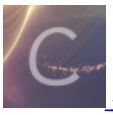
怎样进行代码审计

原创

合天网安实验室 于 2020-03-19 16:00:40 发布 1076 收藏 3

分类专栏： 代码审计

本文链接：https://blog.csdn.net/qq_38154820/article/details/104969514



怎样来php代码审计

概况：

- 工具和常见的几种思想
- 常见的点
- 一套简单的CMS源码审计
- 总结

这篇是关于php；

常见的也是php的代码审计，类似java框架，asp.net，python的Django还是有漏洞之类的，只是比较固定和稀少；

像php这种语言本身就是弱语言类型（很多点都利用了弱类型比较和转换），功能函数也是奇多（导致了其与sql，xml等语言的花式组合），加之处理一个问题的方法也是奇多（其中就难保不会出现漏洞）

当然php的thinkphp框架，有机会来（这款框架就安全性没有其他几种语言那么强的安全措施，倒转还有一个烂大街的log写webshell的常见问题，以及最近挺火的远程代码执行）

一.工具和常见的几种思想

（一）工具

这里 我介绍两款免费常用的审计工具和一款php编写调试工具

1.

Seay源代码审计系统

下载地址：https://www.waitalone.cn/seay-source-code-auditv2.html

2.（PHP代码审计工具——rips）

下载地址: https://sourceforge.net/projects/rips-scanner/

使用教程: https://www.cnblogs.com/Jewel591/p/7477483.html

3.phpstorm

如果对于版本没有什么要求

直接按照http://www.3322.cc/soft/17468.html激活即可

这款工具的调试运行，作者还是多提醒一点

.php等文件必须在www文件夹下，其他的搜下就知道了

4.xdebug

这款debug暂时还没有用，不过看各位前辈经常用到，特提出来

（二）几种常见的审计思想

其实个人认为就两种，由点破面，由面破点

当然还有

1.由点破面

根据经验和工具找漏洞关键词，来溯源调用过程

看是否可控，可控后看调用的输入入口

如果单个可控条件满足我们的要求，但是无法实施漏洞触发，再全局看下哪里有满足我们条件的地方，组合起来触发

其中用到我们的工具Seay源代码审计工具

2.由面破点

通读全文，理清大意，再根据问题关键词，勾画对应位置 —英语阅读

深入理解一套CMS源码就是这样

3.

如果追求速度，那么无疑第一种最好，也是入门首选，暂时只专注于问题地方（作者目前也是用的第一种）

二.常见的点

这里就不写那些有的没的了，只详细给出作者认为个人目前阶段审计挖掘常用的点

推荐学习：

1.php常见漏洞

审计中弱类型比较，首当其冲，其他的函数问题大多围绕这个比较特性

== 与=== 的差别；

数组，字符，数字，对象之间的比较转换；

当然还有其他web漏洞，这里不多说

2.php常见漏洞函数

还是必须要写出来有个清晰的认识，写出一些前辈的总结吧

0.

先写出 == 与 ===的差别

=== 在两个数比较时，也会比较参数的类型

来看一个代码简单理解下

```
<?php
if("0admin" == 0){
    echo "ok1"."<br>";
}
if("0a" === 0){
    echo "ok2"."<br>";
}
```

没错，只输出了 ok1，第二个进行了类型的比较

1.

md5的0开头比较，利用了php其弱类型比较特点

其实也是利用了 ==来比较造成的漏洞

从网上随便抄了两个payload，看下下面的代码

```
<?php
$a=md5("s155964671a");
$b=md5("s214587387a");
echo $a."<br>";
echo $b."<br>";
if($a == $b){
    echo "ok1"."<br>";
}
if($a === $b){
    echo "ok2"."<br>";
}
```

输出也只有 ok1：

0e342768416822451524974117254469
0e848240448830537924465865611904
ok1

2.

ereg的00截断，一道很多ctf类型的题原型

```php
<?php
if (isset ($_GET['password'])) {
   if (ereg ("^[a-zA-Z0-9]+$", $_GET['password']) === FALSE)
   {
      echo '<p>You password must be alphanumeric</p>';
   }
   else if (strlen($_GET['password']) < 8 && $_GET['password'] > 9999999)
   {
      if (strpos ($_GET['password'], '*-*') !== FALSE)
      {
         die('Flag: ' . $flag);
      }
      else
      {
         echo('<p>*-* have not been found</p>');
      }
   }
   else
   {
      echo '<p>Invalid password</p>';
   }
}
?>
```

payload ?password=1e9%00*-*

字符长度满足，内容包括特殊字符且为a-zA-Z0-9，且大于9999999，%00后内容ereg不再判断

3.strcmp

判断输入密码是否相等

如果 str1 小于 str2 返回 < 0；
如果 str1 大于 str2 返回 > 0；
如果两者相等，返回 0。

```php
<?php
$m=$_GET['a'];
$n="admin";
if(strcmp($m,$n) == 0){
   echo "ok1"."<br>";
}
if(strcmp($m,$n) === 0){
   echo "ok2"."<br>";
}
```

paylaod a[]=1

也只返回了ok1

显然还是利用了我们的 ==

这里payload可以是数组，也可以是对象（但是作者目前还没有找到合适的payload）

4.

