Wordpress 5.2.3 未授权页面查看漏洞（CVE-2019-17671）分析

熊本熊本熊 / 2019-11-18 09:22:38 / 浏览数 4519 安全技术 漏洞分析 顶(0) 踩(0)

写在前面

WordPress在5.2.3以及之前版本，存在着一处未授权页面查看漏洞，攻击者可以在未授权的情况下，查看所有私密页面或是已经删除至回收站的页面

这个漏洞，请注意我的描述用词，是私密页面查看，而非私密文章查看，这点很关键，在wordpress中，POST指的是文章，Page指的是页面，这两个不是同一个概念，如下

页面：

用户可以单独建立一个固定页面，可以作为留言板，或者通知的单页面，发布之后是固定的网址。页面并不能被分类、亦不能拥有标签，但是它们可以有层级关系。可将页面

文章：

文章可以通过标签实现相关文章的链接，可以放评论和评论框来实现与用户的互动，页面没有。文章有栏目可以归档，还有标签，页面没有。编辑文章时可选不同的形式，页

利用这个漏洞，攻击者并不能查看未发布的文章，只能有一定几率查看私密或已以至回收站的页面

漏洞分析

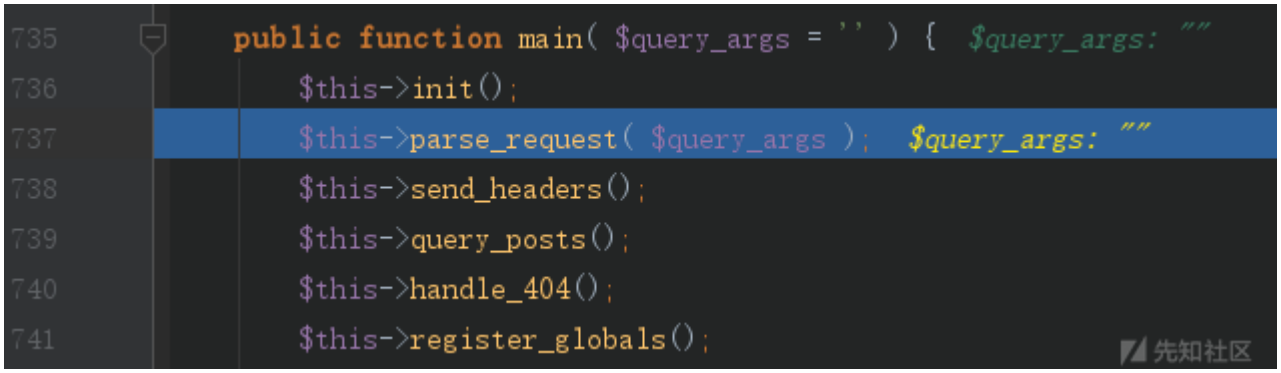
我们将自上而下，从wordpress入口到漏洞触发点，来分析该漏洞

首先来看位于\wp-includes\class-wp.php 中的WP类，该类为WordPress环境设置类

在WP类中，存在\$public_query_vars数组，该数组用来定义公共查询变量，如下图

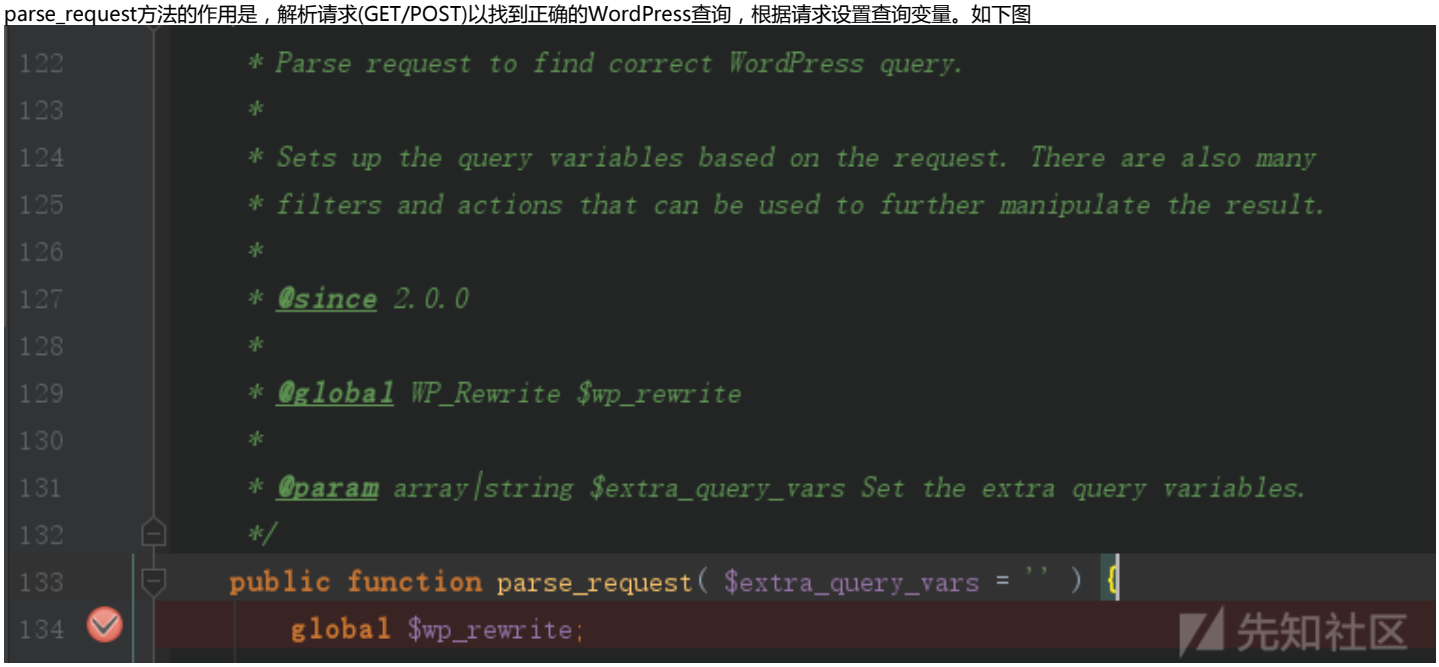
```
class WP {
    /**
     * Public query variables.
     *
     * Long list of public query variables.
     *
     * @since 2.0.0
     * @var string[]
     */
    public $public_query_vars = array( 'm', 'p', 'posts', 'w', 'cat', 'withcomments', 'withoutcomments', 's', 'search', 'exact', 'sentence', 'calendar', 'page',
'paged', 'more', 'tb', 'pb', 'author', 'order', 'orderby', 'year', 'monthnum', 'day', 'hour', 'minute', 'second', 'name', 'category_name', 'tag', 'feed',
'author_name', 'static', 'pagename', 'page_id', 'error', 'attachment', 'attachment_id', 'subpost', 'subpost_id', 'preview', 'robots', 'taxonomy', 'term', 'cpage',
'post_type', 'embed' );  public_query_vars: [48]
```

