[红日安全]代码审计Day16 - 深入理解$_REQUESTS数组

红日安全 / 2018-10-31 09:54:00 / 浏览数 3806 安全技术 漏洞分析 顶(1) 踩(0)

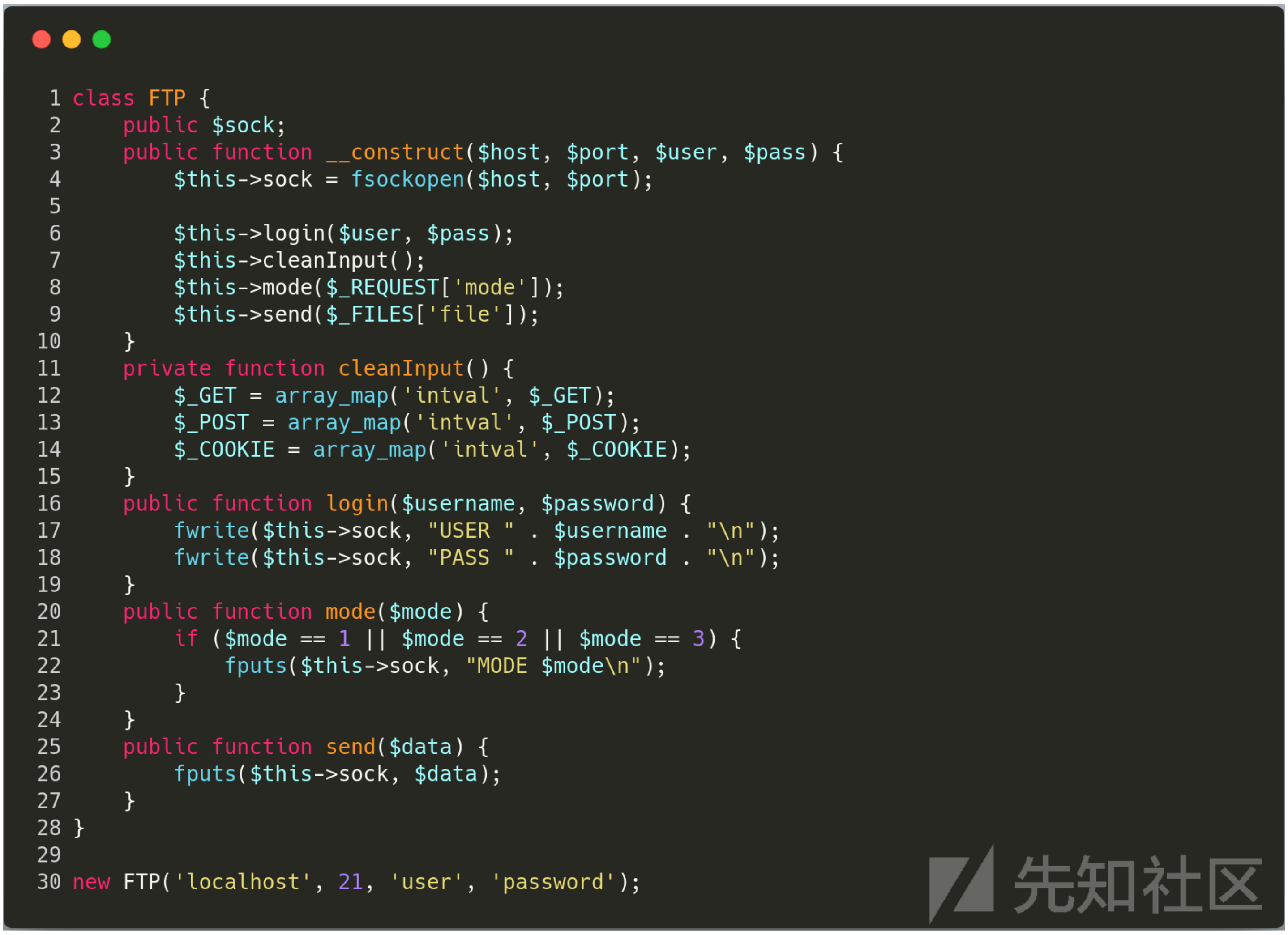
本文由红日安全成员： 七月火 编写，如有不当，还望斧正。

前言

大家好，我们是红日安全-代码审计小组。最近我们小组正在做一个PHP代码审计的项目，供大家学习交流，我们给这个项目起了一个名字叫 PHP-Audit-Labs 。现在大家所看到的系列文章，属于项目 第一阶段 的内容，本阶段的内容题目均来自 PHP SECURITY CALENDAR 2017 。对于每一道题目，我们均给出对应的分析，并结合实际CMS进行解说。在文章的最后，我们还会留一道CTF题目，供大家练习，希望大家喜欢。下面是 第16篇 代码审计文章：

Day 16 - Poem

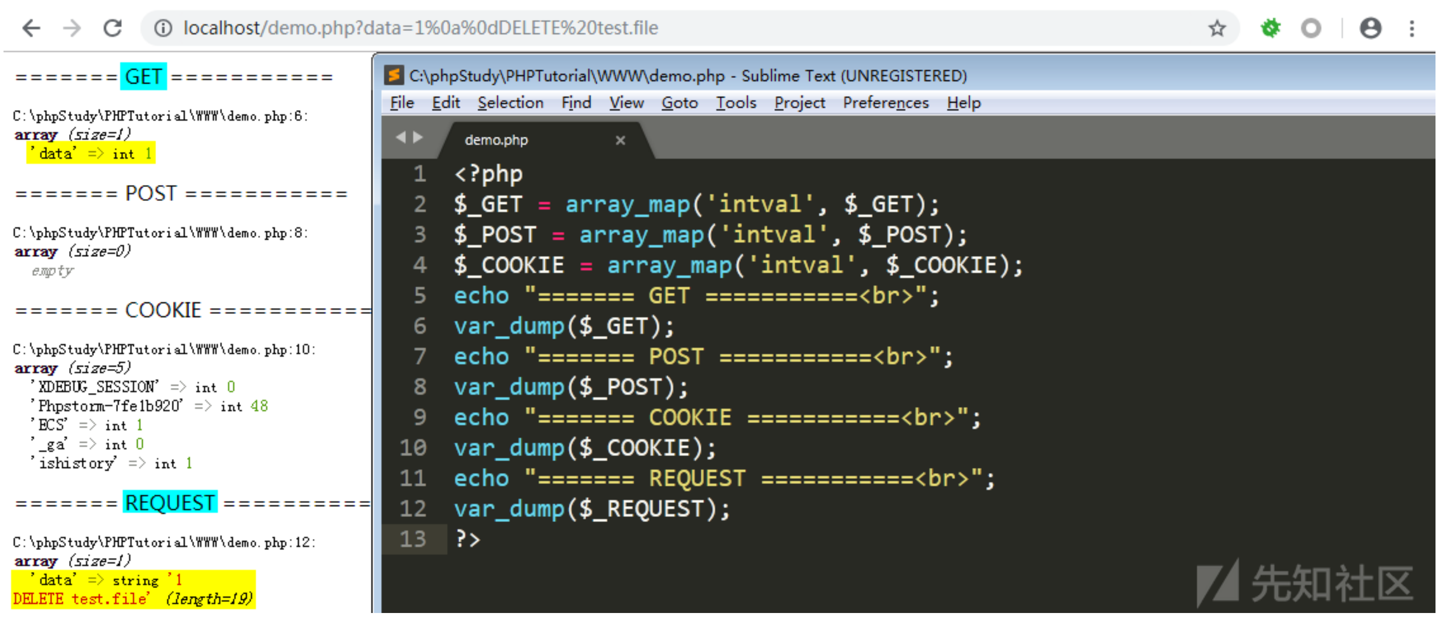
题目叫做诗，代码如下：



```php
 1 class FTP {
 2     public $sock;
 3     public function __construct($host, $port, $user, $pass) {
 4         $this->sock = fsockopen($host, $port);
 5 
 6         $this->login($user, $pass);
 7         $this->cleanInput();
 8         $this->mode($_REQUEST['mode']);
 9         $this->send($_FILES['file']);
10     }
11     private function cleanInput() {
12         $_GET = array_map('intval', $_GET);
13         $_POST = array_map('intval', $_POST);
14         $_COOKIE = array_map('intval', $_COOKIE);
15     }
16     public function login($username, $password) {
17         fwrite($this->sock, "USER " . $username . "\n");
18         fwrite($this->sock, "PASS " . $password . "\n");
19     }
20     public function mode($mode) {
21         if ($mode == 1 || $mode == 2 || $mode == 3) {
22             fputs($this->sock, "MODE $mode\n");
23         }
24     }
25     public function send($data) {
26         fputs($this->sock, $data);
27     }
28 }
29 
30 new FTP('localhost', 21, 'user', 'password');
```

