Chloe Chamberland

March 25, 2021

Two Vulnerabilities Patched in Facebook for WordPress Plugin

On December 22, 2020, our Threat Intelligence team responsibly disclosed a vulnerability in Facebook for WordPress, formerly known as Official Facebook Pixel, a WordPress plugin installed on over 500,000 sites. This flaw made it possible for unauthenticated attackers with access to a site's secret salts and keys to achieve remote code execution through a deserialization weakness.

In addition, on January 27, 2021, we disclosed a separately identified vulnerability in Facebook for WordPress that was introduced in the rebranding of the plugin in version 3.0.0. This flaw made it possible for attackers to inject malicious JavaScript into the plugin's settings, if an attacker could successfully trick an administrator into performing an action such as clicking a link.

We initially reached out to Facebook's security team on December 22, 2020 for the first vulnerability and included the full disclosure details at the time of reaching out. They initially responded on December 25, 2020 requesting further information which was supplied on December 26, 2020. A patch was released on January 6, 2021.

For the second vulnerability, we reached out to Facebook's security team again on January 27, 2021 and included the full disclosure details at the time of reaching out. They initially responded on February 1, 2021 requesting further information which was supplied the same day. An initial patch was released on February 12, 2021, and a fully sufficient patch was released on February 17, 2021.

These are considered high and critical severity vulnerabilities. Therefore, we highly recommend updating to the latest version available containing both patches, 3.0.5, immediately.

Wordfence Premium users received a firewall rule to protect exploit attempts against the first vulnerability on December 22, 2020 and received a firewall rule to protect exploit attempts against the second vulnerability on January 27, 2021. Sites still using the free version of Wordfence received the same protection for the first vulnerability on January 21, 2020 and on February 26, 2021 for the second vulnerability.

PHP Object Injection

Description: PHP Object Injection with POP Chain
Affected Plugin: Facebook for WordPress, *formerly known as Official Facebook Pixel*
Plugin Slug: official-facebook-pixel
Affected Versions: < =2.2.2
CVE ID: CVE-2021-24217
CVSS Score: 9.0 (CRITICAL)
CVSS Vector: CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H
Fully Patched Version: 3.0.0

Facebook for WordPress is a plugin designed to create a seamless integration between Facebook Pixel, a conversion measurement tool, and a WordPress site. The plugin will monitor site traffic and record data when users access pages and perform certain actions on a site.

The plugin registered an `admin_post` action, `admin_post_wp_async_send_server_event`, so that both the ViewContent and AddToCart events would be triggered at the same time when WooCommerce customers added products to their cart.

```
76  public function __construct( $auth_level = self::BOTH ) {
77      if ( empty( $this->action ) ) {
78          throw new Exception( 'Action not defined for class ' . __CLASS__ );
79      }
80      add_action( $this->action, array( $this, 'launch' ), (int) $this->priority, (int) $this->argument_count );
81      if ( $auth_level & self::LOGGED_IN ) {
82          add_action( "admin_post_wp_async_$this->action", array( $this, 'handle_postback' ) );
83      }
84      if ( $auth_level & self::LOGGED_OUT ) {
85          add_action( "admin_post_nopriv_wp_async_$this->action", array( $this, 'handle_postback' ) );
86      }
87  }</pre>
88  <pre>
```

This action was hooked to the `handle_postback` function that ultimately checked for a valid nonce and triggered the `run_action()` function when a valid nonce was supplied.

```
159 public function handle_postback() {
160     if ( isset( $_POST['_nonce'] ) && $this->verify_async_nonce( $_POST['_nonce'] ) ) {
161         if ( ! is_user_logged_in() ) {
162             $this->action = "nopriv_$this->action";
163         }
164         $this->run_action();
165     }
166
167     add_filter( 'wp_die_handler', function() { die(); } );
168     wp_die();
169 }</pre>
170 <pre>
```

Due to the `handle_postback` function requiring a valid nonce, an attacker would need a valid nonce in order to exploit this vulnerability. The generated nonce did not use normal user data when creating a nonce, making the nonce unrelated to any user sessions. We discovered that this nonce could easily be generated with a custom script. This required that an attacker gain access to the site's nonce salt and key either through a SQL injection vulnerability, a directory traversal vulnerability, or a publicly accessible backup of the `wp-config.php` file.

