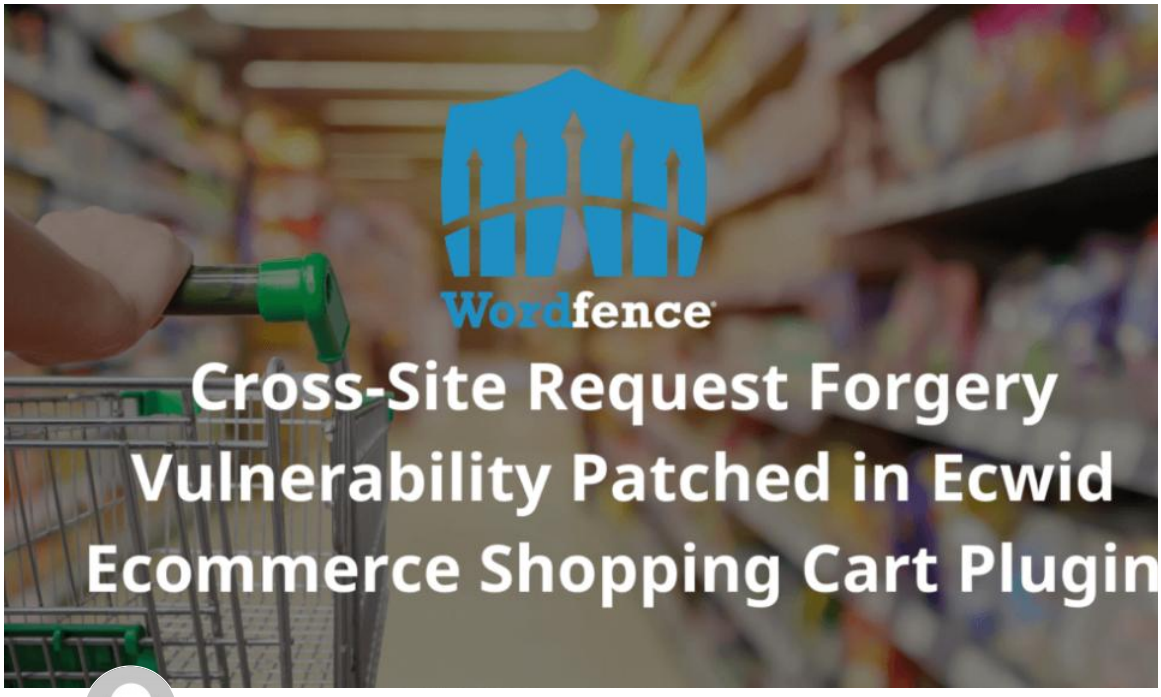

Marco Wotschka August 4, 2022

Cross-Site Request Forgery Vulnerability Patched in Ecwid Ecommerce Shopping Cart Plugin

On June 24, 2022, the Wordfence Threat Intelligence team initiated the responsible disclosure process for a Cross-Si Request Forgery vulnerability we discovered in Ecwid Ecommerce Shopping Cart, a WordPress plugin installed on ov 30,000 sites. This vulnerability made it possible for attackers to modify some of the plugin's more advanced settings a forged request.

We attempted to reach out to the developer on June, 24, 2022 via their ticketing system. After several plugin updates not address the issue and we received no response from the developer, we disclosed this vulnerability to the plugins team on July 11, 2022. The vulnerabilities were fixed a few days later in version 6.10.24 on July 13, 2022.

Due to the nature of Cross-Site Request Forgery vulnerabilities, which involve tricking administrators into performing actions they are allowed to perform, it is not possible to provide adequate protection for these vulnerabilities without blocking legitimate requests. As such, we highly recommend updating to version 6.10.24 or higher of Ecwid Ecomme Shopping Cart to ensure that your site is protected against any exploits targeting this vulnerability.

