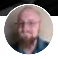
Vulnerabilities Patched in Popup Builder Plugin Affecting Over 100,000 Sites



Ram Gall

March 12, 2020

Vulnerabilities Patched in Popup Builder Plugin Affecting over 100,000 Sites

On March 4th, our Threat Intelligence team discovered several vulnerabilities in Popup Builder, a WordPress plugin installed on over 100,000 sites. One vulnerability allowed an unauthenticated attacker to inject malicious JavaScript into any published popup, which would then be executed whenever the popup loaded. The other vulnerability allowed any logged-in user, even those with minimal permissions such as a subscriber, to export a list of all newsletter subscribers, export system configuration information, and grant themselves access to various features of the plugin.

We privately disclosed these issues to the plugin's author, who responded within a few hours. We worked with the developer over the course of a week to ensure the vulnerabilities were fully patched.

We highly recommend updating to the latest version, 3.64.1, immediately. Wordfence Premium customers received a new firewall rule on March 5, 2020 to protect against exploits targeting these vulnerabilities. Free Wordfence users will receive the rule after thirty days, on April 4, 2020.

Description: Unauthenticated Stored Cross-Site Scripting (XSS)
Affected Plugin: Popup Builder – Responsive WordPress Pop up – Subscription & Newsletter
Plugin Slug: popup-builder
Affected Versions: <= 3.63
CVE ID: CVE-2020-10196
CVSS Score: 8.3 (High)
CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:L
Fully Patched Version: 3.64.1

The Popup Builder plugin allows the creation of various popups on a WordPress site, which includes the ability to run custom Javascript when a popup is loaded. It registered an AJAX hook, wp_ajax_nopriv_sgpb_autosave, intended to allow auto-saving of draft popups.

```
add_action('wp_ajax_nopriv_sgpb_autosave', array($this, 'sgpbAutosave'));
```

Unfortunately, this hook was available to unprivileged users, and the function it called lacked nonce checks or capability checks. This meant that an unauthenticated attacker could send a POST request to wp-admin/admin-ajax.php with an array parameter, 'allPopupData', containing a number of key-value pairs including a popup's ID (visible in the page source) and a malicious JavaScript payload, which would then be saved in that popup's settings and executed whenever a visitor navigated to a page where the popup was displayed.

```
public function sgpbAutosave()
{
    $popupId = @(int)$_POST['post_ID'];
    $postStatus = get_post_status($popupId);
    if ($postStatus == 'publish') {
        echo '';
        wp_die();
    }

    if (!isset($_POST['allPopupData'])) {
        echo true;
        wp_die();
    }
    $popupData = SGPopup::parsePopupDataFromData($_POST['allPopupData']);
    do_action('save_post_popupbuilder');
    $popupType = $popupData['sgpb-type'];
    $popupClassName = SGPopup::getPopupClassNameFormType($popupType);
    $popupClassPath = SGPopup::getPopupTypeClassPath($popupType);
    if (file_exists($popupClassPath.$popupClassName.'.php')) {
        require_once($popupClassPath.$popupClassName.'.php');
        $popupClassName = __NAMESPACE__.'\\'.$popupClassName;
        $popupClassName::create($popupData, '_preview', 1);
    }

    wp_die();
}
```

Note the lack of nonce and permission checks in this function. The function does attempt to prevent changes being saved to any popup in 'publish' status. However, if no 'post_ID' parameter is supplied, this check will be bypassed and the post id supplied in the 'allPopupData' parameter will be updated instead.

Typically, attackers use a vulnerability like this to redirect site visitors to malvertising sites or steal sensitive information from their browsers, though it could also be used for site takeover if an administrator visited or previewed a page containing the infected popup while logged in.

Description: Authenticated Settings Modification, Configuration Disclosure, and User Data Export
Affected Plugin: Popup Builder – Responsive WordPress Pop up – Subscription & Newsletter
Plugin Slug: popup-builder
Affected Versions: <= 3.63
CVE ID: CVE-2020-10195
CVSS Score: 6.3 (Medium)
CVSS Vector: CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L
Fully Patched Version: 3.64.1

In addition to a stored XSS vulnerability, Popup Builder also had a set of vulnerabilities that could be exploited by logged-in users with minimal permissions, such as subscribers. The vulnerable actions included:

```
add_action('admin_post_csv_file', array($this, 'getSubscribersCsvFile'));
add_action('admin_post_sgpb_system_info', array($this, 'getSystemInfoFile'));
```

```
add_action('admin_post_sgpbSaveSettings', array($this, 'saveSettings'), 10, 1);
```

By sending a $_POST request to admin-post.php with the 'action' parameter set to 'sgpbSaveSettings' and the 'sgpb-user-roles[]' parameter set to 'subscriber', an attacker could grant all subscriber-level users (including themselves) a number of permissions related to the plugin's functionality. In addition to granting access to create and manage categories and newsletters, this would allow an attacker to make use of other AJAX functions that *were* protected by nonces, but not by capability checks, since usable nonces were displayed on these pages.

