关于脚本安全这个话题好像永远没完没了，如果你经常到国外的各种各样的 bugtraq 上，你会发现有一半以上都和脚本相关，诸如 SQL injection,XSS,Path Disclosure,Remote commands execution 这样的字眼比比皆是，我们看了之后的用途难道仅仅是抓肉鸡？对于我们想做 web 安全的人来说，最好就是拿来学习，可是万物抓根源，我们要的不是鱼而是渔。在国内，各种各样的 php 程序 1.0 版,2.0 版像雨后春笋一样的冒出来，可是，大家关注的都是一些著名的 cms,论坛,blog 程序，很少的人在对那些不出名的程序做安全检测，对于越来越多的 php 程序员和站长来说，除了依靠服务器的堡垒设置外，php 程序本身的安全多少你总得懂点吧。

有人说你们做 php 安全无非就是搞搞注入和跨站什么什么的，大错特错，如果这样的话，一个 magic_quotes_gpc 或者服务器里的一些安全设置就让我们全没活路了。我今天要说的不是注入，不是跨站，而是存在于 php 程序中的一些安全细节问题。OK!切入正题。

注意一些函数的过滤

有些函数在程序中是经常使用的,像 include(),require(),fopen(),fwrite(),readfile(),unlink(),eval()以及它们的变体函数等等。这些函数都很实用，实用并不代表让你多省心，你还得为它们多费点心。

1、include(),require()和 fopen(),include_once(),require_once()这些都可以远程调用文件，对于它们的危害，google 搜一下你就会很明了，对于所包含调用的变量没过滤好，就可以任意包含文件从而去执行。举个例子，看 print.php

以下为引用的内容：

```
...
if (empty ($bn) ) { //检查是变量$bn 是否为空
include ("$cfg_dir/site_${site}.php"); //把$cfg_dir 这个路径里的 site_${site}.php 包含进来
...
```

不管存不存在$cfg_dir 目录，$site 这个变量你可以很自然的去使用，因为他根本没检查$site 变量啊。可以把变量$site 指定远程文件 http://evil.com/cmd.gif 去调用，也可以是本地的一个文件，你所指定的文件里写上 php 的语句，然后它就去包含执行这个含有 php 语句的文件了。

列出文件目录

甚至可以扩展到包含一些管理员文件，提升权限，典型的像以前 phpwind,bo-blog 的漏洞一样。除了依靠 php.ini 里的 allow_url_fopen 设为 off 禁止远程使用文件和 open_base_dir 禁止使用目录以外的文件外，你还得事先声明好只能包含哪些文件，这里就不多说废话了。

2、fopen(),file(),readfile(),openfile(),等也是该特别留意的地方。函数本身并没什么,它们的作用是去打开文件，可是如果对变量过滤不彻底的话，就会泄露源代码。这样的函数文本论坛里会有很多。

以下为引用的内容：
...
$articlearray=openfile("$dbpath/$fid/$tid.php"); //打开$dbpath/$fid 这个路径的$tid.php 文件
$topic_detail=explode("|",$articlearray[0]); //用分割符|读出帖子的内容
...

很眼熟吧，这是 ofstar 以前版本的 read.php,$fid 和$tid 没有任何过滤，$tid 指定为某个文件提交，就发生了原代码泄露。

http://explame.com/ofstar/read.php?fid=123&tid=../index

$tid 会被加上 php 的后缀，所以直接写 index。这仅仅是个例子，接着看吧。

3、fwrite()和它的变体函数这种漏洞想想都想得出，对于用户提交的字符没过滤的话，写入一段 php 后门又不是不可以。

4、unlink()函数，前段时间，phpwind 里任意删除文件就是利用这个函数，对于判断是否删除的变量没过滤，变量可以指定为任意文件，当然就可以删除任意文件的变量。

5、eval(),preg_replace()函数，它们的作用是执行 php 代码，如果字符串没被经过任何过滤的话，会发生什么呢，我就常看见一些 cms 里面使用，想想，一句话的 php 木马不就是根据 eval()原理制作的吗？