漏洞解析 ：

这道题目包含了两个漏洞，利用这两个漏洞，我们可以往FTP连接资源中注入恶意数据，执行FTP命令。首先看到 第7行 代码，可以发现程序使用 cleanInput 方法过滤 GET 、 POST 、 COOKIE 数据，将他们强制转成整型数据。然而在 第8行 处，却传入了一个从 REQUEST 方式获取的 mode 变量。我们都知道超全局数组 $_REQUEST 中的数据，是 $_GET 、 $_POST 、 $_COOKIE 的合集，而且数据是复制过去的，并不是引用。我们先来看一个例子，来验证这一观点：


localhost/demo.php?data=1%0a%0dDELETE%20test.file
C:\phpStudy\PHPTutorial\WWW\demo.php - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
demo.php

```
======= GET ===========

C:\phpStudy\PHPTutorial\WWW\demo.php:6:
array (size=1)
  'data' => int 1

======= POST ===========

C:\phpStudy\PHPTutorial\WWW\demo.php:8:
array (size=0)
  empty

======= COOKIE ===========

C:\phpStudy\PHPTutorial\WWW\demo.php:10:
array (size=5)
  'XDEBUG_SESSION' => int 0
  'Phpstorm-7fe1b920' => int 48
  'ECS' => int 1
  '_ga' => int 0
  'ishistory' => int 1

======= REQUEST ===========

C:\phpStudy\PHPTutorial\WWW\demo.php:12:
array (size=1)
  'data' => string '1
DELETE test.file' (length=19)
```

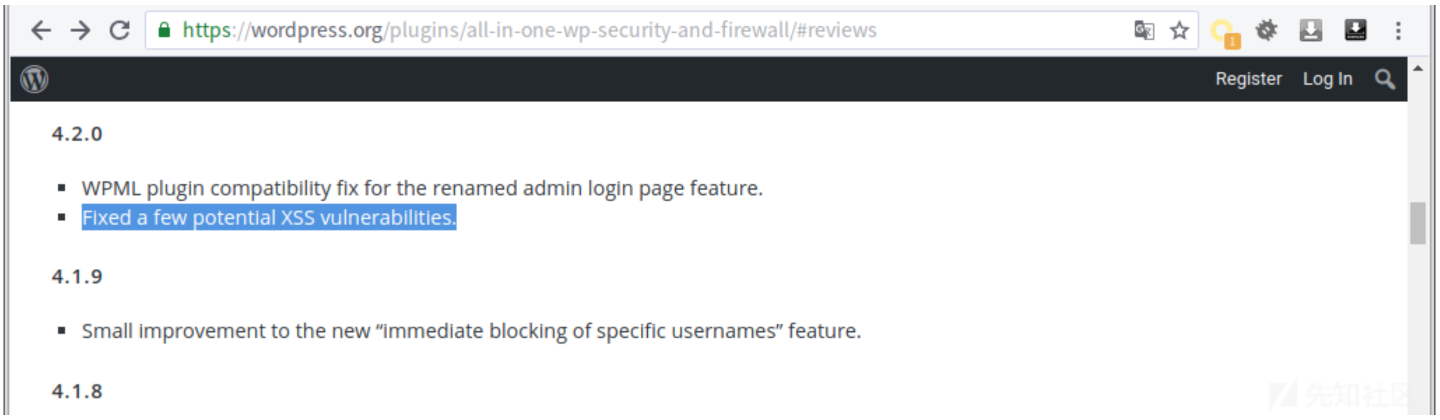
```php
<?php
$_GET = array_map('intval', $_GET);
$_POST = array_map('intval', $_POST);
$_COOKIE = array_map('intval', $_COOKIE);
echo "======= GET ===========<br>";
var_dump($_GET);
echo "======= POST ===========<br>";
var_dump($_POST);
echo "======= COOKIE ===========<br>";
var_dump($_COOKIE);
echo "======= REQUEST ===========<br>";
var_dump($_REQUEST);
?>
```

可以发现 REQUEST 数据丝毫不受过滤函数的影响。回到本例题，例题中的程序过滤函数只对 GET 、 POST 、 COOKIE 数据进行操作，最后拿来用的却是 REQUEST 数据，这显然会存在安全隐患。想了解更多 $_REQUEST 信息，大家自己上官网学习。第二个漏洞的话，在代码 第21行 ，这里用了 == 弱比较。关于这个问题，我们在前面的文章中讲的也很细致了，大家可以参考：[红日安全]PHP-Audit-Labs题解之Day1-4 （ Day4 ）。

至于本次案例的攻击payload，可以使用： ?mode=1%0a%0dDELETE%20test.file ，这个即可达到删除FTP服务器文件的效果。

实例分析

本次实例分析，我们分析的是 WordPress 的 All In One WP Security & Firewall 插件。该插件在 4.1.4 - 4.1.9 版本中存在反射型XSS漏洞，漏洞原因和本次案例中的漏洞成因一致，官方也在 4.2.0 版本中修复了该漏洞。本次，我们将以 4.1.4 版本插件作为案例进行讲解。


https://wordpress.org/plugins/all-in-one-wp-security-and-firewall/#reviews
Register
Log In

