Joomla 3.0-3.4.6 rce分析

小透明yo / 2019-10-10 09:10:06 / 浏览数 5331 安全技术 漏洞分析 顶(1) 踩(0)

前言:

本文只分析发生漏洞得原因，具体pop链简略分析。

joomla中得session会被存入数据库中，这是以前版本得RCE就可以得知得事情。

/libraries/joomla/session/storage.php:

```
public function register()
   {
       // Use this object as the session handler
       session_set_save_handler(
           array($this, 'open'), array($this, 'close'), array($this, 'read'), array($this, 'write'),
           array($this, 'destroy'), array($this, 'gc')
       );
   }
```

通过这里得到目标注册得几个函数，但是此方法为抽象类，也就是说不能实例化的，所以我们需要寻找继承了此类的类进行分析,在JSessionStorageDatabase对象中，均重

0x01 入口：

根据github给出的payload得出路由为：/index.php/component/users
根据路由找到目标文件的真实文件为：/components/com_users/users.php
此文件获取了一个task参数，这个参数不做具体分析，我们只需要得知目标会根据此参数来找到最终提交的函数
payload中有如下：

```
'task': 'user.login',
```

也就是说会提交到 user控制器下面的login方法，直接追过去就好了，具体路径为：components/com_users/controllers/user.php
代码：

```
public function login()
   {
       JSession::checkToken('post') or jexit(JText::_('JINVALID_TOKEN'));

       $app    = JFactory::getApplication();
       $input  = $app->input;
       $method = $input->getMethod();

       // Populate the data array:
       $data = array();

       $data['return']    = base64_decode($app->input->post->get('return', '', 'BASE64'));
       $data['username']  = $input->$method->get('username', '', 'USERNAME');
       $data['password']  = $input->$method->get('password', '', 'RAW');
       $data['secretkey'] = $input->$method->get('secretkey', '', 'RAW');

       // Don't redirect to an external URL.
       if (!JUri::isInternal($data['return']))
       {
           $data['return'] = '';
       }

       // Set the return URL if empty.
       if (empty($data['return']))
       {
           $data['return'] = 'index.php?option=com_users&view=profile';
       }

       // Set the return URL in the user state to allow modification by plugins
       $app->setUserState('users.login.form.return', $data['return']);

       // Get the log in options.
       $options = array();
```

```
        $options['remember'] = $this->input->getBool('remember', false);
        $options['return']   = $data['return'];

        // Get the log in credentials.
        $credentials = array();
        $credentials['username']  = $data['username'];
        $credentials['password']  = $data['password'];
        $credentials['secretkey'] = $data['secretkey'];

        // Perform the log in.
        if (true === $app->login($credentials, $options))
        {
            // Success
            if ($options['remember'] == true)
            {
                $app->setUserState('rememberLogin', true);
            }

            $app->setUserState('users.login.form.data', array());
            $app->redirect(JRoute::_($app->getUserState('users.login.form.return'), false));
        }
        else
        {
            // Login failed !
            $data['remember'] = (int) $options['remember'];
            $app->setUserState('users.login.form.data', $data);
            $app->redirect(JRoute::_('index.php?option=com_users&view=login', false));
        }
    }
```

0x02 进入login中:

这里我们可以看下重点代码：

```
JSession::checkToken('post') or jexit(JText::_('JINVALID_TOKEN'));
```

进入checkToken函数中,具体看下

```
$session = JFactory::getSession();
           if ($session->isNew())
```

跟进后发现这句代码获取了现在的session对象：

```
public static function getSession(array $options = array())
    {
        if (!self::$session)
        {
            self::$session = self::createSession($options);
        }

        return self::$session;
    }
```

这里获取到的对象其实就是当前对象，因为我在下面发现了isNew函数:

```
public function isNew()
    {
        $counter = $this->get('session.counter');
        return (bool) ($counter === 1);
    }
```

然后在跟进get函数:

```
public function get($name, $default = null, $namespace = 'default')
    {
        // Add prefix to namespace to avoid collisions
        $namespace = '__' . $namespace;
        if ($this->_state === 'destroyed')
        {
            // @TODO :: generated error here
            $error = null;
```

```
            return $error;
        }
        if (isset($_SESSION[$namespace][$name]))
        {
            return $_SESSION[$namespace][$name];
        }
        return $default;
    }
```

