欲行大善之事，必谙大恶之道。

三：正餐：XSS

OWASP TOP10

XSS排名第二第三

> Cross Site Scripting 跨站脚本攻击

CSS更常表示层叠样式表（Cascading Style Sheets，CSS）的缩写。

因此，将跨站脚本攻击缩写为XSS。

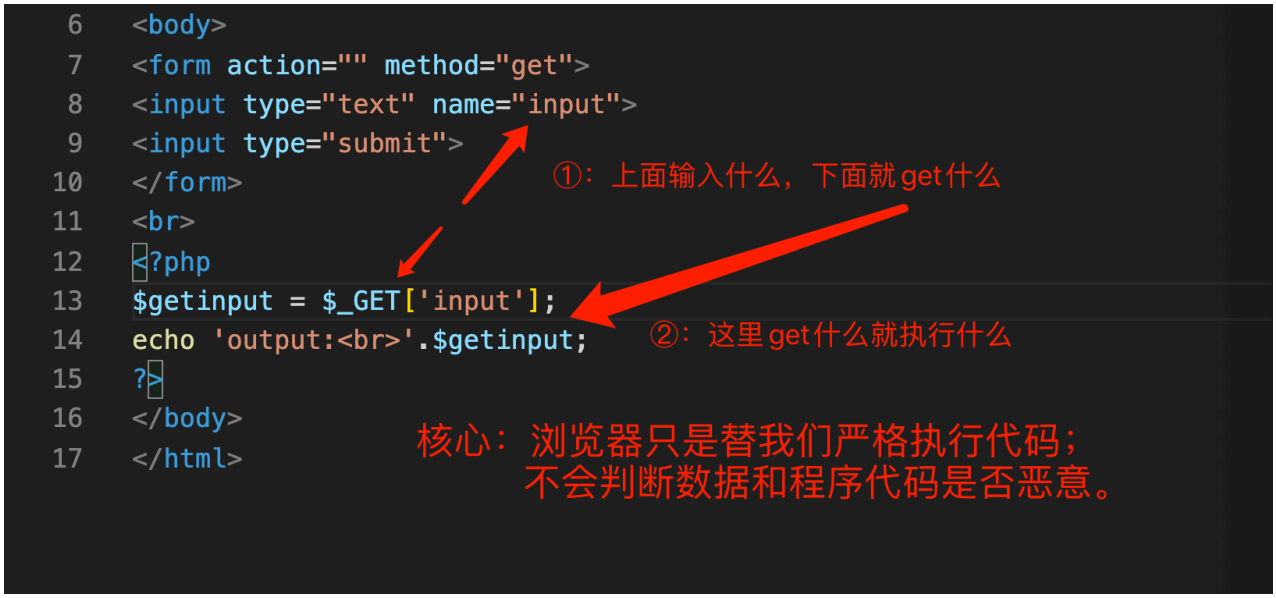
攻击演示：通过一个input，让漏洞网站执行我们的代码。

>>> 为什么能?

>>> 怎么做?

>>> 如何预防?

3个问题其实是1个问题：知道XSS的攻击原理，也就能知道怎么攻击，知道怎么攻击也就知道怎么防御。



核心：浏览器只会按照代码严格执行，不会判断数据和程序代码是否恶意。

> Web浏览器本身的设计是不安全的。

浏览器包含了解析和执行JavaScript等脚本语言的能力，这些语言可用来创建各种格式丰富的功能，而浏览器只会执行，**不会判断数据和程序代码是否恶意。**

> 网站交互增加

随着Web 2.0的流行，网站上**交互功能越来越丰富**。Web 2.0鼓励信息分享与交互，这样用户就有了更多的机会去查看和修改他人的信息，比如通过论坛、Blog 或社交网络，于是黑客也就有了更广阔的空间发动XSS攻击。

如果面试被问到，或者谈起XSS，你需要知道的信息：

XSS攻击大致上分为两类：

- 反射型XSS，又称非持久型XSS，
- 储存型XSS，也就是持久型XSS。

①

反射型跨站脚本（Reflected Cross-site Scripting）也称作非持久型、参数型跨站脚本。这种类型的跨站脚本是最常见，也是使用最广的一种，主要用于将恶意脚本附加到URL地址的参数中，例如：

```
http://www.test.com/search.php?key="><script>alert("XSS")</script>
```

