验证码的三个常见漏洞和修复方法

主要介绍了验证码的三个常见漏洞和修复方法,本文讲解了把验证码存储在Cookie中、没有进行非空判断、没有及时销毁验证码三个常见问题和解决方法,需要的朋友可以参考下

把验证码存储在Cookie中

一般来说，我们会把验证码的值用Session存储起来，通过对比用户提交的验证码和Session中的验证码，就可以知道输入是否正确。由于Session会占用服务器资源，我曾经想过是否可以把验证码的值加密后存储在Cookie中。不过事实证明，这只是异想天开罢了。

假设验证码的值是a，通过sha1加密后得到的值为b = sha1(a)，并且把b存储在Cookie中。而用户提交的验证码值为c，通过判断sha1(c)是否与b相等，可以知道输入的验证码是否正确。然而，Cookie是受客户端控制的。如果用户事先通过肉眼看到验证码的值是a，又从Cookie中得知此时的加密值为b，那么，他只要在提交前把Cookie的值修改为b，提交的验证码值为a，就可以永远通过验证。

没有进行非空判断

这种情况可以直接用代码来说明：

复制代码 代码如下:

```
if (Request["captcha"] == Session["captcha"] as string)
{
    // 验证通过，继续操作
}
```

假设用户绕过了系统提供的表单直接提交数据，此时验证码还没生成，Session["captcha"]为空。用户不提交验证码时，Request["captcha"]也为空。于是，验证通过了。

要解决这个问题，其实只要加个非空判断就可以了：

复制代码 代码如下:

```
if (!String.IsNullOrEmpty(Request["captcha"]) &&
    Request["captcha"] == Session["captcha"] as string)
{
    // 验证通过，继续操作
}
```

没有及时销毁验证码

使用验证码要遵循一个原则，在一次比对之后，无论用户输入正确与否，都要立刻将验证码销毁。

如果不这样做，就可以出现以下情况：

假设用户输入错误，且验证码没有重新生成，那么他就可以一直尝试，直到正确为止。虽然机器对图片的一次性识别率比较低，但是，如果同一张图片你给它无限次机会的话，它还是可以识别出来的。
假设用户输入成功，且验证码没有销毁，那么在Session过期之前，他就可以一直用这个验证码通过验证。