4.2.0

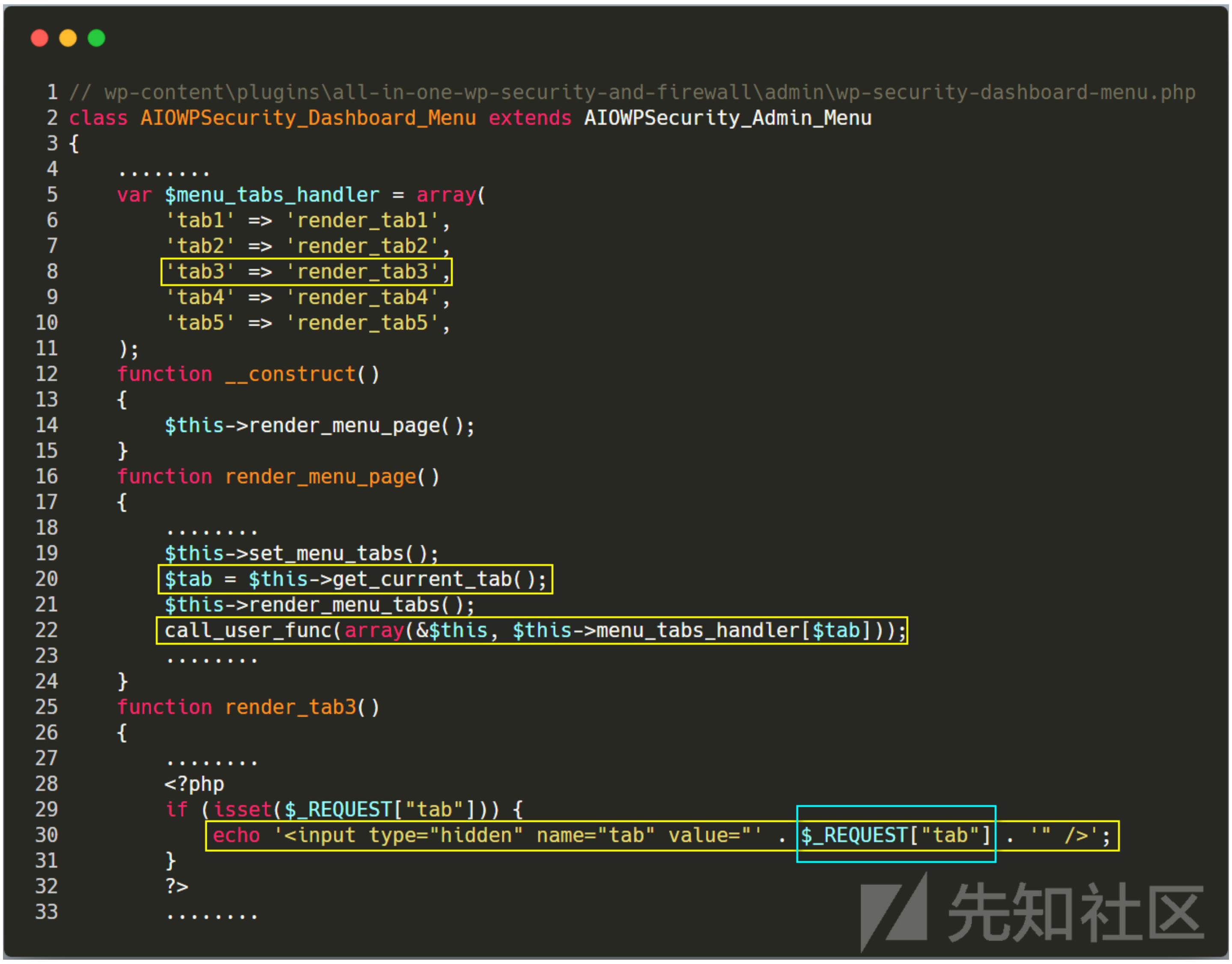
- WPML plugin compatibility fix for the renamed admin login page feature.
- Fixed a few potential XSS vulnerabilities.

4.1.9

- Small improvement to the new "immediate blocking of specific usernames" feature.

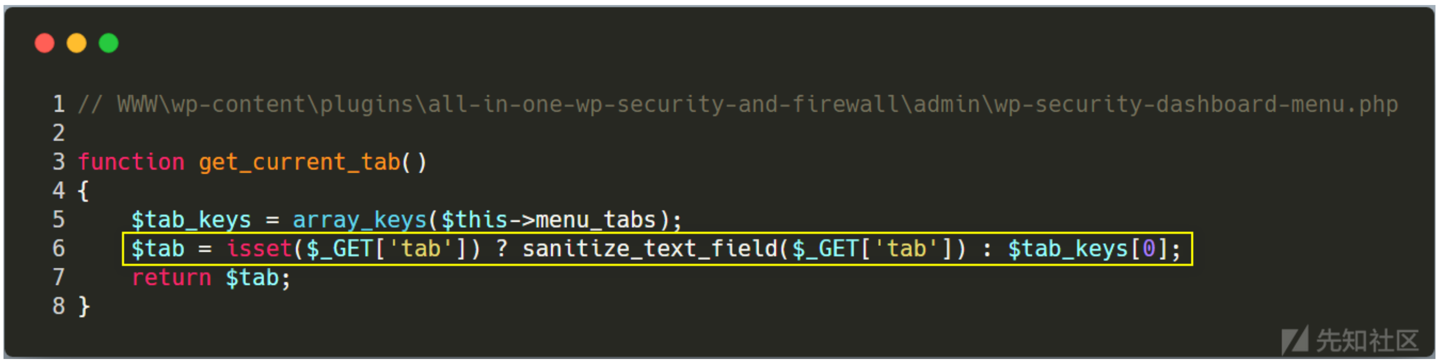
4.1.8

将下载下来的插件zip包，通过后台插件管理上传压缩包安装即可。本次发生问题的文件在于 wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-dashboard-menu.php ，为了方便大家理解，我将问题代码抽取出来，简化如下：