在WP类中，存在main方法，该方法用来设置WordPress环境所需的所有变量，如下图

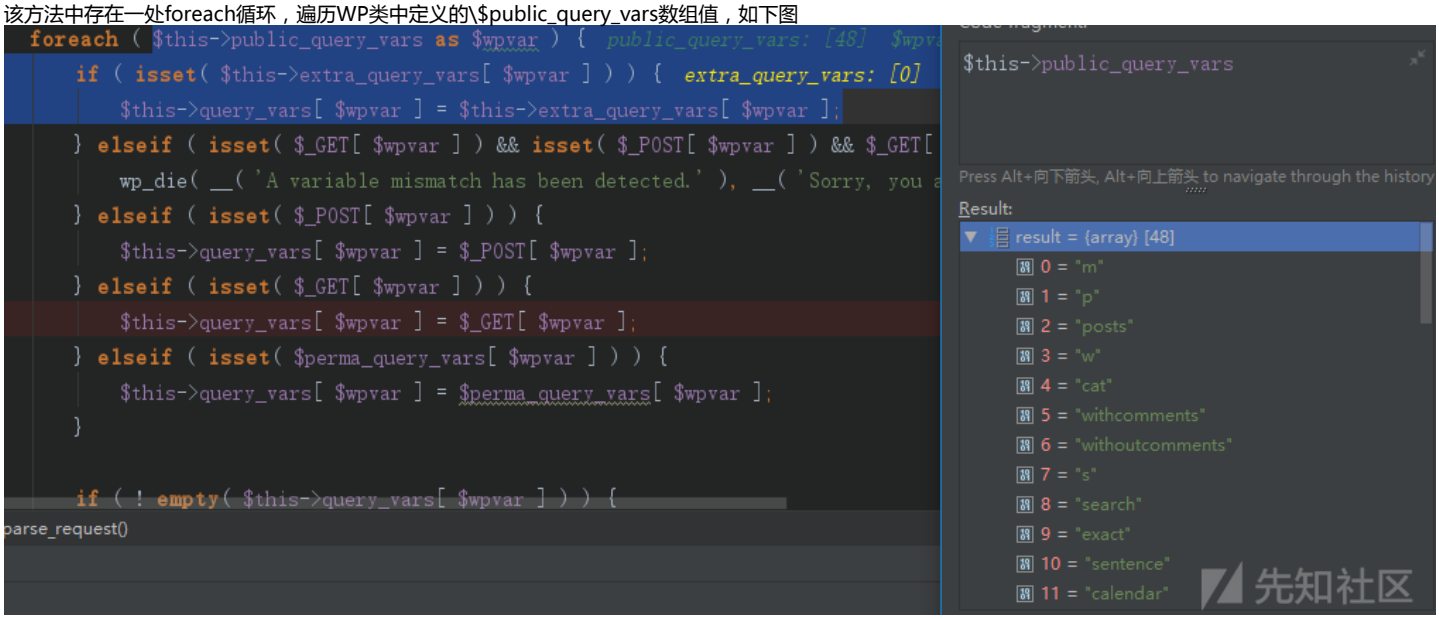


Wordpress启动时，会调用WP类中的main方法，进行环境遍历赋值，位于上图main方法737行处，可见调用parse_request方法

parse_request方法的作用是，解析请求(GET/POST)以找到正确的WordPress查询，根据请求设置查询变量。如下图



该方法中存在一处foreach循环，遍历WP类中定义的\$public_query_vars数组值，如下图



该处循环的作用，是寻找\$public_query_vars数组中的值，是否存在于GET/POST请求的参数中，如过在请求的参数中找到，就将其参数键与值赋值到WP环境变量中去，\

例如有如下payload请求

http://127.0.0.1/wordpress/?static=0&order=asc&kumamon=test

\$public_query_vars数组中存在"order"与" static"

```
public $public_query_vars = array( 'm', 'p', 'posts', 'w', 'cat', 'withcomments', 'withoutcomments', 's', 'search', 'exact', 'sentence', 'calendar', 'page',
'paged', 'more', 'tb', 'pb', 'author', 'order', 'orderby', 'year', 'monthnum', 'day', 'hour', 'minute', 'second', 'name', 'category_name', 'tag', 'feed',
'author_name', 'static', 'pagename', 'page_id', 'error', 'attachment', 'attachment_id', 'subpost', 'subpost_id', 'preview', 'robots', 'taxonomy', 'term', 'cpage',
'post_type', 'embed' );  public_query_vars: [48]
```


而GET请求的参数中也存在这两个参数，于是程序会将GET中order与static的值赋值给\$this->query_vars['order']与\$this->query_vars['static]，如下图

```
} elseif ( isset( $_GET[ $wpvar ] ) ) {
    $this->query_vars[ $wpvar ] = $_GET[ $wpvar ];  $_GET: {static => "0", order => "asc", kumamon => "test"}[3]  $wpvar: "order"  query_vars: [0]
} elseif ( isset( $p    ery_vars[ $wpvar ] ) ) {
```


\$public_query_vars数组中并无"kumamon"，因此GET请求中的kumamon变量不做处理