md5和sha1不能处理数组

先来md5，这个很有意思，数组可以绕开 === 判断

```php
<?php
$a=$_GET['a'];
$b=$_GET['b'];
if(md5($a) == md5($b)){
    echo "ok1"."<br>";
}
if((md5($a) === md5($b))){
    echo "ok2"."<br>";
}
```

payload ?a[]=1&b[]=2

ok1 和 ok2都输出了

同样sha1

```php
<?php
$a=$_GET['a'];
$b=$_GET['b'];
if(sha1($a) == sha1($b)){
    echo "ok1"."<br>";
}
if((sha1($a) === sha1($b))){
    echo "ok2"."<br>";
}
```

payload ?a[]=1&b[]=2

ok1 和 ok2都输出了

5.

is_numeric

输出这个代码观察

```php
<?php
if(is_numeric(0x23)) {
    echo "ok1" . "<br>";
}
if(is_numeric("23aa")) {
    echo "ok1" . "<br>";
}
```

只有ok1，说明我们能用其他编码绕过

6.in_array

也是一个弱类型，当in_array函数第三个参数不为true时

1sh.php会被认为是1.php

具体下面介绍的项目有练习

7.

serialize 和 unserialize漏洞

有点多，有兴趣自己搜索下

浅显的认识可以参考作者的个人博客：
http://47.106.140.244/2018/08/php%e5%8f%8d%e5%ba%8f%e5%88%97%e5%8c%96%e6%bc%8f%e6%b4%9e/

8.filter_var()的问题

FILTER_VALIDATE_URL 这个问题，很容易就可以绕过的

给出几个前辈的paylaod

?url=http://xxx.com()sec-redclub.com

()里换成@&?/,:#这些任意一个即可

9.htmlentities()

ENT_COMPAT（默认值）：只转换双引号。
ENT_QUOTES：两种引号都转换。
ENT_NOQUOTES：两种引号都不转换。

算一个bug吧

10.就不继续列举了，有兴趣自己去扩展下视野

3.

有一个代码审计的项目蛮有趣的，https://github.com/hongriSec/PHP-Audit-Labs，也是作者这几天在学习的内容

4.

这里还是想推荐sqli-labs注入天书，因为审计中存在很多硬刚sql注入的正则绕过

5.也可以参考作者的个人博客总结

http://47.106.140.244/2018/09/%e5%88%9d%e6%8e%a2php%e5%ae%a1%e8%ae%a1/

http://47.106.140.244/category/%E4%BB%A3%E7%A0%81%E5%AE%A1%E8%AE%A1/

分类：

太多，显得臃肿，太少，显得不走心

所以这里作者把知识点分类（读者也可以自己去找下思维图）

- 防护类
- sql注入
- 代码或命令执行
- 文件操作
- 其他类
- 其他的情况

0.防护类

先把它单独拉出来分析，总没错

sql，xss过滤函数等等

无外乎看看正则过滤，preg_replace，replace等等

具体的绕过，下面分析

1.SQL注入类

这个问题，作者反复想了下

主暂时分为三类

```
1.没有任何防护
2.硬刚正则
3.二次注入
```

1.没有任何防护

漏洞嘛，多找找，极具耐心，总能找到一个点的

翻看了以前dede的漏洞，发现很多都是漏掉的防护点导致的问题

自己以前审计的大米cms的存储xss，就是没有任何防护，sql防护倒是可以

不多说

2.硬刚正则

这个可能是极有意思的主题了

目前看到绕过正则的一共这几种

（1）查找正则匹配遗留的关键词

等于= 换成regexp like等

延时注入的代替 推荐一篇文章：https://www.cdxy.me/?p=789

除开-- + – - # %23等，还有一个很有意思的注释 ;%00

（2）其实，更多的作者看到的结合其他的奇思绕过（注释，闭合）

hpp污染，md5(str,true)，超全局变量覆盖，request请求的差异等，addslashes函数和gpc滥用

这个作者只有提醒下，具体要自己慢慢接触

3.二次注入

也简单分为两种

1.入数据库函数一次，出数据库函数一次，就还原了

2.利用其他没有检查的数据进行注入

4.