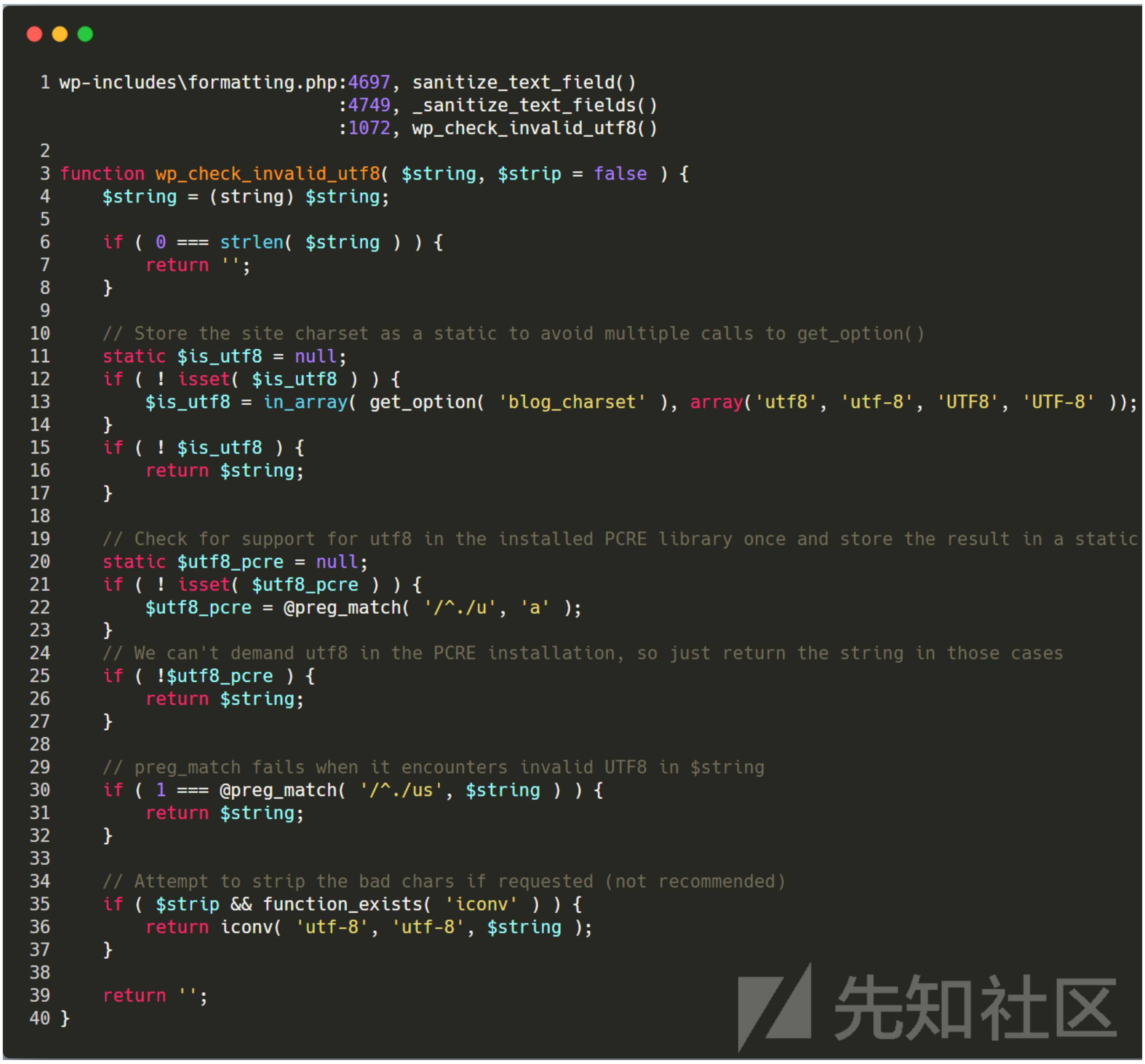
```
 1 // wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-dashboard-menu.php
 2 class AIOWPSecurity_Dashboard_Menu extends AIOWPSecurity_Admin_Menu
 3 {
 4     ........
 5     var $menu_tabs_handler = array(
 6         'tab1' => 'render_tab1',
 7         'tab2' => 'render_tab2',
 8         'tab3' => 'render_tab3',
 9         'tab4' => 'render_tab4',
10         'tab5' => 'render_tab5',
11     );
12     function __construct()
13     {
14         $this->render_menu_page();
15     }
16     function render_menu_page()
17     {
18         ........
19         $this->set_menu_tabs();
20         $tab = $this->get_current_tab();
21         $this->render_menu_tabs();
22         call_user_func(array(&$this, $this->menu_tabs_handler[$tab]));
23         ........
24     }
25     function render_tab3()
26     {
27         ........
28         <?php
29         if (isset($_REQUEST["tab"])) {
30             echo '<input type="hidden" name="tab" value="' . $_REQUEST["tab"] . '" />';
31         }
32         ?>
33         ........
```

我们可以很清晰的看到，问题就出在 第25行 的 render_tab3 方法中，这里直接将 REQUEST 方式获取的 tab 变量拼接并输出。而实际上，在 第20行 已经获取了经过过滤处理的 $tab 变量。我们来看一下 get_current_tab 方法：



```
1 // WWW\wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-dashboard-menu.php
2
3 function get_current_tab()
4 {
5     $tab_keys = array_keys($this->menu_tabs);
6     $tab = isset($_GET['tab']) ? sanitize_text_field($_GET['tab']) : $tab_keys[0];
7     return $tab;
8 }
```

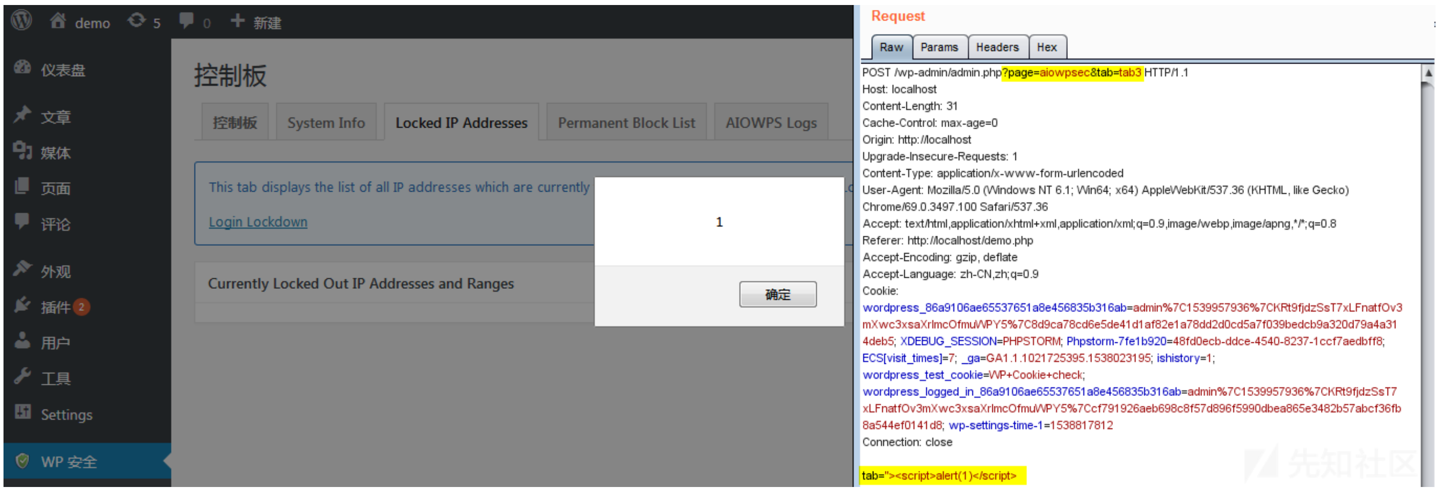
过滤函数的调用链如下图 第1行 ，接着 $tab 变量就会经过 wp_check_invalid_utf8 方法的检测。



```php
wp-includes\formatting.php:4697, sanitize_text_field()
                          :4749, _sanitize_text_fields()
                          :1072, wp_check_invalid_utf8()

function wp_check_invalid_utf8( $string, $strip = false ) {
    $string = (string) $string;

    if ( 0 === strlen( $string ) ) {
        return '';
    }

    // Store the site charset as a static to avoid multiple calls to get_option()
    static $is_utf8 = null;
    if ( ! isset( $is_utf8 ) ) {
        $is_utf8 = in_array( get_option( 'blog_charset' ), array('utf8', 'utf-8', 'UTF8', 'UTF-8' ));
    }
    if ( ! $is_utf8 ) {
        return $string;
    }

    // Check for support for utf8 in the installed PCRE library once and store the result in a static
    static $utf8_pcre = null;
    if ( ! isset( $utf8_pcre ) ) {
        $utf8_pcre = @preg_match( '/^./u', 'a' );
    }
    // We can't demand utf8 in the PCRE installation, so just return the string in those cases
    if ( !$utf8_pcre ) {
        return $string;
    }

    // preg_match fails when it encounters invalid UTF8 in $string
    if ( 1 === @preg_match( '/^./us', $string ) ) {
        return $string;
    }

    // Attempt to strip the bad chars if requested (not recommended)
    if ( $strip && function_exists( 'iconv' ) ) {
        return iconv( 'utf-8', 'utf-8', $string );
    }

    return '';
}
```