6、对于 system()这些系统函数，你会说在 php.ini 里禁止系统函数，对，这也是好办法，可是象一些程序里需要，那是不是就不用了呢？就像上次我看到的一套很漂亮的 php 相册一样。另外对于 popen(),proc_open(),proc_close()函数你也得特别注意，尽管他们执行命令后并没有直接的输出，但你想这到底对黑客们有没有用呢。再这里 php 提供提供了两个函数，escapeshellarg(),escapeshellcmd(),这两个函数用来对抗系统函数的调用攻击，也就是过滤。

对于危害，来举个例子，我们来看某论坛 prod.php

以下为引用的内容：
07 $doubleApp = isset($argv[1]); //初始化变量$doubleApp
...
14 if($doubleApp) //if 语句
15 {
16 $appDir = $argv[1]; //初始化$appDir
17 system("mkdir $prodDir/$appDir"); //使用系统函数 system 来创建目录$prodDir/$appDir

本来是拿来创建$prodDir/$appDir 目录的，再接着看上去，程序仅仅检测是否存在$argv[1],缺少对$argv[1]的必要过滤，那么你就可以这样
/prod.php?argv[1]=|ls%20-la 或者/prod.php?argv[1]=|cat%20/etc/passwd（分割符 | 在这里是

UNIX 的管道参数，可以执行多条命令。）

到这里，常见的漏洞类型应该知道点了吧。

对于特殊字符的重视

对于特殊字符，有句话叫 All puts is invalid.外国人文章里这句话很常见的。所有输入都是有害的。你永远不要对用户所输入的东西省心，为了对付这些危害，程序员都在忙着过滤大把大把的字符，唯恐漏了什么。而有些程序员呢？好像从没注意过这些问题，从来都是敞开漏洞大门的。不说废话，还是先看看下面这些东西吧。

1、其实程序的漏洞里最关键，最让开发者放心不下的就是带着$符号的美元符号，变量，对于找漏洞的人来说，抓着变量两个字就是一切。就像目录遍历这个 bug，很多邮件程序都存在，开发者考虑的很周全，有的甚至加上了网络硬盘这个东西，好是好，就像 http://mail.com/file.php?id=1&put=list&tid=1&file=./

要是我们把 file 这个变量换成./../甚至更上层呢？目录就这样被遍历了。

2、尖括号"<>"跨站你不会不知道吧，一些搜索栏里，文章，留言，像前段时间 phpwind 附件那里的跨站等等。当然，对于跨站问题，你要过滤的远远不止尖括号。不怕过滤时漏掉什么，而是怕你想不起要去过滤。

3、斜杆和反斜杆：对于/和\的过滤，记得魔力论坛的附件下载处的原代码泄露吗？

attachment.php?id=684&u=3096&extension=gif&attach=.\..\..\..\..\..\includes\config.php&filename=1.gif

对于过滤../ \的问题，像 windows 主机不仅要过滤../还要过滤..\,windows 主机对\会解析为/,这些细节跟 SQL injection 比起来，什么才叫深入呢？

4、对于反引号(``),反引号在 php 中很强大，它可以执行系统命令，就像 system()这些系统函数一样，如果用户的恶意语句被它所执行的话就会危害服务器，我想除了服务器设置的很好以外，对于它们，你还是老老实实的过滤好吧。

5、对于换行符,NULL 字符等等，像"\t,\x0B,\n,\r,\0 这些，这些都是很有用的，像动网以前的上传漏洞就是因为上传中的 NULL(\0)字符引起的，对于这些能随意截断程序流程的字符，你说我们在检测的时候应该有多细心呢？

6、分号(;)和分割符(|)

