代码审计基础之超全局变量

之前因为在忙别的事情有好久没有分享了，所以最近会把之前欠的量都补上。代码审计第一步其实要读懂代码，这里读懂，不仅指的是这个函数大致是干什么的，这个全局变量代表什么。而是指，当你看到这些代码的细节之后，脑海里所能想象到的安全问题是否全面。打个比方，如下代码：

```
<?php
preg_replace("/[A-Z]*/e","$i"."test",$str);
?>
```

如果你不知道/e修饰符是可以在替换之后执行将替换式作为代码执行(eval)，那么这个时候可能就忽略这段代码了。(注：5.5已废弃/e)

因此，今天我会大致讲一讲代码审计当中涉及到的超全局变量，以及其相关的安全问题。（跟安全相关的常用函数比较多，就不一一介绍了，建议大家自己多翻翻php手册，当然，在我以后的分享当中会逐步提及。多提一嘴，以后会以漏洞分类并辅以实际例子去讲解代码审计，争取各类漏洞都涉及。）

GET

直接从URL获取参数，参数构成一个数组，传入的参数既可以是字符串形式，也可以是数组形式(这是以前常用的绕过技巧)。

```
<?php
var_dump($_GET);
?>
//输入 ?var[1]=1&var2[][]=test
//输出
array (size=2)
  'var' =>
    array (size=1)
      1 => string '1' (length=1)
  'var2' =>
    array (size=1)
      0 =>
        array (size=1)
          0 => string 'test' (length=4)
```

有一些函数在处理参数时候，遇到意料之外的参数类型，比如本来是int，结果传入了array，那么就会产生意料之外的结果，根据进一步的运行有可能可以绕过。

那如果传入多个相同的变量呢?

```
<?php
var_dump($_GET);
?>
//输入 ?var=1&var=2
//输出
array (size=1)
  'var' => string '2' (length=1)
```

当然这只是在php是这样，其他语言可不是，比如asp会将相同变量进行拼接。(注意，如果传入的是相同的数组，比如传入两个var[]，是不会覆盖的，下标会自动递增)

POST

和GET处理相同，只是传入位置发生变化。建议大家安装hackbar，不用每次POST都打开burp(一款火狐的插件)。 注意，传入的content-type必须是application/x-www-form-urlencoded或者multipart/form-data，否则没有办法生成$_POST数组。因此，PHP处理类似xml文件(content-type:text/xml)时，采取的是别的方法。

COOKIE

与GET，POST不同处在于分隔符为分号(;)，其他一样。

上面讲述的三个超全局变量，往往是过滤的最严格的。一般来讲，这三个变量都会做全局防护，比如遍历调用addslashes等。 REQUEST则是包含了GET,POST和COOKIE，将其三个合并到同一数组中。

FILES

传入时对应的key为`<input type="file" name="xx">`的name值。注意，FILES的type值是客户端传入的MIMETYPE，因此是不可信的。所以比较常用的方法是正则检测后缀名然后白名单筛选，同时对幻数以及相关信息做检测。

```
<?php
    var_dump($_FILES);
?>
array (size=1)
  'test' =>
    array (size=5)
      'name' => string '1.txt' (length=5)
      'type' => string 'text/plain' (length=10)
      'tmp_name' => string '/Applications/MAMP/tmp/php/phpsfA3vQ'
(length=36)
      'error' => int 0
      'size' => int 16
```

SESSION

可以将其理解成一个针对每个用户的单独的存储空间，存储用户的一些信息。一般使用cookie作为对应的key值。存储方式也是为数组。

```
<?php
session_start();//加载session变量
$_SESSION[$_COOKIE['PHPSESSID']] = 1;
var_dump($_SESSION);
session_destroy();//销毁session变量
?>
array (size=1)
  '0eded10f2372c5b05fcb053c5fa48bfa' => int 1
```

如果不销毁，则会存储在临时目录当中。查看临时存储目录可通过pathinfo查看。

SERVER

服务器和请求头的信息。注意，这里即使是非法的头信息，也是可以传入的。

```
<?php
var_dump($SERVER);
?>
```

在发送的过程中，我在请求头中添加了`fuck: you`。随后在页面当中我们看到了。

```
'HTTP_FUCK' => string 'you' (length=3)
```

当然，这个其实并不常用，因为服务器一般不会调用这个变量。除非涉及到一些遍历输出SERVER信息或存储SERVER信息时，可能会用到。 但是其他请求头，如UA等，调用较多，而很多时候对这种信息，开发者往往疏与防范，因此有可能成为突破口。

HTTP_RAW_POST_DATA

前面提到，有些时候，因为MIMETYPE的问题，没法使用POST。那么这个时候就可能会去考虑获取HTTP_RAW_POST_DATA。此时得到的就是一个字符串。因此,POST和HTTP_RAW_POST_DATA是完全两个不同的概念，对POST进行了过滤，不代表对HTTP_RAW_POST_DATA进行了过滤。（当然，现在普遍是使用php://input，原因可参考php手册）

GLOBALS

大杂烩，以数组形式包含了其他变量数组。