漏洞利用

下面我们来看看攻击 payload （向 http://website/wp-admin/admin.php?page=aiowpsec&tab=tab3 POST数据 `tab="><script>alert(1)</script>` ）：



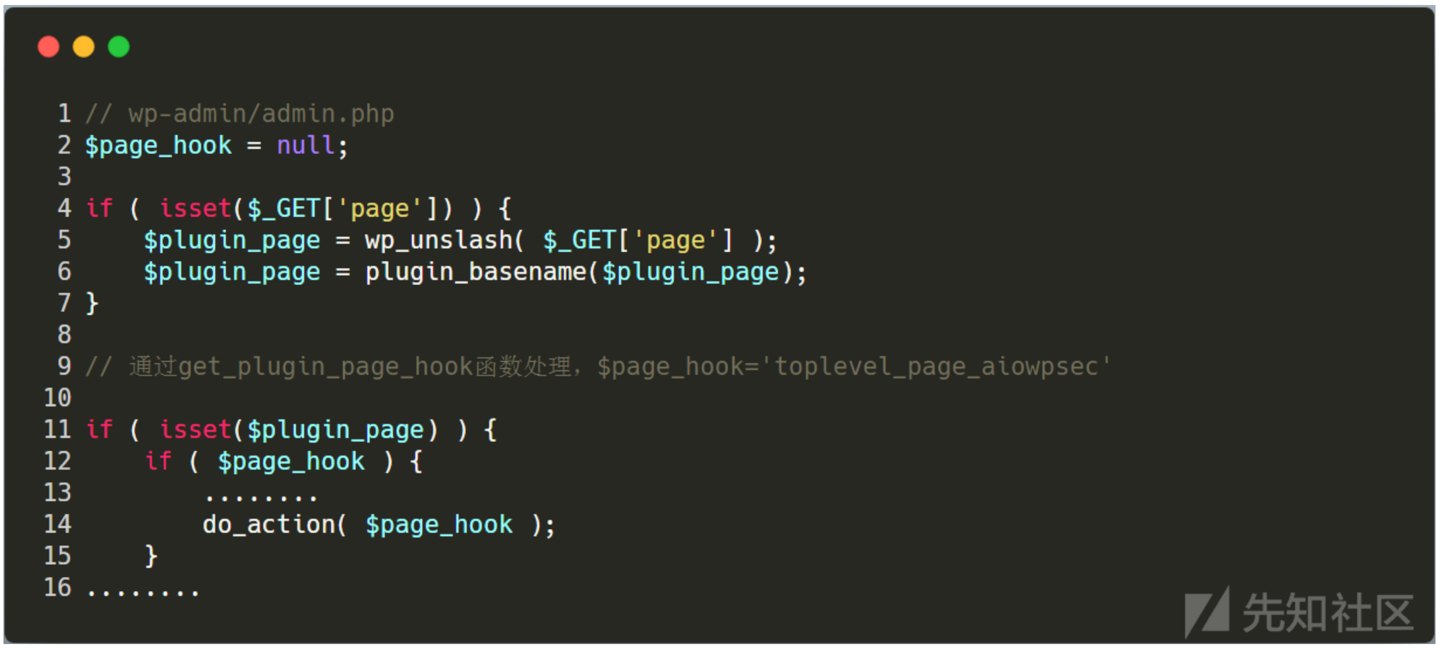
This tab displays the list of all IP addresses which are currently

Login Lockdown

Currently Locked Out IP Addresses and Ranges

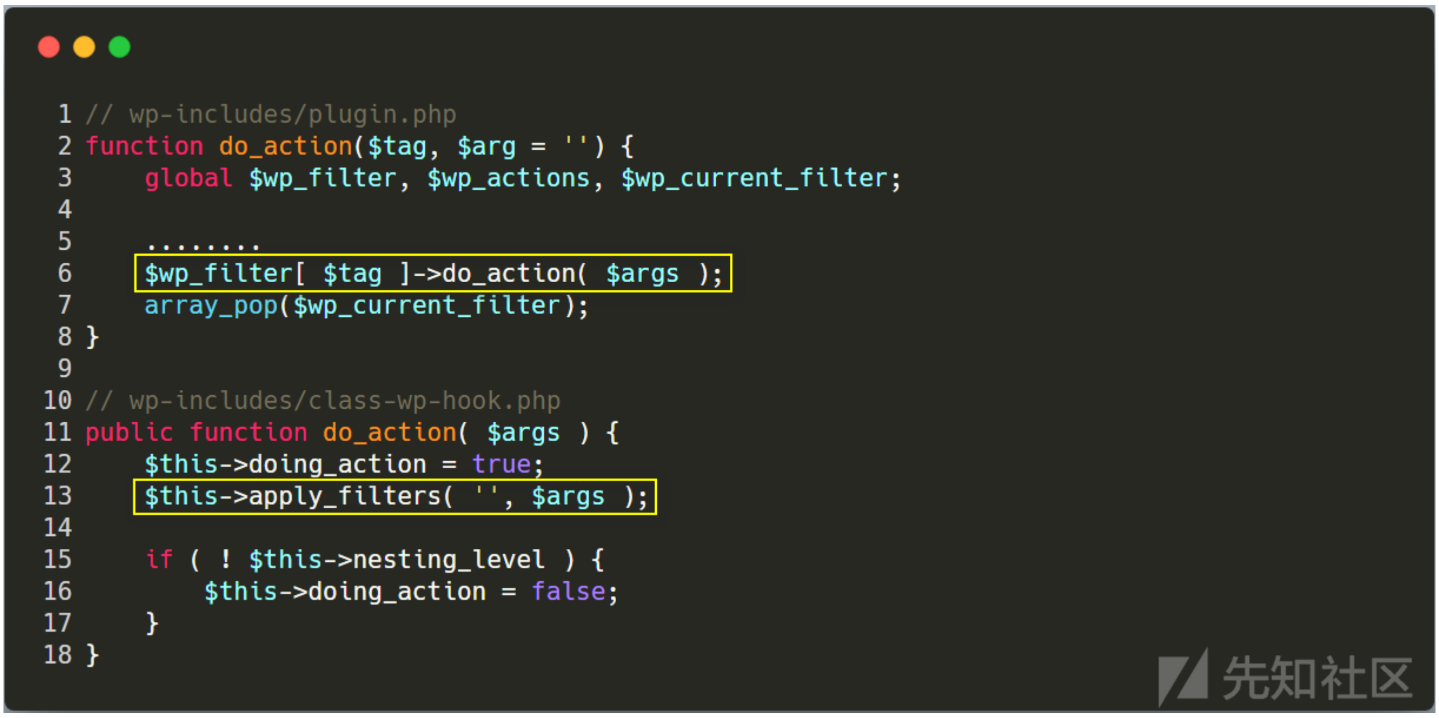
```
POST /wp-admin/admin.php?page=aiowpsec&tab=tab3 HTTP/1.1
Host: localhost
Content-Length: 31
Cache-Control: max-age=0
Origin: http://localhost
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/69.0.3497.100 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://localhost/demo.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie:
wordpress_86a9106ae65537651a8e456835b316ab=admin%7C1539957936%7CKRt9fjdzSsT7xLFnatfOv3
mXwc3xsaXrlmcOfmuWPY5%7C8d9ca78cd6e5de41d1af82e1a78dd2d0cd5a7f039bedcb9a320d79a4a31
4deb5; XDEBUG_SESSION=PHPSTORM; Phpstorm-7fe1b920=48fd0ecb-ddce-4540-8237-1ccf7aedbff8;
ECS[visit_times]=7; _ga=GA1.1.1021725395.1538023195; ishistory=1;
wordpress_test_cookie=WP+Cookie+check;
wordpress_logged_in_86a9106ae65537651a8e456835b316ab=admin%7C1539957936%7CKRt9fjdzSsT7
xLFnatfOv3mXwc3xsaXrlmcOfmuWPY5%7Ccf791926aeb698c8f57d896f5990dbea865e3482b57abcf36fb
8a544ef0141d8; wp-settings-time-1=1538817812
Connection: close

tab="><script>alert(1)</script>
```