也就是说此时return的是：

```
$_SESSION[__default][session.counter]
```

因为\$this->_state === 'destroyed' 判断根本不成立，在start函数中，有如下代码:

```
public function start()
    {
        if ($this->_state === 'active')
        {
            return;
        }
        $this->_start();
        $this->_state = 'active';
```

他会将_state变量预设为active。此时的counter获取出来确实是等于1，所以会直接返回true

```
public function isNew()
{
    $counter = $this->get( name: 'session.counter');  $counter: 1
    return (bool) ($counter === 1);
}
```

```
return (bool) ($counter === 1);
```

返回为真，再次回到checktoken函数：

```
if ($session->isNew())
            {
                // Redirect to login screen.
                $app->enqueueMessage(JText::_('JLIB_ENVIRONMENT_SESSION_EXPIRED'), 'warning');
                $app->redirect(JRoute::_('index.php'));
            }
            else
            {
                return false;
            }
```

然后进入if中的真流程，重点可以看下这句：

```
$app->redirect(JRoute::_('index.php'));
```

我们跟进redirect函数：

```
public function redirect($url, $status = 303)
    {
        // Handle B/C by checking if a message was passed to the method, will be removed at 4.0
        if (func_num_args() > 1)
        {
            $args = func_get_args();

            /*
             * Do some checks on the $args array, values below correspond to legacy redirect() method
             *
             * $args[0] = $url
             * $args[1] = Message to enqueue
             * $args[2] = Message type
             * $args[3] = $status (previously moved)
             */
            if (isset($args[1]) && !empty($args[1]) && (!is_bool($args[1]) && !is_int($args[1])))
```

```
            {
                // Log that passing the message to the function is deprecated
                JLog::add(
                    'Passing a message and message type to JFactory::getApplication()->redirect() is deprecated. '
                    . 'Please set your message via JFactory::getApplication()->enqueueMessage() prior to calling redirect().',
                    JLog::WARNING,
                    'deprecated'
                );

                $message = $args[1];

                // Set the message type if present
                if (isset($args[2]) && !empty($args[2]))
                {
                    $type = $args[2];
                }
                else
                {
                    $type = 'message';
                }

                // Enqueue the message
                $this->enqueueMessage($message, $type);

                // Reset the $moved variable
                $status = isset($args[3]) ? (boolean) $args[3] : false;
            }
        }

        // Persist messages if they exist.
        if (count($this->_messageQueue))
        {
            $session = JFactory::getSession();
            $session->set('application.queue', $this->_messageQueue);
        }

        // Hand over processing to the parent now
        parent::redirect($url, $status);
    }
```

看着这么长一串实际上这玩意，emmm 啥也没干，因为第一个if,我们只传入了一个变量所以直接跳过,第二个if判断中只设置了一个session变量。

```
parent::redirect($url, $status);
```

再次跟入parent::redirect,一长串代码，其实还是什么也没干，在那组合url，到最后执行到了`$this->close();`
而close中的代码为：

```
public function close($code = 0)
    {
        exit($code);
    }
```

分析到此处的时候，我不禁陷入了对人生以及社会的大思考当中。tmd到底在哪里写入了session？后来回到刚刚走过的代码再次认真的看了一次后发现，在_start中还有这么

```
register_shutdown_function('session_write_close');
```

可以看下官方给出的定义:

> 注册一个会在php中止时执行的函数
> 简单的来说就是整个php程序的__destract(),在php结束之前均会执行此代码。

然后可以看下write:

```
public function write($id, $data)
    {
        // Get the database connection object and verify its connected.
        $db = JFactory::getDbo();

        $data = str_replace(chr(0) . '*' . chr(0), '\0\0\0', $data);

        try
```

```
        {
            $query = $db->getQuery(true)
                ->update($db->quoteName('#__session'))
                ->set($db->quoteName('data') . ' = ' . $db->quote($data))
                ->set($db->quoteName('time') . ' = ' . $db->quote((int) time()))
                ->where($db->quoteName('session_id') . ' = ' . $db->quote($id));
```

这里获取的两个参数分别为:cookie中的sessionid以及序列化组合过后的session。
重点看下面这句替换的代码：

```
$data = str_replace(chr(0) . '*' . chr(0), '\0\0\0', $data);
```