Wordpress的环境变量机制了解完毕后，接下来看下漏洞触发点

首先来看下

\wp-includes\class-wp-query.php中的parse_query方法

该方法也是在wordpress启动时入口处被一系列的调用加载进来的,执行顺序位于parse_request方法之后，也就是环境变量赋值之后

```
731    public function parse_query( $query = '' ) {  $query: ""
732        if ( ! empty( $query ) ) {
733            $this->init();
734            $this->query = $this->query_vars = wp_parse_args( $query );  $query: ""
735        } elseif ( ! isset( $this->query ) ) {
736            $this->query = $this->query_vars;  query: [2]
737        }
738
739        $this->query_vars          = $this->fill_query_vars( $this->query_vars );
740        $qv                        = &$this->query_vars;  query_vars: [53]  $qv: {order => "asc", stat
741        $this->query_vars_changed = true;  query_vars_changed: true
```

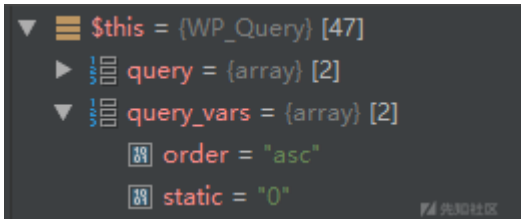

我们重点关注下\$qv变量，如下图红框处

```
730    public function parse_query( $query = '' ) {  $query: ""
731        if ( ! empty( $query ) ) {
732            $this->init();
733            $this->query = $this->query_vars = wp_parse_args( $query );
734        } elseif ( ! isset( $this->query ) ) {
735            $this->query = $this->query_vars;
736        }
737
738        $this->query_vars          = $this->fill_query_vars( $this->query_vars );
739        $qv                        = &$this->query_vars;
740        $this->query_vars_changed = true;
741
```


该变量为\$this->query_vars引用而来的，而\$this->query_vars则是\$this->query_vars经过fill_query_vars方法处理之后的值

