack which, if successfully exploited, could be used to take over a site.

We reached out to the plugin developer, WP Manage Ninja, on March 2, 2021 and received a response within 24 hours. We sent over the full disclosure on March 3, 2021, and a patched version of the plugin, 3.6.67, was released on March 5, 2021.

As it was not possible to block attacks against this vulnerability without interfering with the plugin's basic functionality, we have withheld from sharing details of this vulnerability until the majority of sites using it are up to date. Nonetheless, we do not expect this vulnerability to be attacked at scale as it requires targeted social engineering to exploit.

Description: Cross-Site Request Forgery to stored Cross-Site Scripting(XSS)
Affected Plugin: WP Fluent Forms
Plugin Slug: fluentform
Affected Versions: < 3.6.67
CVE ID: CVE-2021-34620
CVSS Score: 7.1 (High)
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:L
Researcher/s: Ramuel Gall
Fully Patched Version: 3.6.67

WP Fluent Forms is a popular plugin that allows site owners to easily create and manage contact forms. It uses a large set of AJAX actions in order to modify global settings and perform most actions, and all non-public AJAX actions use a single method, `Acl::verify` in order to perform capability checks on these actions.

```php
public static function verify(
    $permission,
    $formId = null,
    $message = 'You do not have permission to perform this action.',
    $json = true
)
{
    $allowed = self::hasPermission($permission, $formId);
    if (!$allowed) {
        if ($json) {
            wp_send_json_error([
                'message' => $message
            ], 422);
        } else {
            throw new \Exception($message);
        }
    }
}

public static function hasPermission($permission, $formId = false)
{
    if (current_user_can('fluentform_full_access')) {
        return true;
    }

    $allowed = current_user_can('fluentform_full_access');
    if ($allowed) {
        return true;
    }

    if (is_array($permission)) {
        foreach ($permission as $eachPermission) {
            $allowed = current_user_can($eachPermission);
            if ($allowed) {
                return apply_filters('fluentform_verify_user_permission_' . $eachPermission, $allowed, $formId);
            } else {
                $isHookAllowed = apply_filters('fluentform_permission_callback', false, $eachPermission, $formId);
                if ($isHookAllowed) {
                    return true;
                }
            }
        }
        return false;
    }

    $allowed = current_user_can($permission);
    $allowed = apply_filters('fluentform_verify_user_permission_' . $permission, $allowed, $formId);

    if ($allowed) {
        return true;
    }

    return apply_filters('fluentform_permission_callback', false, $permission, $formId);
}
```

Unfortunately, however, all of these AJAX actions were vulnerable to Cross-Site Request Forgery because the `Acl::verify` method did not perform a nonce check, though it did use a second method, `Acl::hasPermission` in order to complete the capability check and allow for custom capabilities. As a result, it was possible to trick an administrator into sending a request (for instance, by clicking a link leading to an attacker-controlled page that sends an XHR request or contains a self-submitting form) to make arbitrary changes to the plugin. This attack would also work against any user that had been granted full permission to access the form settings.

The most severe consequence that we discovered was the ability to add arbitrary JavaScript to any form via the `fluentform-save-form-custom_css_js` AJAX action. If malicious JavaScript were added to a form, it would be executed in the browser of any visitor that accessed the form. If an administrator visited such a form, the JavaScript could be used to create an additional malicious administrative account or to add a backdoor to a plugin or theme file.

It was also possible for an attacker to trick an administrator into granting access to the plugin settings to subscriber-level users via the `fluentform_set_access_roles` AJAX action. In order for this to cause any real harm, the attacker would need to have at least a subscriber-level account on the targeted site and successfully socially engineer a vulnerable site owner. Again, an attacker would most likely abuse this access by adding malicious JavaScript to a form on the site.

In both cases, an attack would require targeted social engineering and would be unlikely to be exploited at scale. However, it would be trivial to redirect an administrator to a compromised form immediately after tricking them into submitting it, and the adversary would result in full site takeover.

Timeline

March 2, 2021 – The Wordfence Threat Intelligence team finishes researching vulnerabilities in WP Fluent Forms and initiates contact with the plugin's developer, WP Manage Ninja.
March 3, 2021 – We receive a response from WP Manage Ninja and send over the full disclosure.
March 5, 2021 – A patched version of WP Fluent Forms, 3.6.67, is released.

Conclusion

In today's article, we covered a Cross-Site Request Forgery(CSRF) vulnerability in WP Fluent Forms that could be used to perform a Cross-Site Scripting(XSS) attack which could potentially lead to site takeover. This vulnerability has been patched in version 3.6.67, and while most users of this plugin have updated to a patched version, we urge any remaining users that have not updated to update to the latest version available as soon as possible.

Did you enjoy this post? Share it!

Comments

No Comments