SYNACKTIV
DIGITAL SECURITY

SQL injection in Image Intense <= 3.2.5

Security advisory

2018-08-24

Julien Legras
Thomas Chauchefoin

Vulnerability description

Presentation of *Image-Intense*

"Image Intense is an astonishing hybrid mix of 3 native Divi modules for a truly compelling, superfly UI/UX experience on your website. Comes with 22 different overlay and hover effects plus a multitude of image, text and button options to give you a brand new world of possibilities for your images."[1]

The issue

Synacktiv discovered that the *WordPress* plugin *Image Intense* does not correctly sanitize user-controlled data before using it in SQL queries while handling the *shortcode* `et_pb_image_n10s`. Thus, an attacker could abuse the affected feature to alter the semantic original SQL query and access sensitive database records.

It should be noted that the attacker must be able to create pages / articles on the instance (*Contributor*, *Author*, *Editor* or *Administrator*) to reach the vulnerable code path.

Mitigation

Such code patterns:

```
$sql = "SELECT ID FROM `" . $table . "` WHERE guid='" . $site_url . $image_url . "';";
$attachment = $wpdb->get_col( $sql );
```

Should be replaced by calls to `$wpdb->prepare`, where all the parameters of the SQL query are passed as second argument, as shown in the following example:

```
$attachment = $wpdb->get_results( $wpdb->prepare( "SELECT ID FROM {$table} WHERE guid =
%s;" ), $site_url . $image_url );
```

Affected versions

The last version at the time of this advisory, 3.2.5, is known to be affected.

Timeline

Date	Action
2018-07-16	Advisory sent to *BeSuperFly*.
2018-07-17	The editor does not consider it to be a vulnerability.
2018-08-24	Publication of the advisory.

1 https://besuperfly.com/product/image-intense-plugin/

Technical description and proof-of-concept

The *Image Intense WordPress* plugin allows authors to include images in posts and pages using the *Divi* builder. Behind the scenes, the build only uses *shortcodes* that will be processed by *Image Intense*. However, when such a *shortcode* is processed, the `src` attribute is used without any sanitization and passed to `get_image_url_by_size`:

```
function shortcode_callback( $atts, $content = null, $function_name ) {
 $module_id                 = $this->shortcode_atts['module_id'];
 $module_class              = $this->shortcode_atts['module_class'];
 $src                       = $this->shortcode_atts['src'];
...
 $src_available = '';
 if ( 'full' != $size ) {
     $src_available = $this->get_image_url_by_size( $src, $size);

     if ( '' != $src_available ) {
         // Found a match on media size
         $src = $src_available;
     }
 }
```

The `get_image_url_by_size` function uses directly the first parameter in the SQL statement:

```
private function get_image_url_by_size( $image_url, $size) {
...
 $sql     = "SELECT ID FROM `" . $table . "` WHERE guid='" . $image_url . "';";

 $attachment  = $wpdb->get_col( $sql );
   // If the media size was not found, let's try using the full attachment URL to find the
right GUID
   if ( !isset( $attachment ) || $attachment == false ) {
       $site_url = site_url();
       $sql       = "SELECT ID FROM `" . $table . "` WHERE guid='" . $site_url . $image_url
. "';";

       $attachment  = $wpdb->get_col( $sql );
   }
```

Using a `src` attribute value aiming to alter the original query, it is possible to trigger the SQL injection when a user displays the post or page. For instance, the following payload will call the MySQL `SLEEP` method for 10 seconds:

```
[et_pb_section bb_built="1"][et_pb_row][et_pb_column type="4_4"][et_pb_image_n10s
_builder_version="3.0.82" src="test' OR SLEEP(10) -- " size="azaz"
/][/et_pb_column][/et_pb_row][/et_pb_section]
```

It should be noted that the `size` attribute must not be equal to `full` to reach the vulnerable code path.

Impact

A successful exploitation could allow an attacker authenticated with a role allowing him to create posts or pages to extract records from the database and, depending on the DBMS' permission scheme, access other databases or the local filesystem.