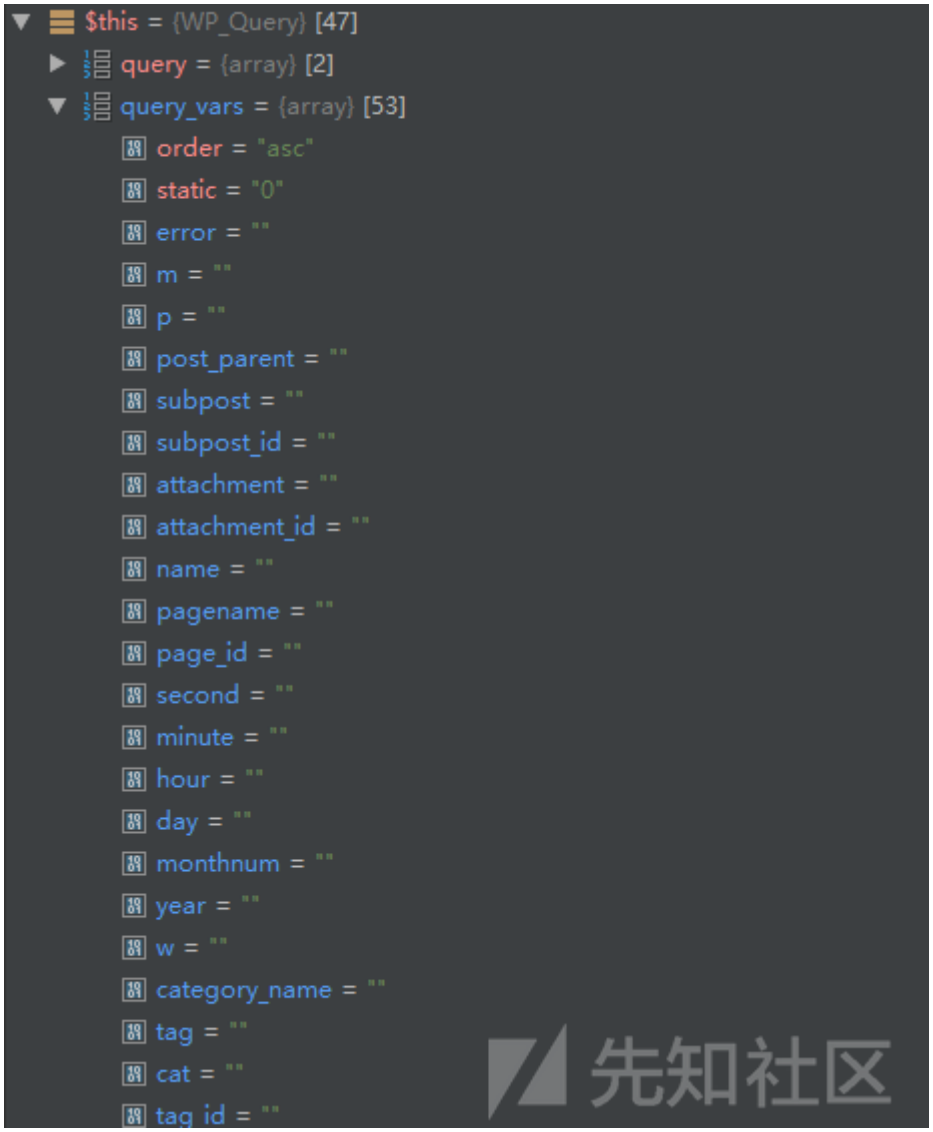
当我们的请求为http://127.0.0.1/wordpress/?static=0&order=asc

\$this->query_vars值如下，该值由parse_request方法得来



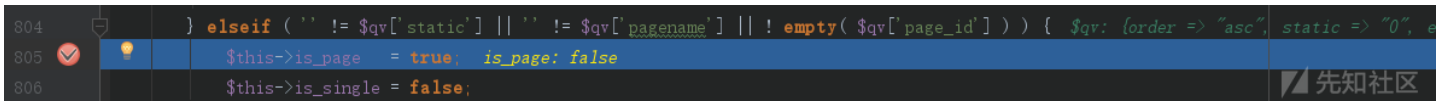
而fill_query_vars方法，是将其他并未从请求中传递与赋值的环境变量用空值赋值

我们这里仅仅通过请求赋值了static与order两个环境变量，因此\$this->query_vars值如下



\$qv为\$this->query_vars引用，因此其中值与上图一致

接下来，位于805行处，有如下if-else条件



由于我们通过GET请求传入static变量，已经将\$qv['static']赋值为0，因此可以进入上图条件分支，使得\$this->is_page=true,
\$this->is_single=false

还记得之前所说，Page指的是页面吗？因此上文的这个if条件，是为了通过检查请求中是否有'static'、'pagename'、'page_id'值，来判断是否要进行页面(Page)处理，如果

继续向下看，位于3043行处，存在如下if条件分支

```
3042        // Check post status to determine if post should be displayed.
3043        if ( ! empty( $this->posts ) && ( $this->is_single || $this->is_page ) ) {  is_page: true  is_single: false
3044            $status = get_post_status( $this->posts[0] );  posts: [5]
3045            if ( 'attachment' === $this->posts[0]->post_type && 0 === (int) $this->posts[0]->post_parent ) {
3046                $this->is_page       = false;
3047                $this->is_single     = true;
3048                $this->is_attachment = true;
3049            }
3050            $post_status_obj = get_post_status_object( $status );
3051
```


此时\$this->is_page=true, \$this->is_single=false，所以if中(
\$this->is_single || \$this->is_page)处的值为true。

只要使得\$this->posts不为空，则可以进入此处if分支

\$this->posts值为如下sql语句的查询值

```
request = "SELECT  wp_posts.* FROM wp_posts  WHERE 1=1  AND wp_posts.post_type = 'page'  ORDER BY wp_posts.post_date ASC"
```