会将chr(0) . '' . chr(0)替换为\0\0\0,正因为这个机制造成了这次的RCE,chr(0).\.chr(0)为三个字节长度,但是\0\0\0为6个字节长度。后面的read代码中，将所有的\0\0\0全部替换成了chr(0).

```
public function read($id)
   {
       // Get the database connection object and verify its connected.
       $db = JFactory::getDbo();

       try
       {
           // Get the session data from the database table.
           $query = $db->getQuery(true)
               ->select($db->quoteName('data'))
           ->from($db->quoteName('#__session'))
           ->where($db->quoteName('session_id') . ' = ' . $db->quote($id));

           $db->setQuery($query);

           $result = (string) $db->loadResult();

           $result = str_replace('\0\0\0', chr(0) . '*' . chr(0), $result);

           return $result;
       }
```

实验：

实验代码：

```
<?php
class a{
   public $a;
   function __construct()
   {
       $this->a = chr(0) . '*' . chr(0);
   }
}

echo serialize(new a());

?>
```

输出:

```
O:1:"a":1:{s:1:"a";s:3:"*";}
```

实验代码2：

```
<?php
class a{
   public $a;
   function __construct()
   {
       $this->a = '\0\0\0';
   }
}

echo str_replace('\0\0\0' , chr(0) . '*' . chr(0),serialize(new a()));
```

```
?>
```

输出:

```
O:1:"a":1:{s:1:"a";s:6:"*";}
```

可以看出将\0替换为了chr(0)后并没有替换长度。

payload分析：

此时的payload为:

```
__default|a:8:{s:15:"session.counter";i:5;s:19:"session.timer.start";i:1570637551;s:18:"session.timer.last";i:1570639080;s:17:
```

然后将\0\0\0替换为特殊字符后:

```
O:1:"a":1:{s:1:"a";s:1995:"__default|a:8:{s:15:"session.counter";i:5;s:19:"session.timer.start";i:1570637551;s:18:"session.tim
```

其中:

```
s:54:"*********"
```

实际占位为27位但是这里却是54位，多出来的27位从后面补入，此时的payload实际上为：

```
s:54:[*********";s:8:"password";s:409:"AAA]
```

最后得出真正被反序列化的是：

```
s:11:"maonnalezzo":O:21:"JDatabaseDriverMysqli":3:{s:4:"*a";O:17:"JSimplepieFactory":0:{}s:21:"*disconnectHandlers";a:1:{i:0;a
```

而username中的N个\0以及password中的AAA早就被程序逻辑导致的溢出吃的一干二净了。

POP链分析：

在payload中可以得到目标pop链的入口为:JDatabaseDriverMysqli,我们直接追进去就好了:

```
public function __destruct()
   {
       $this->disconnect();
   }
```

跟进disconnect方法：

```
public function disconnect()
   {
       // Close the connection.
       if ($this->connection)
       {
           foreach ($this->disconnectHandlers as $h)
           {
               call_user_func_array($h, array( &$this));
           }

           mysqli_close($this->connection);
       }

       $this->connection = null;
   }
```

这里的`call_user_func_array($h, array( &$this));`简直和thinkphp中反序列化pop链那个一模一样，只能控制第一个参数,所以我们需要进行这样调用:

```
call_user_func_array([$obj,"■■■■"],array( &$this))
```

到这一步就很简单了，按照pop链的方法来看就在：
/libraries/simplepie/simplepie.php

```
if ($this->feed_url !== null || $this->raw_data !== null)
       {
           $this->data = array();
           $this->multifeed_objects = array();
           $cache = false;
```

```
        if ($this->feed_url !== null)
        {
            $parsed_feed_url = SimplePie_Misc::parse_url($this->feed_url);
            // Decide whether to enable caching
            if ($this->cache && $parsed_feed_url['scheme'] !== '')
            {
                $cache = call_user_func(array($this->cache_class, 'create'), $this->cache_location, call_user_func($this->c
```

关键点在这:

```
call_user_func(array($this->cache_class, 'create'), $this->cache_location, call_user_func($this->cache_name_function, $this->f
```

点击收藏 | 0 关注 | 1

上一篇：Joomla 3.4.6 RCE 分析 下一篇：iis6.0（cve-2017-7...

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

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