作者看到的大多数情况都要和其他齐思结合，只能说有一个点有问题，就再去寻找另一个点与之结合突破

2.代码或命令执行

像system eval 函数系统命令调用，反引号，动态函数执行就不在这讨论了，遇到了就去考虑绕过或者替换就行了

有一个点作者一直觉得很有意思，preg_replace /e修正执行代码

如果在构造正则表达式的时候，使用了/e修正符，这时，preg_replace() 就会将 replacement 参数当作 PHP代码执行。但是replacement一般都是开发者事先写好的，这是就要考虑怎样去替换掉了，超全局变量覆盖，get，post，request的请求解析顺序，hpp污染

3.文件操作

作者分为四类：上传绕过，任意文件读写，任意文件删除，文件包含

具体文件操作函数，读者自己搜下，因为都有用，所以就不挑到写了

1.上传绕过

作者不多说什么函数问题，个人觉得黑盒的一个知识更有趣；可以不会其他，这个必须会，拿shell呀

https://github.com/c0ny1/upload-labs 推荐一套github上的源码，有20个知识点吧，就不复制过来占位置了

https://github.com/LandGrey/upload-labs-writeup/ 这套源码的解题答案（本来想写文章的，看了这篇解题，瞬间写的想法没有了，哈哈哈）

http://47.106.140.244/2018/10/%e6%96%87%e4%bb%b6%e4%b8%8a%e4%bc%a0%e7%9a%84%e5%a7%bf%e5%8a%bf/ 个人博客关于文件上传的总结

2.任意文件读写

读文件，作者目前还没有观察到明显的函数漏洞

个人遇到比较多的还是log的写shell，如thinkphp的问题

其他的看到后台写模板的情况比较多

3.任意文件删除

unlink函数

一个很容易忽视的点，开发人员和审计人员

4.任意文件包含

分为本地包含和远程包含，看配置文件

一般是绕过正则，进行其他有用文件的包含以及php伪协议进行其他文件内容的查看

4.其他类

这个点是作者认为最有意思的，因为前面写的很多地方都有总结

这个点的宗旨就是结合各种情况进行利用

前面其实作者都写了，只提一点核心思想

既是“**有一个点有问题，就再去寻找另一个点与之结合突破**”

5.其他的情况

不谈学习知识，能不硬刚正则就不硬刚

这种，作者比较关心的偏门的地方和情况

1.header头的sql注入

暂不谈crlf注入以及xss等头跳转，是开发人员对user-agent，refer，cookies进行了一个数据库的入库查询，但是并没有像post和get的数据一样进行check_sql和check_xss等检查

就构成了没有任何防护的注入

其中也有可能系统开了gpc的，但由于它们是在$_SERVERE变量中不受GPC的影响，那我们就可以去查找HTTP_CLIENT_IP和HTTP_X_FORWARDFOR关键字来快速寻找漏洞

2.宽字节只提一笔

3.也是gpc的问题

有些CMS源码中并没有判断gpc情况，直接addslashes一次，如果本身就开启了，相当于两次转义了，可以使用函数get_magic_quotes_gpc() 进行检测

4.超全局变量

常见的三个$_GET $_POST $_REQUEST，当然还有其他_FILE等

这个有什么用？变量覆盖结合其他点进行绕过，注入，写文件，getshell等

PHP-Audit-Labs项目day-15，还是day-14就有一个题，可以去看看

5.这里给出一个很有用的知识点

php中双引号解析变量、而单引号不解析变量（相信很多新手都有一会双信号，一会单引号的习惯）

运行

```
<?php
$s="ok";
echo "1".'$s'."<br>";
echo "2"."$s";
```

6.其他版本以及配置问题，并不在我们开始审计的范畴，就不多讨论了

三.一套简单的CMS源码审计

这套源码是作者写这篇文章现找的，目的就是更真实还原个人是怎样审计的

直接给出哪有问题，对于个人而言太没有学习价值了

可以去这找找http://www.mycodes.net/43/，拿小的CMS练手

这里作者用的phpcom这套源码审计 ， 下载链接： http://www.mycodes.net/43/5429.htm

这里提醒一下，不怕它防，就怕我们看不懂它们，因为知识点其实就那些

1.sql

来看下sql的问题，发现它的数据库操作都是往DB类送

找到class DB，在src/class/db/database.php里

作者发现里面没有任何防御，有点失望的时候，去页面搜索框payload了一下，发现非法字符，搜索后发现确实是做了全局防御的，但是是在phpcom.php里做的

看下输入函数里：

```
if (!MAGIC_QUOTES_GPC) {
    $_GET = phpcom::addslashes($_GET);
    $_POST = phpcom::addslashes($_POST);
    $_COOKIE = phpcom::addslashes($_COOKIE);
    $_FILES = phpcom::addslashes($_FILES);
}
```