这里解释下，为什么sql语句WHERE中wp_posts.post_type = 'page' 且ORDER BY
wp_posts.post_date ASC

首先看下wp_posts.post_type = 'page'

由于上文设置了\$this->is_page=true，因此进入如下条件分支

```
} elseif ( $this->is_page ) {
    $where           .= " AND {$wpdb->posts}.post_type = 'page'";
    $post_type_object = get_post_type_object( post_type: 'page' );
```


此处设置了查询条件为wp_posts.post_type =
'page'。显然，\$this->is_page=true，是要在库中查找page类型的发布

ORDER BY wp_posts.post_date
ASC的原因是由于我们GET请求中传入的order参数为ASC，其order为环境变量，在这里被直接拿来拼接sql语句了

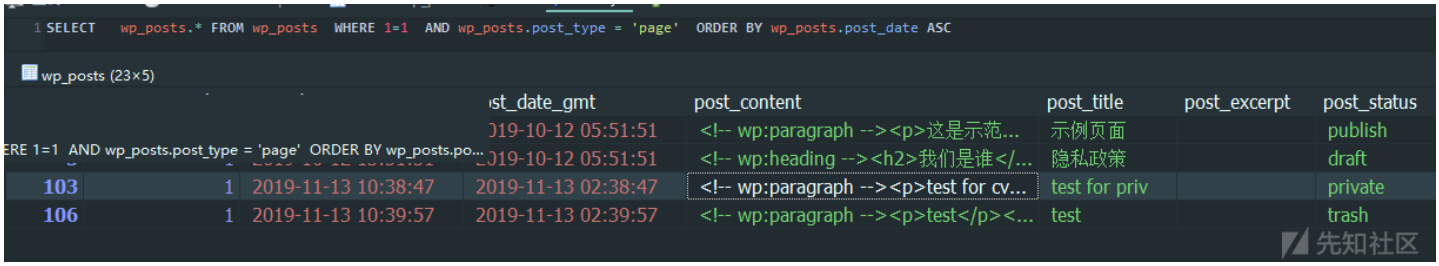
在了解了wordpress此时的查询语句，我们对照后台数据库中wp_posts表的内容分析下

wp_posts表中存储了wordpress所有发布的内容，并同过post_type对其进行类型区分

t	guid	menu_order	post_type	post_mime_type	c
0	http://127.0.0.1/wordpress/?p=1	0	post		
0	http://127.0.0.1/wordpress/?page_id=2	0	page		
0	http://127.0.0.1/wordpress/?page_id=3	0	page		
0	http://127.0.0.1/wordpress/?page_id...	0	page		
6	http://127.0.0.1/wordpress/2019/11/...	0	revision		
3	http://127.0.0.1/wordpress/2019/11/...	0	revision		
0	http://127.0.0.1/wordpress/?p=113	0	post		

Post_type 为post的，代表其为文章(POST)；而page，代表这是一个页面

因此我们的\$this->is_page=true，使得程序从该中查询所有page类型的发布内容，也就是说，把所有的页面都提取出来了



```
1 SELECT   wp_posts.* FROM wp_posts  WHERE 1=1  AND wp_posts.post_type = 'page'  ORDER BY wp_posts.post_date ASC
```

wp_posts (23×5)

			st_date_gmt	post_content	post_title	post_excerpt	post_status
ERE 1=1 AND wp_posts.post_type = 'page' ORDER BY wp_posts.po...			019-10-12 05:51:51	<!-- wp:paragraph --> 这是示范...	示例页面		publish
			019-10-12 05:51:51	<!-- wp:heading --><h2>我们是谁</...	隐私政策		draft
103	1	2019-11-13 10:38:47	2019-11-13 02:38:47	<!-- wp:paragraph --> test for cv...	test for priv		private
106	1	2019-11-13 10:39:57	2019-11-13 02:39:57	<!-- wp:paragraph --> test <...	test		trash

注意这里：这条查询，把所有的page都查询出来，不管其状态是发布(publish)或是private(私密)甚至是回收站(trash)。并且通过发布时间升序排列，至于需要设置升序排列

