**Part1 前言**

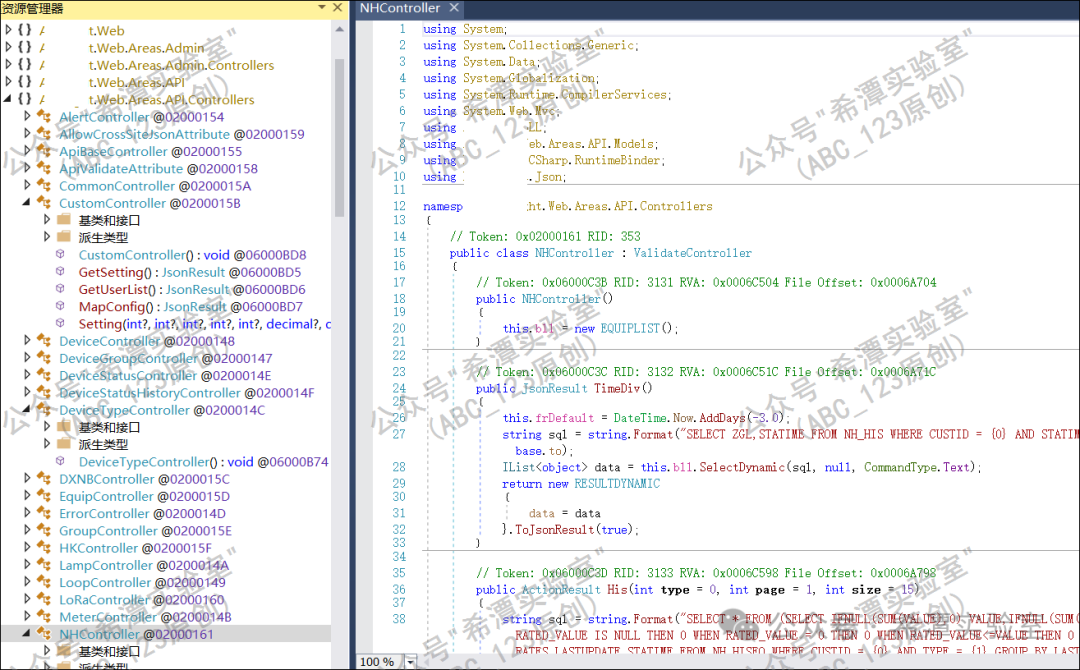
**大家好，我是ABC\_123**。最近在复习整理之前自己审过的一些代码，找到了几年前红队评估项目中审计的一个.Net代码的案例，我平时很少看.NET的代码，于是就拿出代码重新学习总结了一下，整个审计过程比较简单。

**Part2 研究过程**

大致过程如下：目标网站的资产存在一个XXX系统，通过信息收集找到了该系统的供应商。对供应商官网的子域名进行了收集，其中一个子域名搭建的这套系统的测试站点。经过目录扫描，测试站点存在一个网站的备份文件可以直接下载，于是得到了该系统的源代码。

* **反编译工具的使用**

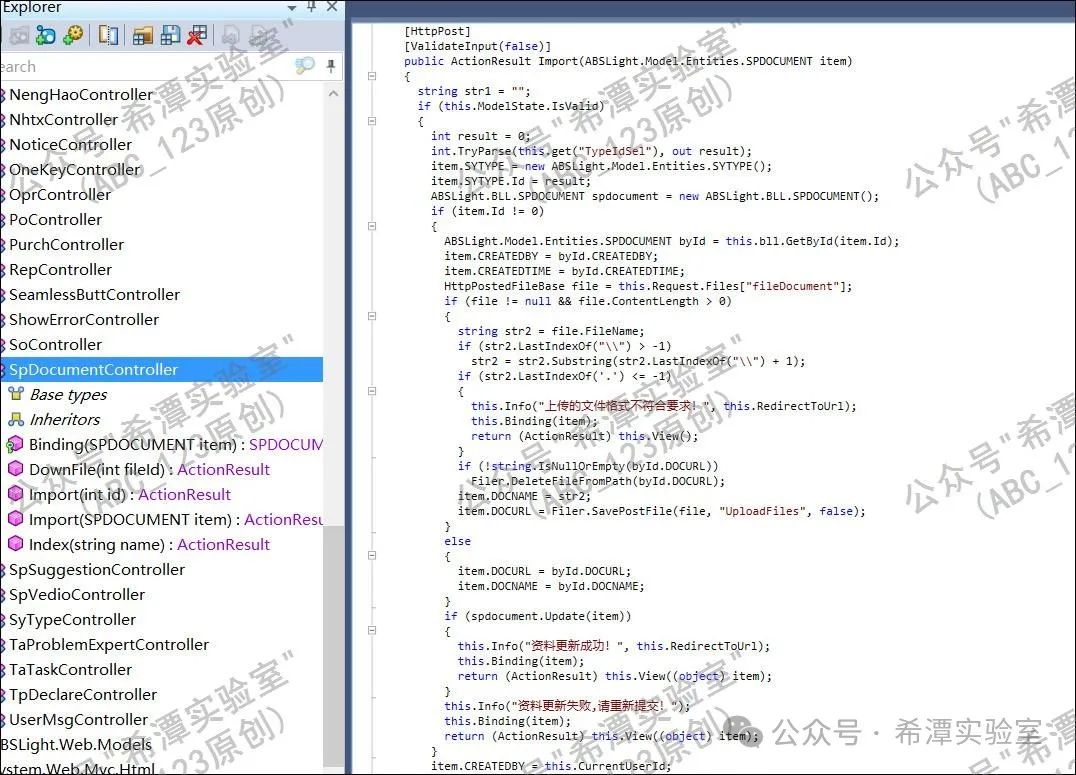
可选择的.NET反编译工具主要有四个：dnSpy（于2020年左右停止更新）、ILSpy、dotPeek（JetBrains发布）、dnSpyx（非官方更新版）。选择一个自己用的顺手的反编译工具对\*.Web.dll文件进行反编译，并导出所有反编译后的代码。我也将反编译结果对比了一下，以下是dnSpyx的反编译结果：