Description: Cross-Site Request Forgery to Settings/Options Update
Affected Plugin: Ecwid Ecommerce Shopping Cart
Plugin Slug: ecwid-shopping-cart
Affected Versions: <= 6.10.23
CVE ID: CVE-2022-2432
CVSS Score: 8.8 (High)
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
Researcher/s: Marco Wotschka
Fully Patched Version: 6.10.24

Ecwid Ecommerce Shopping Cart is a plugin offered by Lightspeed that allows site owners to set up an eCommerce store on a WordPress website and then sync it across various services such as Amazon and social media. It also off

settings that could be managed. Unfortunately, this was implemented insecurely making it possible for attackers to update these settings via a forged request.

More specifically, the plugin provides the following `admin_post` action hooked to the '`ecwid_update_plugin_params` function for that purpose:

```
1958 add_action('admin_post_' . ecwid_get_update_params_action(), 'ecwid_update_plugin_params');
```

An initial check in the '`ecwid_update_plugin_params`' function ensures that the current user is allowed to manage si options, which is a capability that belongs to administrators. However, the nonce check performed shortly after is on executed if a nonce is set due to how the function uses `&&` to combine the checks to validate that a nonce is present the request and to verify that the nonce is valid. This means that if the `wp-nonce` parameter is not supplied in the request, the verification step is skipped. This makes the functionality susceptible to Cross-Site Request Forgery attac in vulnerable versions.

```
1962 add_action('admin_post_' . ecwid_get_update_params_action(), 'ecwid_update_plugin_params');
1963 function ecwid_update_plugin_params()
1964 {
1965    if ( !current_user_can('manage_options') ) {
1966       wp_die( __( 'Sorry, you are not allowed to access this page.' ) );
1967    }
1968
1969    if ( isset($_POST['wp-nonce']) && !wp_verify_nonce(sanitize_text_field(wp_unslash($_POST['wp-nonce'])), ecwid
1970       wp_die( __( 'Sorry, you are not allowed to access this page.' ) );
1971    }
1972
1973    $options = ecwid_get_update_params_options();
1974
1975    $options4update = array();
1976
1977    foreach ( $options as $key => $option ) {
1978       if( !isset($_POST['option'][$key]) ) {
1979          continue;
1980       }
1981
1982       if ( isset($option['type']) && $option['type'] == 'html' ) {
1983          $options4update[$key] = sanitize_textarea_field(wp_unslash( $_POST['option'][$key] ));
1984       } else {
1985          $options4update[$key] = sanitize_text_field(wp_unslash( $_POST['option'][$key] ));
1986       }
1987
1988       if( $key == 'ecwid_store_id' ) {
1989          $options4update[$key] = intval($options4update[$key]);
1990       }
1991    }
1992
1993    foreach ($options4update as $name => $value) {
1994       update_option($name, $value);
1995    }
1996
1997    wp_safe_redirect('admin.php?page=ec-params');
1998    exit();
1999 }
```

This makes it possible for an attacker to change the `ecwid_store_id`, which uniquely identifies the store and is need for support requests as well as for embedding the store on other sites. Another site option that could be modified is t site's `ecwid_store_page_id`, which is the storefront page in the WordPress installation. Both of these settings may result in a loss of availability of the storefront if changed to an invalid store ID. There are several other settings that c be affected in addition to those two. The plugin does not provide a direct link to the settings page in any of its menus and a warning on the page suggests that modifying these settings may significantly affect the plugin functionality.

The Importance of Properly Implementing Nonces

Nonces are a critical component used to prevent Cross-Site Request Forgery vulnerabilities. As such, it's important tc ensure they are properly implemented throughout plugin and theme code to prevent unauthorized execution of that code. Fortunately, WordPress provides several mechanisms to create and validate proper nonces.

Nonce Creation

The first step to proper nonce implementation is nonce creation.

One option to properly implement nonce creation involves the `wp_nonce_url()` function. This function expects a targ

links to perform actions such as deletion of posts with a proper nonce that allows the code to verify that the action w initiated from within the site as opposed to a click on a link elsewhere, such as in an email.

An additional option is adding a nonce to a form to secure any submissions originating from that form. The function `wp_nonce_field()` is frequently used in these cases and expects an action string. This will add a hidden input field t the form. Upon form submission, the created nonce can be checked and verified.