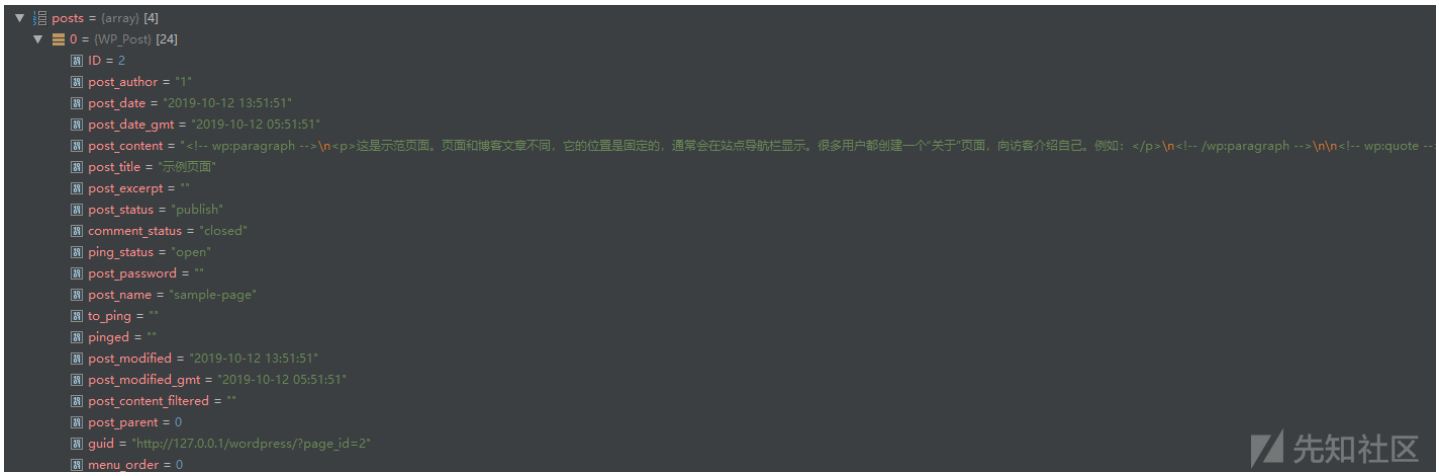
继续回到漏洞点

```
3042        // Check post status to determine if post should be displayed.
3043        if ( ! empty( $this->posts ) && ( $this->is_single || $this->is_page ) ) {
3044            $status = get_post_status( $this->posts[0] );
3045            if ( 'attachment' === $this->posts[0]->post_type && 0 === (int) $this->posts[0]->post_parent ) {  posts: [4]
3046                $this->is_page       = false;  is_page: true
3047                $this->is_single     = true;  is_single: false
3048                $this->is_attachment = true;  is_attachment: false
3049            }
3050            $post_status_obj = get_post_status_object( $status );  $post_status_obj: {label => "已发布", label_count => [6], 
3051
3052            // If the post_status was specifically requested, let it pass through.
3053            if ( ! $post_status_obj->public && ! in_array( $status, $q_status ) ) {  $q_status: [0]  $status: "publish"
3054
```


此时我们已经搞清楚\$this->posts是什么了，如上文所说，\$this->posts存放着从wp_posts表中取出的所有页面(Page)，并且通过时间顺序升序排列

```
▼ posts = {array} [4]
  ▶ 0 = {WP_Post} [24]
  ▶ 1 = {WP_Post} [24]
  ▶ 2 = {WP_Post} [24]
  ▶ 3 = {WP_Post} [24]
```


这时候\$this->post[0]即为存放在数据库中最早发布的那篇页面(Page)，由于我的示范页面没有删，所以这里的\$this->post[0]就是那篇示范页面



```
▼ posts = {array} [4]
  ▼ 0 = {WP_Post} [24]
      ID = 2
      post_author = "1"
      post_date = "2019-10-12 13:51:51"
      post_date_gmt = "2019-10-12 05:51:51"
      post_content = "<!-- wp:paragraph -->\n<p>这是示范页面。页面和博客文章不同，它的位置是固定的，通常会在站点导航栏显示。很多用户都创建一个"关于"页面，向访客介绍自己。例如：</p>\n<!-- /wp:paragraph -->\n\n<!-- wp:quote --
      post_title = "示例页面"
      post_excerpt = ""
      post_status = "publish"
      comment_status = "closed"
      ping_status = "open"
      post_password = ""
      post_name = "sample-page"
      to_ping = ""
      pinged = ""
      post_modified = "2019-10-12 13:51:51"
      post_modified_gmt = "2019-10-12 05:51:51"
      post_content_filtered = ""
      post_parent = 0
      guid = "http://127.0.0.1/wordpress/?page_id=2"
      menu_order = 0
```


而\$this->post[2]就是我们的私密的页面，如下图，可见post_status为private

```
posts = {array} [4]
  0 = {WP_Post} [24]
  1 = {WP_Post} [24]
  2 = {WP_Post} [24]
    ID = 103
    post_author = "1"
    post_date = "2019-11-13 10:38:47"
    post_date_gmt = "2019-11-13 02:38:47"
    post_content = "<!-- wp:paragraph -->\n<p>test for cve-2019-17671</p>\n<!-- /wp:paragraph -->"
    post_title = "test for priv"
    post_excerpt = ""
    post_status = "private"
    comment_status = "closed"
    ping_status = "closed"
    post_password = ""
    post_name = "test-for-priv"
    to_ping = ""
    pinged = ""
    post_modified = "2019-11-13 10:40:58"
    post_modified_gmt = "2019-11-13 02:40:58"
    post_content_filtered = ""
    post_parent = 0
    guid = "http://127.0.0.1/wordpress/?page_id=103"
    menu_order = 0
```

