Chloe Chamberland

May 11, 2020

Vulnerabilities Patched in Page Builder by SiteOrigin Affects Over 1 Million Sites

On Monday, May 4, 2020, the Wordfence Threat Intelligence team discovered two vulnerabilities present in Page Builder by SiteOrigin, a WordPress plugin actively installed on over 1,000,000 sites. Both of these flaws allow attackers to forge requests on behalf of a site administrator and execute malicious code in the administrator's browser. The attacker needs to trick a site administrator into executing an action, like clicking a link or an attachment, for the attack to succeed.

We first contacted the plugin developer on May 4, 2020. After establishing an appropriate communication channel, we provided the full disclosure later that day. The developer quickly released a patch the next day, which was May 5, 2020.

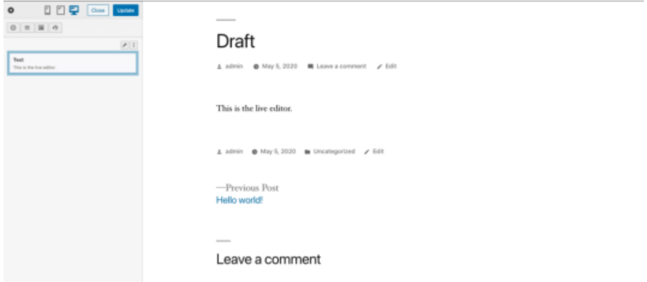
These are considered high-risk security issues that could lead to full site takeover. We recommend an immediate update of Page Builder by SiteOrigin to the latest version available. At the time of writing, that is version 2.10.16.

Both the free and Premium version of the Wordfence firewall protect against these vulnerabilities via the built in cross site scripting (XSS) protection in the Wordfence firewall.

Description: Cross-Site Request Forgery to Reflected Cross-Site Scripting
Affected Plugin: Page Builder by SiteOrigin
Plugin Slug: siteorigin-panels
Affected Versions: <= 2.10.15
CVE ID: CVE-2020-13643
CVSS Score: 8.8 (High)
CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
Fully Patched Version: 2.10.16

Page Builder by SiteOrigin is a plugin that simplifies page and post editing in WordPress. Users can create "responsive column based content" using both widgets from WordPress and widgets from the SiteOrigin Widgets Bundle plugin.

The plugin has a built-in live editor so users can update content and drag/drop widgets while observing those changes made in real time. This makes the editing and designing for a page or post a much smoother process.



A view of the Page Builder by SiteOrigin live editor feature.

In order to show the modifications in real time through the live editor, the plugin registers the `is_live_editor()` function to check if a user is in the live editor.

```
static function is_live_editor(){
  return ! empty( $_GET['siteorigin_panels_live_editor'] );
}
```

If the user is in the live editor, the `siteorigin_panels_live_editor` parameter will be set to "true" and register that a user is accessing the live editor. The plugin will then attempt to include the live editor file which renders all of the content.

```
// Include the live editor file if we're in live editor mode.
    if ( self::is_live_editor() ) {
        SiteOrigin_Panels_Live_Editor::single();
    }
```

Any content changes made are set and sent as a $_POST parameter: `live_editor_panels_data`. This parameter contains all of the content that has been edited by the user and is required to render a live preview of the changes.

Once a change has been made, the live-editor-preview.php file is used to render the content provided from the `live_editor_panels_data` parameter and update the page preview displaying any changes made changes in real-time.

```
<?php if ( ! defined( 'ABSPATH' ) ) { exit; // Exit if accessed directly. } wp_enqueue_style( 'siteorigin-preview-styl
<!DOCTYPE html>
```

```
    <link rel="profile" href="http://gmpg.org/xfn/11">
    <link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>">
    <?php wp_head(); ?>
</head>

<body <?php body_class(); ?>>

<div id="content" class="site-content">

<div class="entry-content">
            <?php if( !empty( $_POST['live_editor_panels_data'] ) ) { $data = json_decode( wp_unslash( $_POST['live_ed
                }
                echo siteorigin_panels_render( 'l' . md5( serialize( $data ) ), true, $data);
            }
            ?>
        </div>

<!-- .entry-content -->
    </div>

    <?php wp_footer(); ?>
</body>
</html>
```

While capability checks were present in the `post_metadata` function to ensure a user accessing the live editor was allowed to edit posts, there was no nonce protection to verify that an attempt to render content in the live editor came from an legitimate source.

```
function post_metadata( $value, $post_id, $meta_key ) {
    if (
        $meta_key == 'panels_data' &&
        current_user_can( 'edit_post', $post_id ) &&
        ! empty( $_POST['live_editor_panels_data'] ) &&
        $_POST['live_editor_post_ID'] == $post_id
    ) {
        $value = array( json_decode( wp_unslash( $_POST['live_editor_panels_data'] ), true ) );
    }
    return $value;
}
```

As part of this exploit, some of the available widgets, such as the "Custom HTML" widget, could be used to inject malicious JavaScript into a rendered live page. If a site administrator was tricked into accessing a crafted live preview page, any malicious Javascript included as part of the "Custom HTML" widget could be executed in the browser. The data associated with a live preview was never stored in the database, resulting in a reflected XSS flaw rather than stored XSS flaw, in conjunction with the CSRF flaw.