以下是ILSpy的反编译结果：

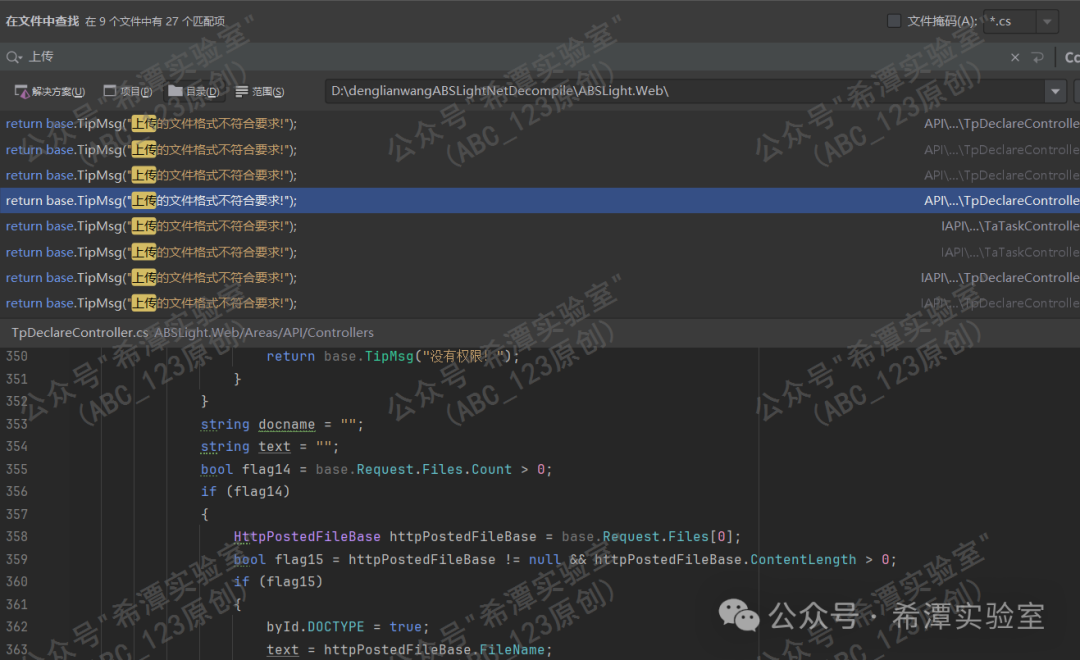


以下是dotpeek的反编译结果：



* **审计上传漏洞**

将反编译的代码导入Rider64（IntelliJ IDEA公司出品的），使用IDE进行代码审计、搜索关键字更方便一些。全局搜索”**上传**”关键字，挨个查看有可能实现了上传功能的.Net类。