可以看到成功引发XSS攻击。我们最后再根据 payload 对代码的调用过程进行分析。首先，我们的 payload 会传入 wp-admin/admin.php 文件中，最后进入第14行 的 do_action('toplevel_page_aiowpsec'); 代码。



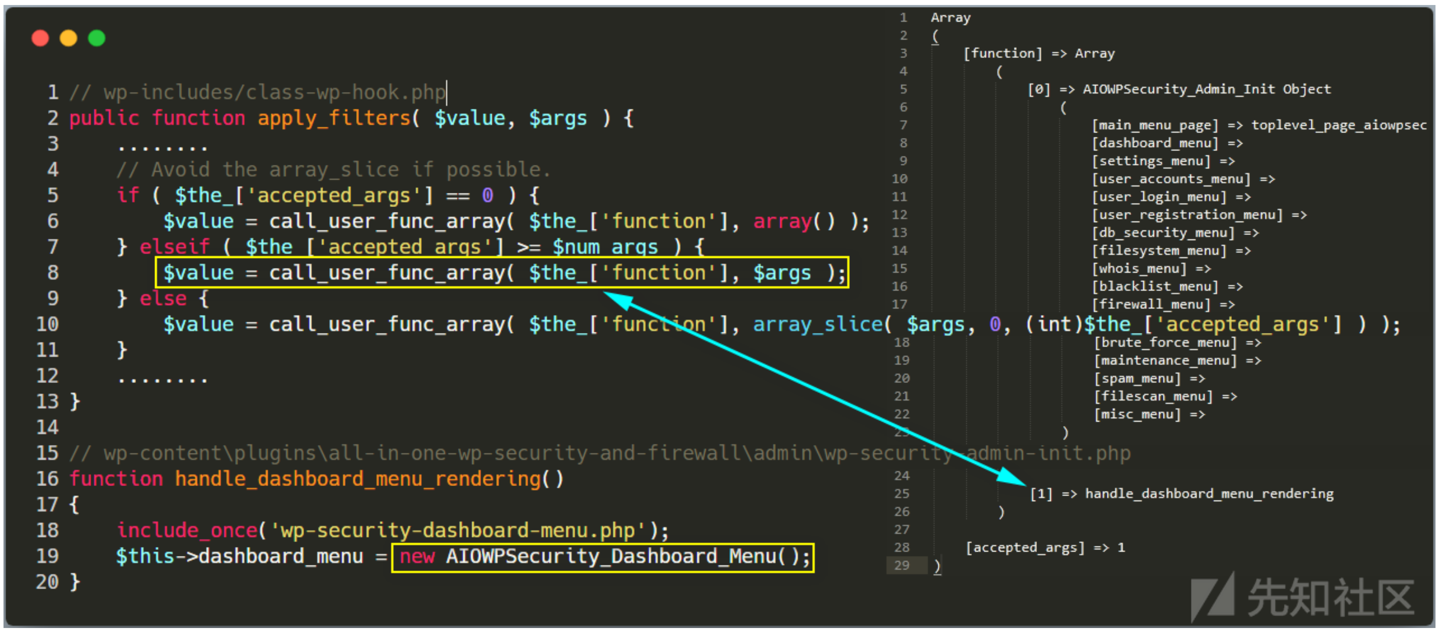
```php
// wp-admin/admin.php
$page_hook = null;

if ( isset($_GET['page']) ) {
    $plugin_page = wp_unslash( $_GET['page'] );
    $plugin_page = plugin_basename($plugin_page);
}

// 通过get_plugin_page_hook函数处理，$page_hook='toplevel_page_aiowpsec'

if ( isset($plugin_page) ) {
    if ( $page_hook ) {
        ........
        do_action( $page_hook );
    }
........
```

在 wp-includes/plugin.php 文件中，程序又调用了 WP_Hook 类的 do_action 方法，该方法调用了自身的 apply_filters 方法。



```php
// wp-includes/plugin.php
function do_action($tag, $arg = '') {
    global $wp_filter, $wp_actions, $wp_current_filter;

    ........
    $wp_filter[ $tag ]->do_action( $args );
    array_pop($wp_current_filter);
}

// wp-includes/class-wp-hook.php
public function do_action( $args ) {
    $this->doing_action = true;
    $this->apply_filters( '', $args );

    if ( ! $this->nesting_level ) {
        $this->doing_action = false;
    }
}
```