The core of the PHP Object Injection vulnerability was within the `run_action()` function. This function was intended to deserialize user data from the `event_data` POST variable so that it could send the data to the pixel console. Unfortunately, this `event_data` could be supplied by a user. When user-supplied input is deserialized in PHP, users can supply PHP objects that can trigger magic methods and execute actions that can be used for malicious purposes.

```
protected function run_action() {
  try {
```

```
    $events = unserialize(base64_decode($_POST['event_data']));
    // When an array has just one object, the deserialization process
    // returns just the object
    // and we want an array
    if( $num_events == 1 ){
      $events = array( $events );
    }

    FacebookServerSideEvent::send($events);
  }</pre>
<pre>
```

On its own, a deserialization vulnerability is relatively harmless, however, when combined with a gadget, or magic method, significant damage can be done to a site. In this case, we did find a magic method within the plugin that could be used to upload arbitrary files and achieve remote code execution on a vulnerable target.

The Complete POP Chain

The Facebook for WordPress plugin uses Guzzle, a PHP HTTP client, which has the `__destruct` magic method used in the FileCookieJar class. This method saves a file when shutting down to store sessions cookies. This meant that a properly constructed payload that calls the FileCookieJar class could allow an attacker to create a file.

```
<?php
namespace GuzzleHttp\Cookie;

/**
 * Persists non-session cookies using a JSON formatted file
 */
class FileCookieJar extends CookieJar
{
    /** @var string filename */
    private $filename;

    /** @var bool Control whether to persist session cookies or not. */
    private $storeSessionCookies;
```

```
/**
 * Saves the file when shutting down
 */
public function __destruct()
{
    $this->save($this->filename);
}

/**
 * Saves the cookies to a file.
 *
 * @param string $filename File to save
 * @throws \RuntimeException if the file cannot be found or created
 */
public function save($filename)
{
    $json = [];
    foreach ($this as $cookie) {
        /** @var SetCookie $cookie */
        if (CookieJar::shouldPersist($cookie, $this->storeSessionCookies)) {
            $json[] = $cookie->toArray();
        }
    }

    $jsonStr = \GuzzleHttp\json_encode($json);
    if (false === file_put_contents($filename, $jsonStr, LOCK_EX)) {
        throw new \RuntimeException("Unable to save file {$filename}");
    }
}
```

We were able to construct the following payload, with the use of an external tool, PHPGGC, that calls the FileCookieJar class once deserialized. This payload sets the FileCookieJar filename to `/var/www/html/new.php`, which is the path and filename of the file that will be created during deserialization. The SetCookie data parameter sets the malicious PHP payload in the newly created file to:`<?php phpinfo();?>`. Finally, it also sets a few other values like FileCookieJar storeSessionCookies to enabled and SetCookie to true.

```
O:31:"GuzzleHttp\Cookie\FileCookieJar":4:{s:41:"GuzzleHttp\Cookie\FileCookieJar*filename";s:21:"/var/www/html/new.php";
```

This meant that an attacker could generate a PHP file new.php in a vulnerable site's home directory with the contents `<?php phpinfo();?>`. The PHP file contents could be changed to anything, like `<?php shell_exec($_GET['cmd']);?>` which would allow an attacker to achieve remote code execution.

Note that the presence of a full POP chain also meant that any other plugin with an object injection vulnerability, including those that did not require knowledge of the site's salts and keys, could potentially be used to achieve remote code execution as well if it was installed on a site with the Facebook for WordPress plugin.

Cross-Site Request Forgery

Description: Cross-Site Request Forgery to Stored Cross-Site Scripting
Affected Plugin: Facebook for WordPress, *formerly known as Official Facebook Pixel*
Plugin Slug: official-facebook-pixel
Affected Versions: 3.0.0 – 3.0.3
CVE ID: CVE-2021-24218
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
Fully Patched Version: 3.0.4

In addition to the PHP Object Injection vulnerability, we discovered a Cross-Site Request Forgery to Stored Cross-Site Scripting vulnerability in the updated version of the plugin. In version 3.0.0, the developers rewrote much of the plugin's code while rebranding and introducing some new functionality.

One of the changes they made while updating the plugin addressed the functionality behind saving the plugin's settings. This was converted to an AJAX action to make the integration process more seamless. The new version introduced the `wp_ajax_save_fbe_settings` AJAX action tied to the `saveFbeSettings` function.

```
public function init(){
      add_action('wp_ajax_save_fbe_settings', array($this, 'saveFbeSettings'));
      add_action('wp_ajax_delete_fbe_settings',
      array($this, 'deleteFbeSettings')
      );
  }
```