This flaw could be used to redirect a site's administrator, create a new administrative user account, or, as seen in the recent attack campaign targeting XSS vulnerabilities, be used to inject a backdoor on a site.

Description: Cross-Site Request Forgery to Reflected Cross-Site Scripting
Affected Plugin: Page Builder by SiteOrigin
Plugin Slug: siteorigin-panels
Affected Versions: <= 2.10.15
CVE ID: CVE-2020-13642
CVSS Score: 8.8 (High)
CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
Fully Patched Version: 2.10.16

We discovered an additional Cross-Site Request Forgery flaw in the `action_builder_content` function of the plugin. This was tied to the AJAX action `wp_ajax_so_panels_builder_content`.

```
add_action( 'wp_ajax_so_panels_builder_content', array( $this, 'action_builder_content' ) );
```

This function's purpose was to transmit content submitted as panels_data from the live editor to the WordPress editor in order to update or publish the post using the content created from the live editor. This function did have a permissions check to verify that a user had the capability to edit posts for the given `post_id`. However, there was no nonce protection to verify the source of a request, causing the CSRF flaw.

```
/**
 * Get builder content based on the submitted panels_data.
 */
function action_builder_content() {
    header( 'content-type: text/html' );

    if ( ! current_user_can( 'edit_post', $_POST['post_id'] ) ) {
        wp_die();
    }
```

As part of the process, the content from the `panels_data` function was purposefully retrieved so that could be echoed to the page.

```
if ( empty( $_POST['post_id'] ) || empty( $_POST['panels_data'] ) ) {
            echo '';
            wp_die();
        }

        // echo the content
        $old_panels_data        = get_post_meta( $_POST['post_id'], 'panels_data', true );
        $panels_data            = json_decode( wp_unslash( $_POST['panels_data'] ), true );
        $panels_data['widgets'] = $this->process_raw_widgets(
            $panels_data['widgets'],
            ! empty( $old_panels_data['widgets'] ) ? $old_panels_data['widgets'] : false,
            false
        );
        $panels_data            = SiteOrigin_Panels_Styles_Admin::single()->sanitize_all( $panels_data );
```

With this function, the "Custom HTML" widget did not create an XSS flaw in the same way as the previous vulnerability due to some sanitization features. However, we discovered that the "Text" widget could be used to inject malicious JavaScript due to the ability to edit content in a 'text' mode rather than a 'visual' mode. This allowed potentially malicious JavaScript to be sent unfiltered. Due to the widget data being echoed, any malicious code that was a part of the text widgets data could then be executed as part of a combined CSRF to XSS attack in a victim's browser.

As with the previously mentioned CSRF to Reflected XSS vulnerability, this could ultimately be used to redirect a site's administrator, create a new administrative user account, or, as seen in the recent attack campaign targeting XSS vulnerabilities, be used to inject a backdoor on a site.

Proof of Concept Walkthrough

Disclosure Timeline

May 4, 2020 – Initial discovery and analysis of vulnerabilities. We verify the Wordfence built-in XSS firewall rule offers sufficient protection. Initial outreach to the plugin's team.

May 5, 2020 – Developer acknowledges vulnerabilities and advises that they should have a patch released later in the day.
May 5, 2020 – A sufficient patch is released.

Conclusion

In today's post, we detailed two flaws present in the Page Builder by SiteOrigin plugin that allowed attackers to forge requests on behalf of a site administrator and execute malicious code in that administrator's browser. These flaws have been fully patched in version 2.10.16. We recommend that users immediately update to the latest version available.

Sites running Wordfence Premium, as well as sites using the free version of Wordfence, are protected from Cross-Site Scripting attacks against this vulnerability due to the Wordfence firewall's built-in protection. If you know a friend or colleague who is using this plugin on their site, we highly recommend forwarding this advisory to them to help keep them protected.

Special thanks to Greg Priday, the plugin's developer, for an extremely prompt response and for releasing a patch very quickly.

Did you enjoy this post? Share it!

Comments

5 Comments

Anna Anderson *
May 11, 2020
4:03 pm

Great job guys! Your proactive scanning and testing plugins for potential vulnerablities is keeping our websites safe.

I've had the Premium Vantage theme by SiteOrigin on my website for few years now, using their Page Builder of course. Great, solid theme with many widgets, well maintained, with fantastic support. I would have never thought that it could have any vulnerabilities - but there you go...

I'm sure the plugin developers are grateful for your discovery, too. Likewise, they are a great team.

Well done and thank you for your hard work!

johnzoet *
May 11, 2020
5:32 pm

What again was the tool used in this video for tracing the HTTP request and forging a XSS request ?

Chloe Chamberland *
May 12, 2020
10:02 am

Hi John!

The tool I was using is called Burp Suite.

zakopiec *
May 12, 2020
7:02 am

Good Job guys. . I going to use your plugin on my site after migrate to better hosting.
regards

Gracious Store *
May 16, 2020
4:30 pm

Thanks for all you do to protect our sites against those bad guys

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