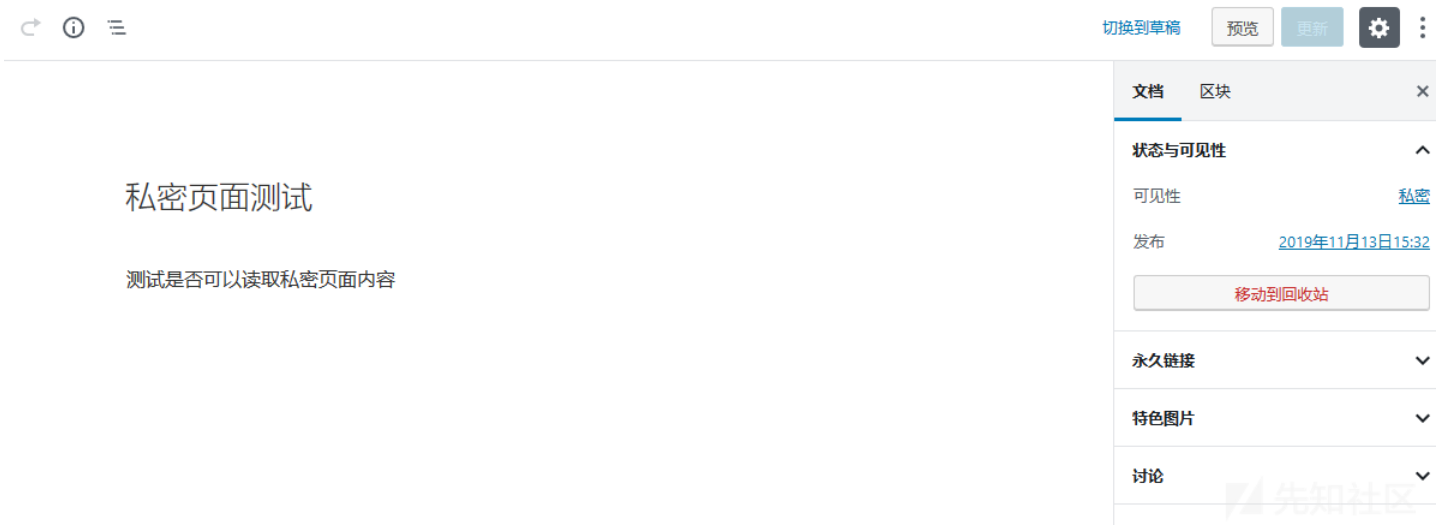

为什么要用ASC将我们最早发布的页面放在\$this->post[0]位置呢？原因如下

```
3042        // Check post status to determine if post should be displayed.
3043        if ( ! empty( $this->posts ) && ( $this->is_single || $this->is_page ) ) {
3044            $status = get_post_status( $this->posts[0] );
3045            if ( 'attachment' === $this->posts[0]->post_type && 0 === (int) $this->posts[0]->post_parent ) {  posts: [4]
3046                $this->is_page       = false;  is_page: true
3047                $this->is_single     = true;  is_single: false
3048                $this->is_attachment = true;  is_attachment: false
3049            }
3050            $post_status_obj = get_post_status_object( $status );  $post_status_obj: {label => "已发布", label_count => [6], exclude_from_sear
3051
3052            // If the post_status was specifically requested, let it pass through.
3053            if ( ! $post_status_obj->public && ! in_array( $status, $q_status ) ) {  $q_status: [0]  $status: "publish"
3054
3055                if ( ! is_user_logged_in() ) {
3056                    // User must be logged in to view unpublished posts.
3057                    $this->posts = array();
```


程序会检查\$this->post[0]的发布状态，如上图前两个红框，并判断\$this->post[0]的发布状态是否是public，若\$this->post[0]的发布状态不为public，则接下来进行登

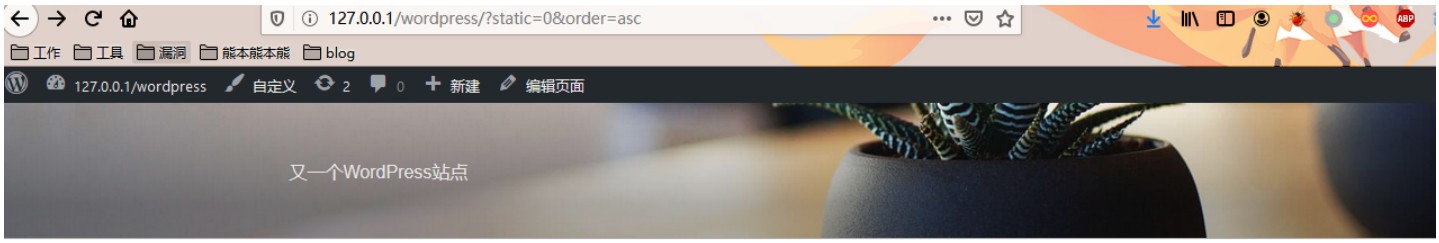
因此通过ASC升序排列，尽可能的把最早发布的页面排在\$this->post[0]，这个\$this->post[0]大概率是wordpress示例页面或者是网站自行添加的说明页面，其状态大概