分号截断程序流程，shell_exec("del ./yourpath/$file"); //使用系统函数 shell_exec 删除文件$file

变量$file 没指定，那么直接写 zizzy.php;del ./yourpath ,这样你的 yourpath 目录也就被 del 了。

分割符(|)是 UNIX 里自带的管道函数，可以连接几条命令来执行。有时候加在过滤不严的系统函数中执行。

逻辑错误

验证不完全和一些逻辑错误在程序里也很容易找到，特别是现在的程序员，只顾深入的学习，而对于逻辑错误等等这样的安全意识都没有培养的意识，其实这是是靠自己去培养，而不是等着人来报告 bug 给你。对于逻辑错误的判断，我们只能说，多练练吧，经验才是最重要的。

1、对于登陆验证的问题。举个例子：我们看某论坛的 admin.php 片断

它这里 username 和 password 好像不对劲吧，存在管理员的 username 和 password 就直接通过验证，那就意味着没有用户名，没密码也行吧。我们提交

以下为引用的内 GET /bbs/admin/index.php?page=general HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Maxthon)
Host: 127.0.0.1
Connection: Keep-Alive
Cookie: username='or isnull(1/0) AND level=3/*; password=;

这是我们伪造的一个数据包(你问我咋伪造地？抓包再修改呗),我们使用 GET 来提交数据，原理也就是在 cookie 这里构造欺骗语句。

接着，整个 SQL 语句就成这样

Select * FROM users Where username=''or isnull(1/0) AND level=3/*' AND password=''

这里仅仅用一个'or'='or'的原理，就把 username 和 password 的检测给绕开了，而 level=3 则是伪造的等级。从而就饶过了检测，进入了管理后台。

对于后台的验证不能这么马虎，两行代码就算完事，你还得从 SESSION(会话),Cookie 这些地方来增强验证。

2、上传漏洞

有次我看到在一个程序 config.php 里对上传文件类型限制是这样的

$IllegalExtentions = array('exe','asp','php','php3','bat','cgi','pl','com','vbs','reg','pcd','pif','scr','bas','inf','vb','vbe','wsc','wsf','wsh'); //

对于上传文件的限制，只允许上传exe,asp,php,php3,bat,cgi,pl,com,vbs,reg,pcd,pif,scr,bas,inf,vb,vbe,wsc,wsf,ws'这些文件。

规定不许用户上传什么什么文件，其它都可以上传，这种逻辑好不好呢？如果我上传.inc, ,.php4 .phtml, .html, .pwml 这样的类型呢？为什么你不把这种逻辑思维改为规定用户除了这几种文件能传，其它的统统不允许上传。就像这样，数组改成逆向的思维。

$IllegalExtentions = array('rar','gif','jpg','bmp','pdf'）//只能上传 rar,gif,jpg,bmp,pdf 几种格式

其实这个跟你们上传 cer,asa 是一个道理。

3、典型的逻辑错误

在一些 cms（整站程序）中随便注册个用户，你会发现修改资料的地方不要求输入原来的密码，只通过判断用户 id 或者 email，你把网页保存到本地，把 id 或 email 改成管理员的，action 改为修改提交地址，提交你就成了管理员。解决办法不太难，只要我们增加密码验证，增强那个 mysql 的 update 语句的过滤也就 ok 了。

这些我们也没办法，多数程序员对于安全根本不去在意，本来一个人可以去做的事，为什么偏偏要分出搞 web 安全的和 web 开发两种人呢？

长度问题

别以为找漏洞的就是为了拿个管理员密码或者 webshell,也有些不安分的人，也就是 DDOSer(拒绝服务攻击者),他们的花样很多，但对于程序员来说，关键就在过滤。我所说的长度问题，不仅仅是个字符的长度，也包括时间的长度，你一定见过有人写个脚本，一下就注册成千上万的用户，或者纯粹的写垃圾数据把数据库拖死。这个时候，限制数据提交时间和验证码就起作用了。不过要真的遇到狠毒的人，一个变量的过滤问题就可以把网站搞瘫痪，这比用什么网络僵尸那些软件来得更快。