反射型XSS的利用一般是攻击者通过特定手法（比如利用电子邮件），诱使用户去访问一个包含恶意代码的URL，当受害者单击这些专门设计的链接的时候，恶意JavaScript代码会直接在受害者主机上的浏览器执行。它的特点是只在用户单击时触发，而且**只执行一次，非持久化**，所以称为反射型跨站式脚本。

此类 XSS 通常出现在网站的**搜索栏、用户登入口**等地方，常用来窃取客户端 Cookies 或进行钓鱼欺骗。

其实就是：**输入输出交互**的地方。

```
"因为需要用户点击，所以黑客不会这么直接的暴露目的，而是会通过编码，隐藏恶意URL："

"http://www.test.com/search.php?key="><script>alert("XSS")</script>""
把上面的script标签编码隐藏：
"http://www.test.com/search?
q=%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%22%48%69%22%29%3b%3c%2f%73%63%72%69%70%74%
3e"
```

如果把这里的alert("XSS")，换成alert(document.cookie)，就能获取到cookie。

②

持久型XSS**不需要用户去单击URL进行触发**，所以它的危害比反射型XSS大，黑客可以利用它渗透网站、挂马、钓鱼.....

持久型 XSS 一般出现在网站的留言、评论、博客日志等交互处，恶意脚本被存储到客户端或者服务器的数据库中，当其他用户浏览该网页时，站点即从数据库中读取恶意用户存入的非法数据，然后显示在页面中，即在受害者主机上的浏览器执行恶意代码。

留言系统漏洞：

```
<textarea><script>alert(/XSS/)</script><textarea>
```

这个XSS在管理员查看留言时才触发。这种XSS场景可以发挥的空间很大，因为它**攻击的对象是后台管理员**，攻击者能利用此种 XSS 劫持管理员会话而执行任意操作，比如修改密码、添加新闻、备份数据库等。

因为留言系统需要管理员审核，所以这个应该是form获取表单数据的时候，**程序没有过滤任何有害字符**，直接把留言信息写进数据库中，从而导致一个持久型XSS的产生。

攻击者把XSS代码写进留言信息中，当管理员登录后台并查看留言时，便会触发XSS漏洞。由于该XSS是在后台触发的，所以攻击的对象是后台管理员。通过注入JavaScript代码，**攻击者便能劫持管理员会话执行某些操作，从而达到提升权限的目的。** 比如，攻击者想利用XSS添加一个后台管理员账号，他只需要截取添加管理员账号时的HTTP请求信息，然后使用XMLHTTP对象在后台发送一个HTTP请求即可。 截获往返于网络之间的HTTP通信数据，可以采用HTTP抓包工具或Web代理软件，然后就可以**在后台悄悄添加一个管理员账户等等操作。**

更严重的是，利用此类 XSS 能够轻易编写危害性更大的 **XSS 蠕虫**，跨站蠕虫是使用Ajax/JavaScript脚本语言编写的蠕虫病毒，能够在网站中实现病毒的几何数级传播，其感染速度和攻击效果都非常可怕。XSS蠕虫会直接影响到网站的所有用户，也就是一个地方出现XSS漏洞，同站点下的所有用户都可能被攻击!

最后，XSS可能会给网站和用户带来的危害简单概括如下：

- （1）网络钓鱼，包括**盗取各类用户账号**；
- （2）**窃取用户cookies资料**，从而获取用户隐私信息，或利用用户身份进一步对网站执行操作；
- （3）**劫持用户（浏览器）Session会话**，从而执行任意操作，例如进行非法转账、强制发表日志、发送电子邮件等；
- （4）强制弹出广告页面、刷流量等；
- （5）网页挂马；
- （6）进行恶意操作，例如任意**篡改页面信息**、删除文章等；
- （7）进行大量的客户端攻击，如**DDoS攻击**；
- （8）**获取客户端信息**，例如用户的浏览历史、真实IP、开放端口等；
- （9）**控制受害者机器**向其他网站发起攻击；
- （10）结合其他漏洞，如CSRF漏洞，实施进一步作恶；
- （11）**提升用户权限**，包括进一步渗透网站；
- （12）**传播跨站脚本蠕虫**等。