The vulnerable function code:

```php
public function saveSettings()
{
    $postData = $_POST;
    $deleteData = 0;

    if (isset($postData['sgpb-dont-delete-data'])) {
        $deleteData = 1;
    }
    $userRoles = @$postData['sgpb-user-roles'];

    update_option('sgpb-user-roles', $userRoles);
    update_option('sgpb-dont-delete-data', $deleteData);

    wp_redirect(admin_url().'edit.php?post_type='.SG_POPUP_POST_TYPE.'&page='.SG_POPUP_SETTINGS_PAGE);
}
```

Alternatively, a $_POST request could be sent to admin-post.php with the 'action' parameter set to 'csv_file', making it possible to export a list of newsletter subscribers. As a result, an attacker could gain access to sensitive newsletter subscriber information and use this during a social engineering attack against those subscribers.

The vulnerable function code:

```php
public function getSubscribersCsvFile()
{
    global $wpdb;
    $query = AdminHelper::subscribersRelatedQuery();
    if (isset($_GET['orderby']) && !empty($_GET['orderby'])) {
        if (isset($_GET['order']) && !empty($_GET['order'])) {
            $query .= ' ORDER BY '.esc_sql($_GET['orderby']).' '.esc_sql($_GET['order']);
        }
    }
    $content = '';
    $exportTypeQuery = '';
    $rows = array('first name', 'last name', 'email', 'date', 'popup');
    foreach ($rows as $value) {
        $content .= $value;
        if ($value != 'popup') {
            $content .= ',';
        }
    }
    $content .= "\n";
    $subscribers = $wpdb->get_results($query, ARRAY_A);

    $subscribers = apply_filters('sgpbSubscribersCsv', $subscribers);

    foreach($subscribers as $values) {
        foreach ($values as $key => $value) {
            $content .= $value;
            if ($key != 'subscriptionTitle') {
                $content .= ',';
            }
        }
        $content .= "\n";
    }

    $content = apply_filters('sgpbSubscribersContent', $content);

    header('Pragma: public');
    header('Expires: 0');
    header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
    header('Cache-Control: private', false);
    header('Content-Type: application/octet-stream');
    header('Content-Disposition: attachment; filename=subscribersList.csv;');
    header('Content-Transfer-Encoding: binary');
    echo $content;
}
```

Furthermore, the 'action' parameter could be changed to 'sgpb_system_info' and reveal potentially sensitive system configuration information, including all installed plugins and their activation status. This data could be used by an attacker to craft a more sophisticated attack against a target site. If another vulnerable plugin was installed on the site, an attacker could discover this and attempt to escalate their attack by exploiting it.

The vulnerable function code:

```php
public function getSystemInfoFile()
{
    $content = AdminHelper::getSystemInfoText();

    header('Pragma: public');
    header('Expires: 0');
    header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
    header('Cache-Control: private', false);
    header('Content-Type: application/octet-stream');
    header('Content-Disposition: attachment; filename=popupBuilderSystemInfo.txt;');
    header('Content-Transfer-Encoding: binary');

    echo $content;
}
```

Disclosure Timeline

March 4, 2020 – Wordfence Threat Intelligence discovers and analyzes vulnerabilities in the Popup Builder plugin.
March 5, 2020 – Firewall rule released for Wordfence Premium users. Initial outreach to plugin vendor. Plugin vendor responds within a few hours, and we send over the full vulnerability report.
March 6, 2020 – Plugin vendor sends patched version to us for review. Additional guidance provided to strengthen security.
March 11, 2020 – Fully patched version released.
April 4, 2020 – Firewall rule available to free users.

Conclusion

In today's post, we detailed several vulnerabilities including unauthenticated stored XSS, settings modification, configuration disclosure, and user data export found in the Popup Builder plugin. These flaws have been patched in version 3.64.1 and we recommend that users update to the latest version available immediately. While we have not detected any malicious activity targeting Popup Builder, the stored XSS vulnerability can have a serious impact on site visitors and potentially even allow site takeover. Sites running Wordfence Premium have been protected from attacks against these vulnerabilities since March 5, 2020. Sites running the free version of Wordfence will receive the same firewall rule update on April 4, 2020.

Did you enjoy this post? Share it!

Comments

5 Comments

Marshall *
March 12, 2020
6:05 pm

Following all these vulnerability notices from Wordfence has convinced me I will never be a plugin developer. It seems most of what

Ram Gall *
March 13, 2020
10:48 am

Hi Marshall!
There are indeed a number of learning resources available that include examples of how to perform capability checks and use nonces. Unfortunately, some older learning resources fail to emphasize security, so beginner developers still make mistakes. Additionally, some plugins contain hastily-added features, or still have technical debt in the form of unresolved security issues, even if their newer features are properly protected, but at this point in time the majority of popular plugins seem to use capability checks and nonces correctly.

Gracious Store *
March 15, 2020
9:13 pm

Those of us that are not developers but simple site owners depend on the expertise of the developers to keep our sites safe , for that reason we pay the yearly subscriptions.

Fred *
March 18, 2020
5:48 am

Hello everybody!
I found many of my sites (with Popup builder installed) with folder permissions set to "write", and it seems really strange (I have never changed them). In your opinion, is it possible that this attack had some influence on these permissions?
Thanks for those who want to answer!

Ram Gall *
March 18, 2020
7:19 am

Hi Fred!
None of the vulnerabilities we found allowed direct file or folder access, but stored XSS can be used to create new administrative accounts if an existing admin visits an affected page, and an attacker with administrative capabilities could make this sort of change on some configurations. We recommend checking to see if any unfamiliar administrative accounts have been setup on your site - if not, the issue you're experiencing is likely unrelated.