目录

一、题目

1、源码

```php
class FTP {
    public $sock;
    public function __construct($host, $port, $user, $pass) {
        $this->sock = fsockopen($host, $port);

        $this->login($user, $pass);
        $this->cleanInput();
        $this->mode($_REQUEST['mode']);
        $this->send($_FILES['file']);
    }
    private function cleanInput() {
        $_GET = array_map('intval', $_GET);
        $_POST = array_map('intval', $_POST);
        $_COOKIE = array_map('intval', $_COOKIE);
    }
    public function login($username, $password) {
        fwrite($this->sock, "USER " . $username . "\n");
        fwrite($this->sock, "PASS " . $password . "\n");
    }
    public function mode($mode) {
        if ($mode == 1 || $mode == 2 || $mode == 3) {
            fputs($this->sock, "MODE $mode\n");
        }
    }
    public function send($data) {
        fputs($this->sock, $data);
    }
}

new FTP('localhost', 21, 'user', 'password');
```

2、知识点

知识点	说明
fsockopen()	用于打开网络的Socket连接
fputs()	将内容写入一个打开的文件中

3、解读

```
    1）第30行，实例化FTP对象。
    2）第3行，实例化对象时会自动触发魔法函数__construct()，设置sock变量为socket连接。
    3）第6行，调用了函数login()，传入用户名和密码。
    4）第16-19行，函数login()中，将用户名和密码写入到了socket通道。
    5）第7行，调用了函数cleanInput()。
    6）第11-15行，函数cleanInput()中，过滤GET、POST、COOKIE的数据，强转为整数类型。
    7）第8行，调用了函数mode()，并传入$_REQUEST方法接收到的mode值。
    8）第20-24行，函数mode()中，如果接收到的mode值等于1、2、3中任意一个值，就将其写入到socket通道中。
    9）第9行，调用了函数send()，并传入$_FILE方法接收到的file值。
    10）第25-27行，函数send()中，将file值写入到socket通道中。
```

4、分析

```
    1）漏洞1：第7行，函数cleanInput()中将GET、POST、COOKIE中的数据强转为整数类型，但是在第8行，又使用了$_REQUEST接收数据，那么$_REQUEST是可以接收GET 和 POST方法中的数据的，此时也就等于函数cleanInput()无法发挥作用。
    2）漏洞2：第21行，对mode的值做了弱类型比较，也就导致可以在传入数据时带上其他数据。
```

5、利用

```
    ?mode=1%0a%0dDELETE%20xxx.file
```

二、CMS-All In One WP Security & Firewall 4.14

1、知识点

```
    Null
```

2、分析

```
    1）该漏洞产生于 4.14 - 4.19版本，在 wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-dashboard-menu.php（简化）文件中。第8行，tab3的值引用了函数render_tab3()，跟进到25行的render_tab3()，其中通过$_REQUEST()接收tab的值放入HTML标签中。随后在20行调用了函数get_current_tab()。
```

```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"] . '" />';
        }
        ?>
        ........
```

2) 在20行中，对GET方法接收到的tab进行了判断（由于这个tab值是通过$_REQUEST接收的，也就是GET和POST方法都接收，那么这里并没有对POST中的tab进行判断），并使用函数sanitize_text_field()过滤。

```php
// WWW\wp-content\plugins\all-in-one-wp-security-and-firewall\admin\wp-security-dashboard-menu.php

function get_current_tab()
{
    $tab_keys = array_keys($this->menu_tabs);
    $tab = isset($_GET['tab']) ? sanitize_text_field($_GET['tab']) : $tab_keys[0];
    return $tab;
}
```

3) 过滤函数的调用链在下图第一行，最后是进入到了函数wp_check_invalid_utf8()的检测。这里没有对POST中的tab进行过滤，并且数据最后拼接在了HTML标签中，所以就造成了XSS攻击。

```
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 '';
}
```

3、利用

```
GET：http://xxx.com/wp-admin/admin.php?page=aiowpsec&tab=tab3
POST：tab="><script>alert(1)</script>
```

4、修复方案

```
1）$_REQUEST接收的数据就要同时处理POST和GET。
2）对$tab进行过滤，最好使用HTML实体编码，例如htmlentites函数。
```

5、参考链接

```
https://github.com/hongriSec/PHP-Audit-Labs/blob/master/Part1/Day16/files/README.md
```