防范手段

都说知己知彼方能百战不殆，知道了xss攻击的原理那么防御的方法也就显而易见了。

- 首先是**过滤**。对诸如 `<script>`、`<img>`、`<a>` 等标签进行过滤。
- 其次是**编码**。像一些常见的符号，如<>在输入的时候要对其进行转换编码，这样做浏览器是不会对该标签进行解释执行的，同时也不影响显示效果。
- 最后是**限制**。通过以上的案例我们不难发现xss攻击要能达成往往需要较长的字符串，因此对于一些可以预期的输入可以通过限制长度强制截断来进行防御。

四： 甜点：所以，我们为什么要学XSS?

- **代码安全很重要；**
- 通过XSS等攻击**原理**，我们能更深入细致的了解浏览器、网络协议、解释器、代码的原理。

比如：

上面反射型跨站脚本隐藏恶意url一带而过。但是真正稍微有点安全意识的程序员和公司，肯定不会这么简单一个编码就被骗过去了。

而且，就算你写了过滤器，攻击者也还是可以有针对性的绕过过滤——**就比谁对底层了解的更深、更细。**

①

比如你写了一个**过滤函数，过滤了<>、JavaScript。**黑客还可以用JavaScript的语言特性，就是**通过tab，分割JavaScript。**

比如，< img src="javas cript:alert(/xss/)" width=100>

原理：

下面的代码，虽然语句中间有一个换行符，但变量的赋值完全成功：`var a = 'hello world';` `alert(a);` **引擎没有把换行符解释为语句的终止符**，因为到换行处并不是一个完整的语句，JavaScript会继续处理发现的内容，**直到遇到一个分号或发现语句完整为止。**

②

再比如，因为**HTML中属性值本身支持ASCII码形式。**所以，还可以把`< img src="javas cript:alert(/xss/)" width=100>`专成`< img src="javascrip&#116&#58alert(/XSS/);" >`

③

利用各种事件触发XSS：

除了onclick事件，

利用其他事件，也能触发自己的xss代码：

`< img src="#" onerror=alert(/xss/)>` **onerror**是IMG标记的一个事件，只要页面中发生错误，该事件立即被激活。在这个示例中，当浏览器解释IMG标记的时候，会加载src属性引用的图片地址，倘若该图片不存在就会触发onerror事件。

④

利用css作为载体，执行xss跨站脚本。

```
<style> body {background-image: url("javascript:alert('XSS')");} </style>
```

CSS属性可以是元素固有的属性，也可以是自定义属性。如果CSS属性后面为一段JavaScript表达式，则其值等于JavaScript表达式计算的结果。

当你以为自己设置了严密的过滤规则而万事大吉之时，却可能因为不够了解浏览器解析，而功亏一篑。

⑤

比如：`< img src="java/*/*javascript*/script/*javascript*/*/script:alert();" >`

样式表中的//**会被浏览器忽略，因此可以运用**//来注释字符。目前**大多数过滤系统都采用黑名单式的过滤，用户可以结合使用注释字符干扰和欺骗过滤器。**

再比如，我们可以通过攻击来更深入的了解cookie和session会话：

cookie虽然可以加密，但窃取Cookie的人不需要知道这些字符串的含义，只要把Cookie信息向服务器提交并通过验证后，他们就可以冒充受害人的身份登录网站，这种行为一般叫做**Cookie欺骗或Cookie会话攻击。**

Session 的中文意思是会话，其实就是访问者从到达特定主页到离开的那段时间，在这个过程中，每个访问者都会得到一个单独的Session。**Session是基于访问的进程，记录了一个访问的开始到结束，**当浏览器或进程关闭之后，Session也就“消失”了。 在Session机制中，客户端和服务端通过标识符来识别用户身份和维持会话，但这个标识符也会有被其他人利用的可能。

Session和Cookie的最大区别在于：Session是保存在服务端的内存里面，而Cookie保存于浏览器或客户端文件里面。