Finally, a nonce can also be generated using `wp_create_nonce()`, which accepts an action string argument and retu just a nonce. This can be implemented in a variety of different ways and allows more flexibility as a developer to implement nonce validation. We frequently see this function contained in HTML on setting pages that is later used to validate the origin of the request when saving those settings.

Remember that nonces are specific to individual users and sessions and are invalidated after 24 hours, or on logout.

Nonce Validation

The second step to proper nonce implementation involves nonce verification. A plugin can implement an appropriate nonce on a form or settings update, but without proper validation of that nonce, there won't be adequate protection.

The first option to properly validate a nonce involves the use of the `check_admin_referer()` function. This function accepts the protected action as well as the name of the nonce as an argument and checks the nonce as well as the referer, thus helping to ensure that the nonce provided is correct and that the request was initiated from an admin pa

When implementing an AJAX action, the `check_ajax_referer()` function can be used. This will verify the nonce but not check the referer like the `check_admin_referer()` function does although it will validate that the request is an A request.

A final option for validating a nonce is the more general `wp_verify_nonce()` function. With this function, an action a nonce name will need to be supplied, and it will verify that the proper nonce was set. This provides the most flexible implementation of nonce validation for developers.

In the case of today's disclosure, the nonce was created on the settings form itself with `wp_create_nonce()`:

```
13 <input type="hidden" name="nonce" value="<?php echo wp_create_nonce( ecwid_get_update_params_action() ); ?>" />
```

However, the nonce verification was not performed correctly:

```
1969 if ( isset($_POST['wp-nonce']) && !wp_verify_nonce(sanitize_text_field(wp_unslash($_POST['wp-nonce'])), ecwid_ge
1970     wp_die( __( 'Sorry, you are not allowed to access this page.' ) );
1971 }
```

As you can see, this statement will call `wp_verify_nonce()` but only if the nonce is set due to the `if (isset($_POST['wp-nonce'])` check and the use of `&&` in combination with the `wp_verify_nonce()` function. Therefore, if the nonce is omitted, this check will not take place.

This serves as an important reminder to not only properly implement nonces and validation checks but also to ensur that the check fails when the nonce value is empty or otherwise not present.

As a final important note: never rely on nonce checks alone. Developers should always remember to include a capabi check using a function like `current_user_can()` in order to ensure the user initiating the action is indeed allowed to perform it. Nonces should never be used for authentication, authorization or any form of access control as they are simply intended to verify the origin of the request, and do not perform any authorization.

Timeline

- **June 24, 2022** – Initial outreach to the plugin developer.
- **July 11, 2022** – We escalate the issue to the WordPress.org plugins team and send them the full disclosure details.
- **July 13, 2022** – The vulnerability is patched in version 6.10.24.

Conclusion

administrators into updating plugin settings due to improper nonce verification. The vulnerability was patched by ensuring that a proper nonce was set and by verifying it.

Please remember that due to the nature of Cross-Site Request Forgery vulnerabilities, it is not possible to provide adequate protection via the Wordfence firewall without blocking legitimate requests. As such, we highly recommend updating to version 6.10.24 or higher of Ecwid Ecommerce Shopping Cart to ensure that your site is protected agains any exploits targeting this vulnerability.

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 of these products include hands-on suppo case you need further assistance.

If you have any friends or colleagues who are using this plugin, please share this announcement with them and encourage them to update to the latest patched version of Ecwid Ecommerce Shopping Cart as soon as possible.

Comments

2 Comments

Ilnur *
August 8, 2022
1:17 am

Hi, Marko!

This is Ilnur from Ecwid by Lightspeed team.

Thanks for your report! Appreciate it!

Unfortunately, on June 24 we did not receive your letter on this issue. We would fix it right away. Anyway, thank you.

If you have any reports to submit, please contact us at whitehat AT ecwid.com. More about our bug bounty program here https://github.com/Ecwid/security-bug-bounty.

Marco Wotschka *
August 8, 2022
10:07 am

Thanks for sharing this information! We were not aware you had a disclosure program, and did not find those details when we made our initial outreach. We will be sure to use that address if we ever discover any issues in the future. Hopefully this information can also help any other security researchers that may discover any issues.

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