This function is used to update the plugin's settings with the Facebook Pixel ID, access token, and external business key. These settings help establish a connection with the Facebook pixel console so that event data can be sent from the WordPress site to the appropriate Facebook pixel account.

```
public function saveFbeSettings(){
    if (!current_user_can('administrator')) {


    $external_business_id = $_POST['externalBusinessId'];
    $settings = array(
        FacebookPluginConfig::PIXEL_ID_KEY => $pixel_id,
        FacebookPluginConfig::ACCESS_TOKEN_KEY => $access_token,
        FacebookPluginConfig::EXTERNAL_BUSINESS_ID_KEY =>
            $external_business_id,
        FacebookPluginConfig::IS_FBE_INSTALLED_KEY => '1'
    );
    \update_option(
        FacebookPluginConfig::SETTINGS_KEY,
        $settings
    );
    return $this->handleSuccessRequest($settings);
}
```

There was a permission check on this function, blocking users lower than administrators from being able to access it, however, there was no nonce protection. This meant that there was no verification that a request was coming from a legitimate authenticated administrator session. This made it possible for attackers to craft a request that would be executed if they could trick an administrator into performing an action while authenticated to the target site.

The action could be used by an attacker to update the plugin's settings to point to their own Facebook Pixel console and steal metric data for a site. Worse yet, since there was no sanitization on the settings that were stored, an attacker could inject malicious JavaScript into the setting values. These values would then be reflected on the settings page, causing the code to execute in a site administrator's browser while accessing the settings page. Ultimately, this code could be used to inject malicious backdoors into theme files or create new administrative user accounts that could be used for complete site takeover.

In addition to the AJAX action used to update settings, there was an AJAX action `wp_ajax_delete_fbe_settings` hooked to the `deleteFbeSettings` function. This function was used to erase the plugin's settings. It also lacked the appropriate nonce protection, making it possible for attackers to reset plugin settings through CSRF. Though it could be annoying for the plugin's settings to be reset, and it could cause a brief loss of metric data, there was no overly impactful harm that could be done on a WordPress site with this vulnerability.

Disclosure Timeline [PHP Object Injection]

December 22, 2020 – Conclusion of the plugin analysis that led to the discovery of the PHP Object Injection vulnerability in Facebook for WordPress. We develop firewall rules to protect Wordfence customers and release them to Wordfence Premium users. We initiate contact with the Facebook Pixel security team. We receive automated responses confirming receipt of our report.
December 25, 2020 – We receive a response requesting additional information.
December 28, 2020 – We send additional requested details.
January 4, 2021 – We receive confirmation that the plugin developers were able to duplicate our report and have begun working on a fix.
January 6, 2021 – A patched version of the plugin is released as version 3.0.0. We verify that the vulnerability has been patched.
January 21, 2021 – Free Wordfence users receive firewall rule.

Disclosure Timeline [CSRF to Stored XSS]

January 27, 2021 – Conclusion of the plugin analysis that led to the discovery of the CSRF to Stored XSS vulnerability in Facebook for WordPress. We develop firewall rules to protect Wordfence customers and release them to Wordfence Premium users. We initiate contact with the Facebook Pixel security team. We receive automated responses confirming receipt of our report.
February 1, 2021 – We receive a response requesting additional information. We send additional details the same day.
February 5, 2021 – We receive confirmation that the plugin developers were able to duplicate our report and have begun working on a fix.
February 12, 2021 – A patched version of the plugin is released as version 3.0.3.
February 15, 2021 –We follow up to inform the plugin developers of missing sanitization on the stored settings leading to an insufficient patch.
February 17, 2021 – An additional and fully sufficient patch is released.
February 26, 2021 – Free Wordfence users receive firewall rule.

Conclusion

In today's post, we detailed two flaws in the Facebook for WordPress plugin that granted attackers the ability to achieve remote code execution due to a PHP Object Injection vulnerability and inject malicious JavaScript due to a CSRF vulnerability, both of which can be used for complete site takeover. The flaws are both fully patched in version 3.0.4. We recommend that users immediately update to the latest version available, which is version 3.0.5 at the time of this publication.

Wordfence Premium users received firewall rules protecting against the first vulnerability on December 22, 2020 and the second vulnerability on January 27, 2021, while those still using the free version of Wordfence received the same protection for the first vulnerability on January 21, 2020 and on February 26, 2021 for the second vulnerability.

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 their sites protected as these are severe vulnerabilities that can result in remote code execution and full site compromise.
Did you enjoy this post? Share it!

Comments

2 Comments

Robert *
March 26, 2021
7:13 am

How about the "Facebook for woocommerce" plugin is it also affected, they seem to share a decent amount of code base?

Chloe Chamberland *
March 29, 2021
9:28 am

Hi Robert,

This does not affect the "Facebook for WooCommerce" plugin.

Thanks!

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