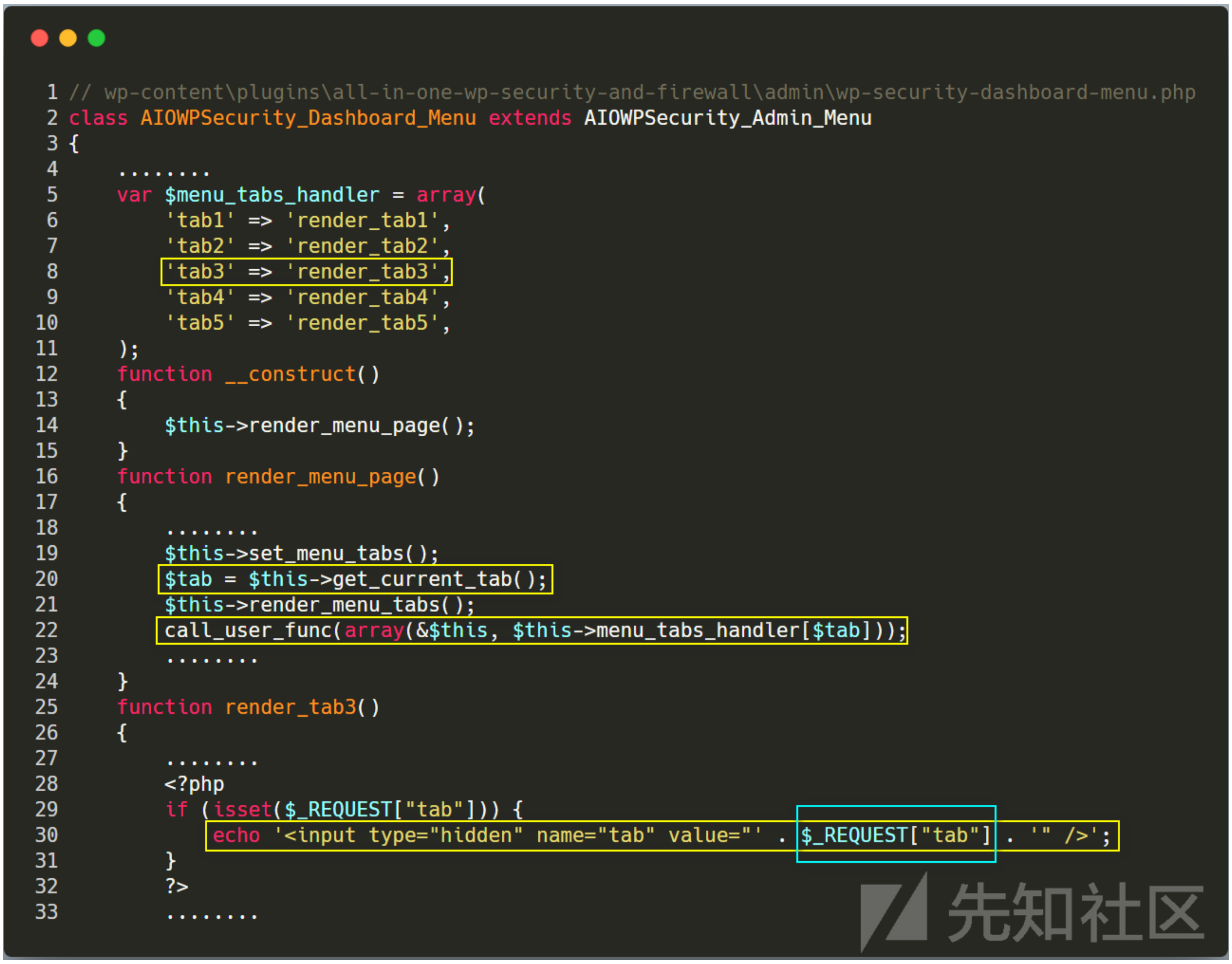
然后 apply_filters 方法调用了 wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-admin-init.php 文件的 handle_dashboard_menu_rendering 方法，并实例化了一个 AIOWPSecurity_Dashboard_Menu 对象。



```php
// wp-includes/class-wp-hook.php
public function apply_filters( $value, $args ) {
    ........
    // Avoid the array_slice if possible.
    if ( $the_['accepted_args'] == 0 ) {
        $value = call_user_func_array( $the_['function'], array() );
    } elseif ( $the_['accepted_args'] >= $num_args ) {
        $value = call_user_func_array( $the_['function'], $args );
    } else {
        $value = call_user_func_array( $the_['function'], array_slice( $args, 0, (int)$the_['accepted_args'] ) );
    }
    ........
}

// wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-admin-init.php
function handle_dashboard_menu_rendering()
{
    include_once('wp-security-dashboard-menu.php');
    $this->dashboard_menu = new AIOWPSecurity_Dashboard_Menu();
}
```

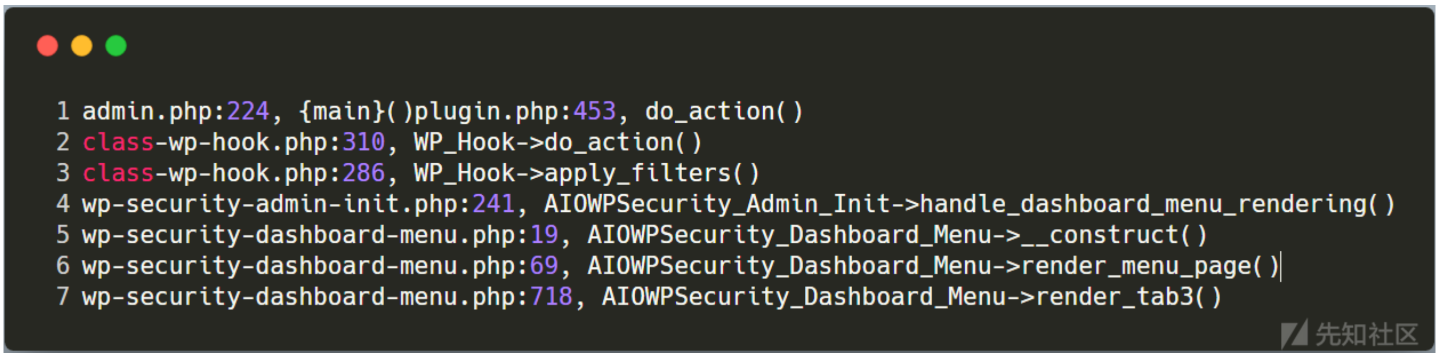
```
Array
(
    [function] => Array
        (
            [0] => AIOWPSecurity_Admin_Init Object
                (
                    [main_menu_page] => toplevel_page_aiowpsec
                    [dashboard_menu] =>
                    [settings_menu] =>
                    [user_accounts_menu] =>
                    [user_login_menu] =>
                    [user_registration_menu] =>
                    [db_security_menu] =>
                    [filesystem_menu] =>
                    [whois_menu] =>
                    [blacklist_menu] =>
                    [firewall_menu] =>
                    [brute_force_menu] =>
                    [maintenance_menu] =>
                    [spam_menu] =>
                    [filescan_menu] =>
                    [misc_menu] =>
                )

            [1] => handle_dashboard_menu_rendering
        )

    [accepted_args] => 1
)
```

