Yii防注入攻击笔记

网站表单有注入漏洞须对所有用户输入的内容进行个过滤和检查，可以使用正则表达式或 者直接输入字符判断，大部分是只允许输入字母和数字的，其它字符度不允许；对于内容复杂表单的内容，应该对html和script的符号进行转义替换：尤 其是<,>,',"",&这几个符号

这里有个转义对照表：  
http://blog.csdn.net/xinzhu1990/article/details/7032301

单引号替换成两个

直接将客户端传过来的的参数值直接组成字符串sql，而不是使用statment填充参数的方式，也没有进行字符串处理和过滤，这些地方都有注入的漏洞，尤其是没有对单引号过滤

php里有转义的函数可以直接调用addslashes(), stripslashes(), htmlspecialchars(), htmlentities(), nl2br() 等函数.

addslashes(), stripslashes() 一般是入数据库和出库的时候使用,以免变量中存储类似引号这些关键词

htmlspecialchars() 函数只用来转义少量HTML, &,双引号,大于号和小于号.并不会全部转换成 HTML 所定的 ASCII 转换

htmlentities()这个是全部转换html实体，和htmlspecialchars()区别在于，这个函数是转义全部的字符，而htmlspecialchars()仅仅转义上面限定的5个特殊字符！

htmlspecialchars（） 转义特别的字符为HTML实体；

   '&' (ampersand) becomes '&amp;'

   '"' (double quote) becomes '&quot;' when ENT\_NOQUOTES is not set.

   ''' (single quote) becomes '&#039;' only when ENT\_QUOTES is set.

   '<' (less than) becomes '&lt;'

   '>' (greater than) becomes '&gt;'

当显示文本时，调用 CHtml::encode() 对其中的HTML编码。这可以防止XSS注入.

使用cookies时使用Yii内置的cookies组件来进行cookie操作，不要使用$\_COOKIES。

这样可以防止cookie攻击

$cookie=Yii::app()->request->cookies[$name];

$value=$cookie->value;

Yii::app()->request->cookies[$name]=$cookie;

对于是知道类型的post，get参数，得先转换类型，尤其是知道是整型的

防范XSS攻击的最重要的措施之一就是：在显示用户输入的内容之前进行内容检查。

Yii提供了一个很有用的组件CHtmlPurifier可以将通过有效的审查、安全和白名单功能来把所审核的内容中的所有的恶意代码清除掉，并且确保过滤之后的内容过滤符合标准。

CHtmlPurifier组件可以作为一个widget或者filter来使用。 当作为一个widget来使用的时候，CHtmlPurifier可以对在视图中显示的内容进行安全过滤。

<?php $this->beginWidget('CHtmlPurifier'); ?>

//...这里显示用户输入的内容...

<?php $this->endWidget(); ?>

跨 站请求伪造(简称CSRF)攻击，即攻击者在用户浏览器在访问恶意网站的时候，让用户的浏览器向一个受信任的网站发起攻击者指定的请求。 Yii实现了一个CSRF防范机制，用来帮助防范基于POST的攻击。默认情况下，CSRF防范是禁用的。如果你要启用它，可以编辑应用配置 中的组件中的CHttpRequest部分。

return array(

   'components'=>array(

       'request'=>array(

           'enableCsrfValidation'=>true,

       ),

   ),

);

要显示一个表单，请使用CHtml::form而不要自己写HTML代码。

Yii实现了一个cookie验证机制，可以防止cookie被修改。启用之后可以对cookie的值进行HMAC检查。

return array(

   'components'=>array(

       'request'=>array(

           'enableCookieValidation'=>true,

       ),

   ),

);

直接使用AR对象赋值操作数据库的Yii会自动做防注入处理。

不使用AR对象赋值方式，使用直接sql的，也应该使用这种参数赋值的方式，可以防止注入攻击：

$db = Yii::app()->db;

$sql = "select \* from table where userid=:userid and date between :date\_start and :date\_end"

$results = $db->createCommand($sql)->query(array(

 ':userid' => 115,':date\_start'=>'2009-12-1',':date\_end'=>'2009-12-31',

));

...