最终发现了如下代码，有可能存在上传漏洞，**[HttpPost]**提示该请求只支持POST请求形式：

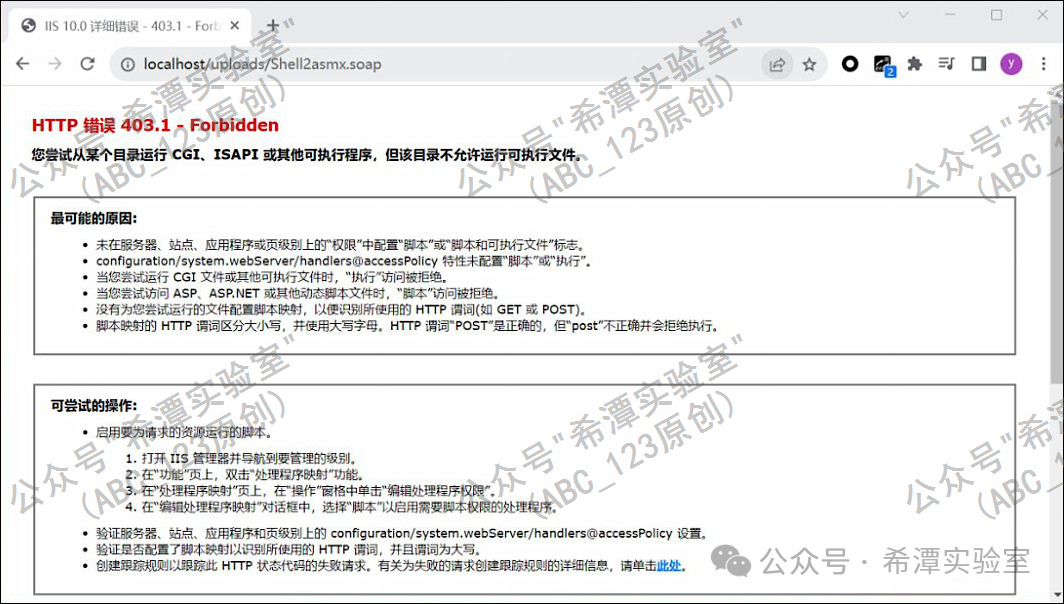


构造上传数据包如下，最终通过**; filename;;;;**的方式结合**.cer**扩展名绕过了WAF。

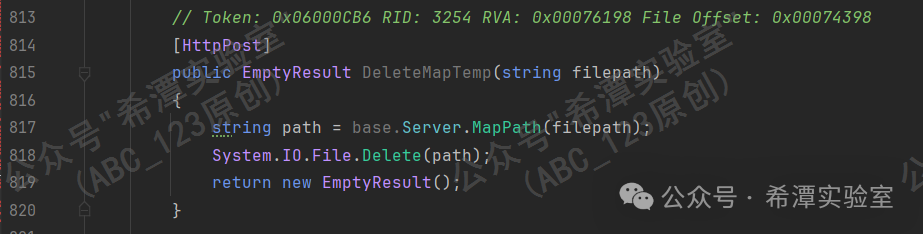


* **禁止脚本执行问题解决**

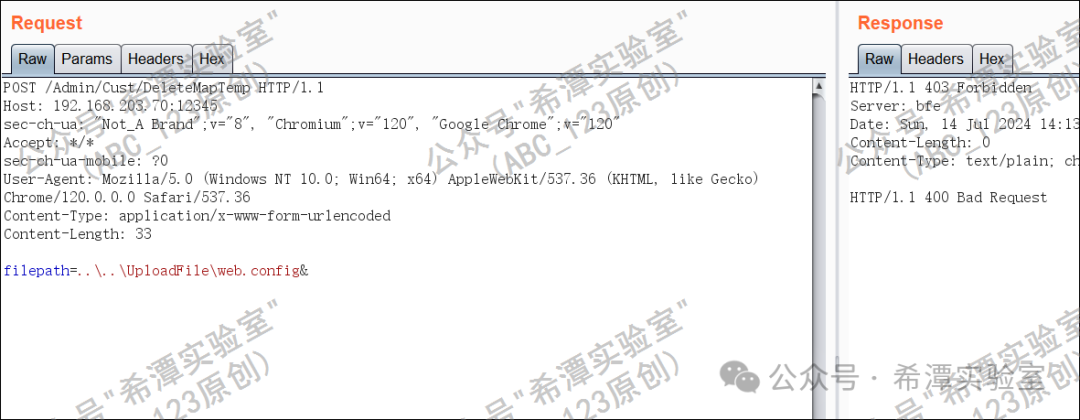
等到第二天起来的时候，访问webshell发现提示“HTTP 错误403.1 - Forbidden”，该目录不允许运行可执行文件，也就意味着webshell在当前目录无法执行。经过判断，猜测蓝队防守方做了加固，在webshell所在文件夹下放置了**web.config**文件，更改了默认配置，使脚本在当前目录下不能执行。



接下来想了解决办法，我们可以找一处任意文件删除漏洞，将web.config文件删掉，从而使web.config文件失去作用，使Webshell可以继续执行。搜索**delete**关键字挨个查看，找到了一处任意文件删除漏洞，关键代码是**System.IO.File.Delete(path)**，[HttpPost]提示该请求只支持POST形式：



最终按照代码拼接成如下数据包，通过..\..\跳转，成功将web.config文件删掉：



终于webshell又可以正常执行了，后续我们在当前目录放置了一个可以允许脚本执行的web.config文件，防止蓝队防守发现，在内网也做了新的权限维持。

* **任意文件读取漏洞**

如果担心任意文件删除漏洞会影响业务，可以提前审计一个任意文件读取漏洞，查看一下web.config文件的内容，判断一下删除该文件是否会对业务产生影响，或者是提前与甲方沟通好。



**Part3 总结**

**1.**  任意文件删除漏洞的审计也是有价值的，在本次案例中得到了印证。

**2.**  如果担心删除文件会影响业务，可以审计一个文件读取漏洞，查看一下文件内容，具体评估一下风险。