不大不小的问题

1、绝对路径的泄露

这个问题可真是不大不小，很多程序都有，这也算安全的一部分。至少你玩注入 loadfile() 需要吧。当然，这时的 php.ini 中的 display_errors 也可以起作用了。

2、对后台的验证

不要说不信，我就曾看到一些程序这样，你去测试，注册个用户，提交管理员编辑用户的 URL, 比如 admin_member.php?action=edit&id=55&level=4&username=zizzy& amp;power=1 这样相应的添加管理员的 URL,你会发现几乎没验证，直接成功了。所以，对于后台的检测，也很有必要，就像刚出的 Discuz 的那个漏洞。

过滤问题不知不觉就说了那么多，写了好多处该过滤的提醒，现在也该说说了如何进行过滤。

1、在用户输入任何数据，也就是提交变量进数据库时，我们必须使用 addslashes()进行过滤，像我们的注入问题，一个 addslashes()也就搞定了。其实在涉及到变量取值时，intval()函数对字符串的过滤也是个不错的选择。

2、在 php.ini 中开启 magic_quotes_gpc 和 magic_quotes_runtime。magic_quotes_gpc 可以把 get,post,cookie 里的引号变为斜杠 magic_quotes_runtime 对于进出数据库的数据可以起到格式话的作用。其实，早在以前注入很疯狂时，这个参数就很流行了。

3、在使用系统函数时，必须使用 escapeshellarg(),escapeshellcmd()参数去过滤，这样你也就可以放心的使用系统函数。

4、对于跨站，strip_tags(),htmlspecialchars()两个参数都不错，对于用户提交的的带有 html 和 php 的标记都将进行转换。比如尖括号"<"就将转化为 "<"这样无害的字符。

5、对于相关函数的过滤，就像先前的 include(),unlink,fopen()等等，只要你把你所要执行操作的变量指定好或者对相关字符过滤严密，我想这样也就无懈可击了。

服务器安全设置

谈服务器安全设置，我觉得很不实际的，我们大多数人都用虚拟主机，对于 php.ini 怎么设，那个只有网管自己看着办了。不过我还是说下，

1、设置"safe_mode"为"on"

这对于广大空间商来说是一个伟大的选项，它能极大地改进 PHP 的安全性。

2、禁止"open_basedir" ，这个选项可以禁止指定目录之外的文件操作，还能有效地消除本地文件或者是远程文件被 include()等函数的调用攻击。

3、expose_php 设为 off,这样 php 不会在 http 文件头中泄露信息。

4、设置"allow_url_fopen"为"off" 这个选项可以禁止远程文件功能，极力推荐

5、"log_errors"为"on" 错误日至得带上吧

6、对于"display_errors，register_globals"两项要视情况而定了，display_errors 太消极了，错误全关，想调试脚本都不行。至于 register_globals(全局变量)把它开起来，关了会很麻烦，现在大多数程序没它支持就别想用了。

这些是最必要的设置。关于 php 服务器更高的安全设置是门学问，也就不在本文探讨范围内了。

这篇文章到这里就要结束了，也许你会说，你说的这些都是对开源的程序才有用，对那些 zend 加密的程序不就没办法可使了吗？其实，对安全来说，固其根本才是重要的吧，你再怎么加密难道逃得过黑盒测试？总有一天会被发现的吧。

限于篇幅也就到这里了,我们对于 php 程序安全也有了初步的探索。为广大读者朋友考虑，举的例子也算是很容易去理解地，当然前提是你得会点 php，要不然又是看天书了(哇，不要看到这里才问我啥是 php?)。整篇文章并不是黑客教学，而是为那些想在 php 安全上发展的初学者和 php 程序员写的。如果以后你又听到谁谁又发现什么漏洞了，再怎么变也就是那些基本的东西而已。我希望本文能为你们开阔下思路，更好的发展下去。嗜酒成痴剑亦狂，重燃你的 php 安全之火，带着对 php 的执着上路吧。