这里就做的很好，先判断了的

作者没有发现ip入库操作，但是有获取ip的操作，不过在phpcom.php用了htmlspecialchars函数来防御，其他文件继承了它的防御

2.xss

phpcom.php里

来看下这个防御xss的函数

```
public function check_urlxss() {
    $uri = strtoupper(urldecode(urldecode($_SERVER['REQUEST_URI'])));
    if (strpos($uri, '<') !== FALSE || strpos($uri, '"') !== FALSE || strpos($uri, 'CONTENT-TRANSFER-ENCODING') !== FALSE) {
        showmessage('request_tainting');
    }
    return TRUE;
}
```

两次url解码后再全部转为大写，看到strpos函数，想起了前几天看到的一个有趣的花式

strpos 函数返回查找到的子字符串的下标。如果字符串开头就是我们要搜索的目标，则返回下标 **0**；如果搜索不到，则返回 **false**（而本来搜索到是返回true的）

这里我们可以不用<，但必须用到双引号" 来闭合value的双引号，本来是过了strpos的，可惜用了 === 来判断，如果前面过了，后面的双引号可以用注释符号，这样只是页面会出错而已

```
if(strpos($a,'"') == FALSE){
    echo "222";
}
```

（2）作者还看到用了几次的header函数

```
phpcom::header('Location: ' . $_SERVER['HTTP_REFERER']);
```

```
public static function header($string, $replace = true, $http_response_code = 0) {
    $string = str_replace(array("\r", "\n"), array('', ''), $string);
    @header($string, $replace, $http_response_code);
    if (preg_match('/^\s*location:/is', $string)) {
        exit();
    }
}
```

过滤 \r \n 预防crlf

3.因为是练习，所以install.php文件，顺便看下

```
$lockfile = ROOT_PATH . 'data/install.lock';
if(file_exists($lockfile)) {
    show_message('install_locked');
```

它在文件开头直接判断了的，不存在调到step的步骤问题，略

4.代码执行

作者翻到一个反引号，可以作为执行代码，数组的键值

```
function implode_field_value($array, $glue = ',') {
    $sql = $comma = '';
    foreach ($array as $k => $v) {
        $sql .= $comma."`$k`='$v'";
        $comma = $glue;
    }
    return $sql;
}
```

跟踪它，可以发现，phpcom.php 数据库的增删改查都用到了它，说明，全套源码有数据库操作的页面，就可以看下是否存在变量覆盖漏洞

溯源跟踪代码每个页面，搜索看有没有变量覆盖的地方（结合seay审计工具查看变量覆盖特征函数）

作者搜索发现像 $$ 这种覆盖，键值都是确定了的，意味着无法入手，考虑结合变量覆盖

这套源码还有像parse_str这样的覆盖，也是人工的，这里无用

找到eval函数，也是发现写死了的

找到call_user_func_array函数，暂时无理解，不分析，记得webshell的免杀它很有用

preg_replace /e操作是注释了的，看来开发人员也拿不准

如果有幸存在超全局变量，也是个思路

5.文件操作（能力原因，还有很大的操作空间）

用seay工具最烦的就是文件操作类，因为大部分都是它误报（因为基本上关于文件的变量都是写死了的），所以放到最后看

（1）像include等包含太多了，作者也没有跟踪代码

文件写入和读取

（2）寻找php file_get_contents

有两个地方转了4，5个文件发现文件名变量前面两个是环境变量，写死了的

其实只要一个最后一个可控就行，就可能进行目录穿梭读取

（3）寻找file_put_contents

也是写死了的

（4）@ulink删除文件函数的跟踪有时间再来观察

（5）当然还有fread等文件操作函数，实在不想跟了，留下来自己看吧

6.当然如果老老实实的把上面做完，还有一个黑盒和逻辑要做

夜深了，脚冷，就写到这吧。

四.总结

最后认怂去搜这套源码的漏洞，没有百度到这套源码的历史漏洞，运气确实有点霉

如果练习，作者记得有一套老的phpcms，上面什么漏洞都有，可以去搜下