我们新建一个私密测试页面，如下图



建立一个回收站页面测试，并把它丢到回收站，如下图

通过我们的payload，可见私密以及回收站的Page都可被查看



示例页面
编辑

回收站页面测试
编辑

这是示范页面。页面和博客文章不同，它的位置是固定的，通常会在站点导航栏显示。很多用户都创建一个"关于"页面，向访客介绍自己。例如：

欢迎！我白天是个邮递员，晚上就是个有抱负的演员。这是我的网站。我住在天朝的帝都，有条叫做Jack的狗。

……或这个：

XYZ Doohickey公司成立于1971年，自从建立以来，我们一直向社会贡献着优秀doohickies。我们的公司总部位于天朝魔都，有着超过两千名员工，对魔都政府税收有着巨大贡献。

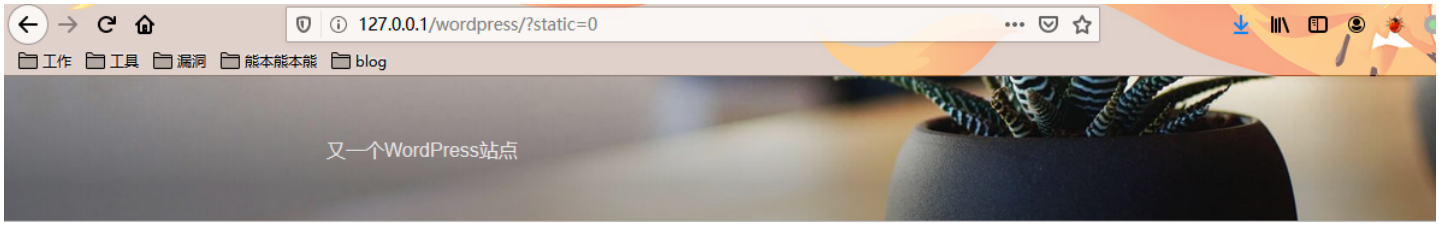
而您，作为一个WordPress用户，我们建议您访问控制板，删除本页面，然后添加您自己的页面。祝您使用愉快！

私密：私密页面测试
编辑

测试回收站里的页面是否可以显示

测试是否可以读取私密页面内容

若我们没有设置order=asc，则\$this->post[0]为我最后丢掉回收站里的Page(默认是用发布时间降序排序，而丢到回收站的那篇是我测试时最后新建的),它的状态是trash而



有点尴尬诶！该页无法显示。

这儿似乎什么都没有，试试搜索?

搜索...

漏洞修复

wp-includes/class-wp-query.php

```
@@ -538,7 +538,6 @@ public function fill_query_vars( $array ) {
                        'attachment',
                        'attachment_id',
                        'name',
-                       'static',
                        'pagename',
                        'page_id',
                        'second',
@@ -802,7 +801,7 @@ public function parse_query( $query = '' ) {
                        // If year, month, day, hour, minute, and second are set, a single
                        // post is being queried.
                        $this->is_single = true;
-               } elseif ( '' != $qv['static'] || '' != $qv['pagename'] || ! empty( $qv['page_id'] ) ) {
+               } elseif ( '' != $qv['pagename'] || ! empty( $qv['page_id'] ) ) {
                        $this->is_page   = true;
                        $this->is_single = false;
                } else {
```

wp-includes/class-wp.php

```
@@ -14,7 +14,7 @@ class WP {
         * @since 2.0.0
         * @var string[]
         */
-       public $public_query_vars = array( 'm', 'p', 'posts', 'w', 'cat', 'withcomments', 'withoutcomments', 's', 'search', 'exact', 'sentence', 'calendar', 'page', 'paged', 'more', 'tb', 'pb', 'author', 'order', 'orderby', 'year', 'monthnum', 'day', 'hour', 'minute', 'second', 'name', 'category_name', 'tag', 'feed', 'author_name', 'static', 'pagename', 'page_id', 'error', 'attachment', 'attachment_id', 'subpost', 'subpost_id', 'preview', 'robots', 'taxonomy', 'term', 'cpage', 'post_type', 'embed' );
```

漏洞修复其实很容易理解：

开发者把\$public_query_vars数组中的static给删了，这样就算请求中传入static的值，也会被忽略，记得上文http://127.0.0.1/wordpress/?static=0&order=asc&kuman
中的kumamon参数吗？

其次，开发者把''
!=\$qv['static']这个条件也删除了，这样的话，只能通过pagename或者page_id查询单条page了，然而单条page在显示时，是需要验证其状态的，非public的单条page是

点击收藏 | 0 关注 | 1

上一篇：社会工程学攻击-钓鱼 下一篇：.Cobain勒索病毒分析

1. 0 条回复
   - 动动手指，沙发就是你的了！

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板