接下来就是开头文章分析的部分，也就是下面这张图片：



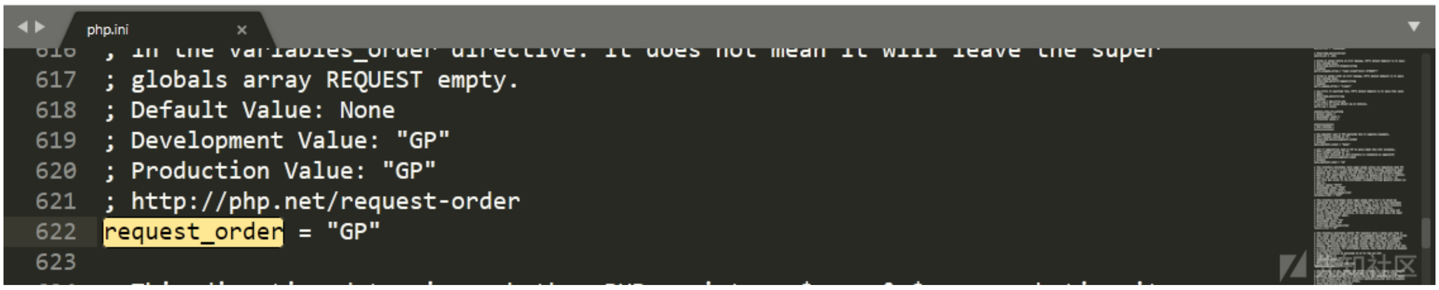
```php
// wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-dashboard-menu.php
class AIOWPSecurity_Dashboard_Menu extends AIOWPSecurity_Admin_Menu
{
    ........
    var $menu_tabs_handler = array(
        'tab1' => 'render_tab1',
        'tab2' => 'render_tab2',
        'tab3' => 'render_tab3',
        'tab4' => 'render_tab4',
        'tab5' => 'render_tab5',
    );
    function __construct()
    {
        $this->render_menu_page();
    }
    function render_menu_page()
    {
        ........
        $this->set_menu_tabs();
        $tab = $this->get_current_tab();
        $this->render_menu_tabs();
        call_user_func(array(&$this, $this->menu_tabs_handler[$tab]));
        ........
    }
    function render_tab3()
    {
        ........
        <?php
        if (isset($_REQUEST["tab"])) {
            echo '<input type="hidden" name="tab" value="' . $_REQUEST["tab"] . '" />';
        }
        ?>
        ........
```

整个漏洞的攻击链就如下图所示：



```
admin.php:224, {main}()plugin.php:453, do_action()
class-wp-hook.php:310, WP_Hook->do_action()
class-wp-hook.php:286, WP_Hook->apply_filters()
wp-security-admin-init.php:241, AIOWPSecurity_Admin_Init->handle_dashboard_menu_rendering()
wp-security-dashboard-menu.php:19, AIOWPSecurity_Dashboard_Menu->__construct()
wp-security-dashboard-menu.php:69, AIOWPSecurity_Dashboard_Menu->render_menu_page()
wp-security-dashboard-menu.php:718, AIOWPSecurity_Dashboard_Menu->render_tab3()
```

这里还有一个小知识点要提醒大家的是，案例中 $_REQUEST["tab"] 最后取到的是 $_POST["tab"] 的值，而不是 $_GET["tab"] 变量的值。这其实和 php.ini 中的 request_order 对应的值有关。例如在我的环境中， request_order 配置如下：



```
; in the variables_order directive. It does not mean it will leave the super
; globals array REQUEST empty.
; Default Value: None
; Development Value: "GP"
; Production Value: "GP"
; http://php.net/request-order
request_order = "GP"

```

这里的 "GP" 表示的是 GET 和 POST ，且顺序从左往右。例如我们同时以 GET 和 POST 方式传输 tab 变量，那么最终用 $_REQUEST['tab'] 获取到的就是 $_POST['tab'] 的值。更详细的介绍可以看如下PHP手册的定义：

```
request_order string
This directive describes the order in which PHP registers GET, POST and Cookie variables into the _REQUEST array. Registration

If this directive is not set, variables_order is used for $_REQUEST contents.

Note that the default distribution php.ini files does not contain the 'C' for cookies, due to security concerns.
```

修复建议

对于这个漏洞的修复方案，我们只要使用过滤后的 $tab 变量即可，且变量最好经过HTML实体编码后再输出，例如使用 htmlentities 函数等。

结语

看完了上述分析，不知道大家是否对 $_REQUEST 数组有了更加深入的理解，文中用到的 CMS 可以从这里(All In One WP Security & Firewall)下载，当然文中若有不当之处，还望各位斧正。如果你对我们的项目感兴趣，欢迎发送邮件到 hongrisec@gmail.com 联系我们。Day16 的分析文章就到这里，我们最后留了一道CTF题目给大家练手，题目如下：

```
// index.php
<?php
function check_inner_ip($url)
{
   $match_result=preg_match('/^(http|https)?:\/\/.*(\/)?.*$/',$url);
   if (!$match_result){
       die('url fomat error1');
   }
   try{
       $url_parse=parse_url($url);
   }
   catch(Exception $e){
       die('url fomat error2');
   }
   $hostname=$url_parse['host'];
   $ip=gethostbyname($hostname);
   $int_ip=ip2long($ip);
   return ip2long('127.0.0.0')>>24 == $int_ip>>24 || ip2long('10.0.0.0')>>24 == $int_ip>>24 || ip2long('172.16.0.0')>>20 == $i
}

function safe_request_url($url)
{
   if (check_inner_ip($url)){
       echo $url.' is inner ip';
   }
   else{
       $ch = curl_init();
       curl_setopt($ch, CURLOPT_URL, $url);
       curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
       curl_setopt($ch, CURLOPT_HEADER, 0);
       $output = curl_exec($ch);
       $result_info = curl_getinfo($ch);
       if ($result_info['redirect_url']){
           safe_request_url($result_info['redirect_url']);
       }
       curl_close($ch);
       var_dump($output);
   }
}

$url = $_POST['url'];
if(!empty($url)){
   safe_request_url($url);
}
else{
   highlight_file(__file__);
}
//flag in flag.php

?>

// flag.php
<?php
if (! function_exists('real_ip') ) {
   function real_ip()
   {
       $ip = $_SERVER['REMOTE_ADDR'];
       if (is_null($ip) && isset($_SERVER['HTTP_X_FORWARDED_FOR']) && preg_match_all('#\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}#s',
           foreach ($matches[0] AS $xip) {
               if (!preg_match('#^(10|172\.16|192\.168)\.#', $xip)) {
                   $ip = $xip;
                   break;
               }
           }
       } elseif (is_null($ip) && isset($_SERVER['HTTP_CLIENT_IP']) && preg_match('/^([0-9]{1,3}\.){3}[0-9]{1,3}$/', $_SERVER['
           $ip = $_SERVER['HTTP_CLIENT_IP'];
       } elseif (is_null($ip) && isset($_SERVER['HTTP_CF_CONNECTING_IP']) && preg_match('/^([0-9]{1,3}\.){3}[0-9]{1,3}$/', $_S
           $ip = $_SERVER['HTTP_CF_CONNECTING_IP'];
       } elseif (is_null($ip) && isset($_SERVER['HTTP_X_REAL_IP']) && preg_match('/^([0-9]{1,3}\.){3}[0-9]{1,3}$/', $_SERVER['
           $ip = $_SERVER['HTTP_X_REAL_IP'];
       }
       return $ip;
   }
}
$rip = real_ip();
if($rip === "127.0.0.1")
   die("HRCTF{SSRF_can_give_you_flag}");
else
   die("You IP is {$rip} not 127.0.0.1");
?>
```

题解我们会阶段性放出，如果大家有什么好的解法，可以在文章底下留言，祝大家玩的愉快！

点击收藏 | 0 关注 | 2

上一篇：Symmetric block c... 下一篇：GPlayed银行木马程序事件分析

1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区