0x00 前言

如有技术交流或渗透测试/代码审计/红队方向培训/红蓝对抗评估需求的朋友

欢迎联系QQ/VX-547006660

0x01 起因

某项目靶标，是一个人员管理系统，通过webpack暴露的接口

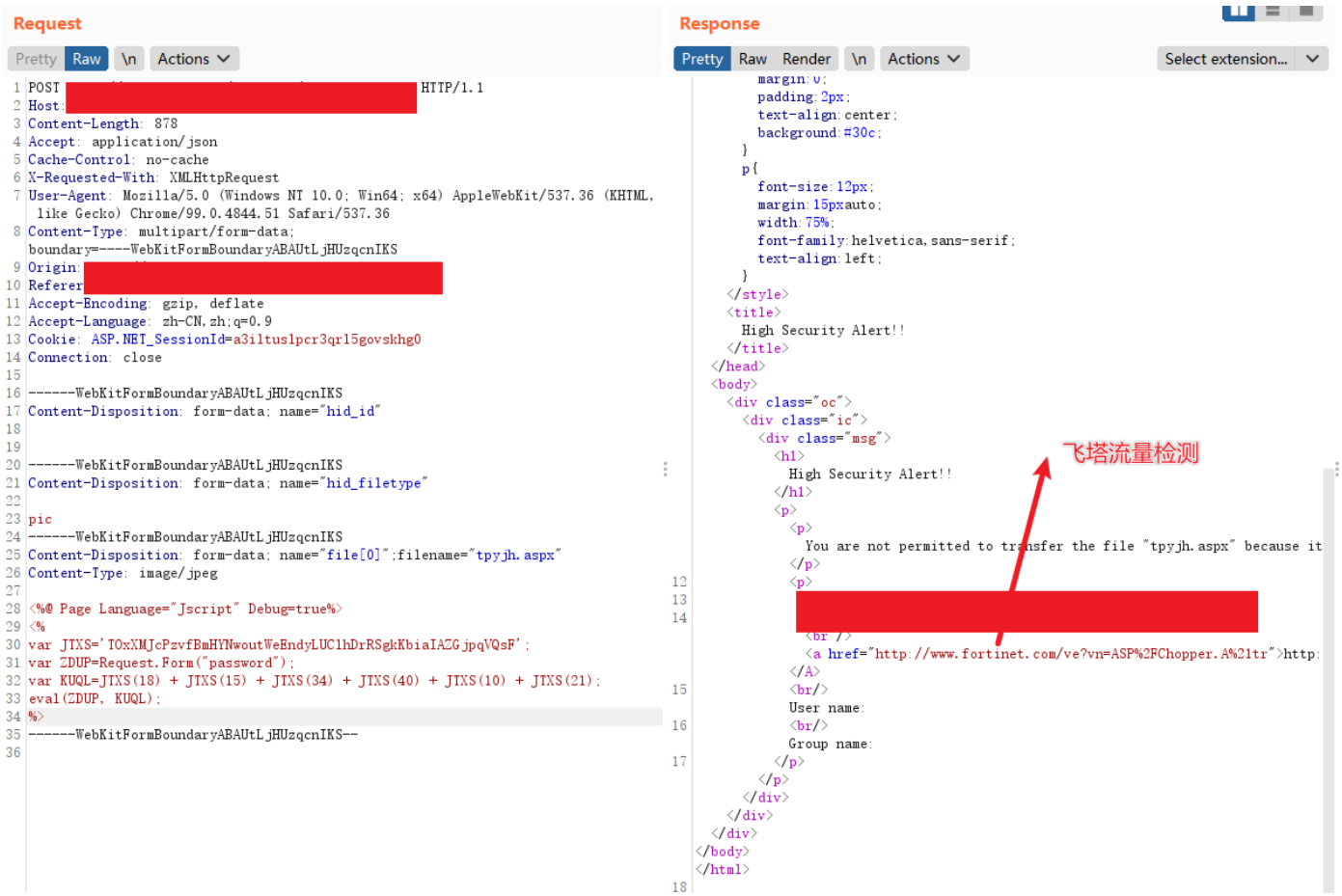
我们成功找到了一个未鉴权的密码修改接口，通过fuzz

我们获取到了该接口的参数username与password，并成功修改了admin账号密码进入后台

在一个任意上传的后台功能点，我们遇到了飞塔WAF+流量检测设备，于是便有了本文

0x02 摸清检测规则

众所周知，飞塔公司的WAF+流量检测设备还是比较牛逼的，我直接尝试上传一个免杀webshell，毫无疑问直接GG了



经过测试，上传无害内容1111但文件名为XXX.aspx也会被拦截

发现飞塔对form表单的文件名和文件内容都有检测，我们必须得把这俩检测都安排掉才能成功上传

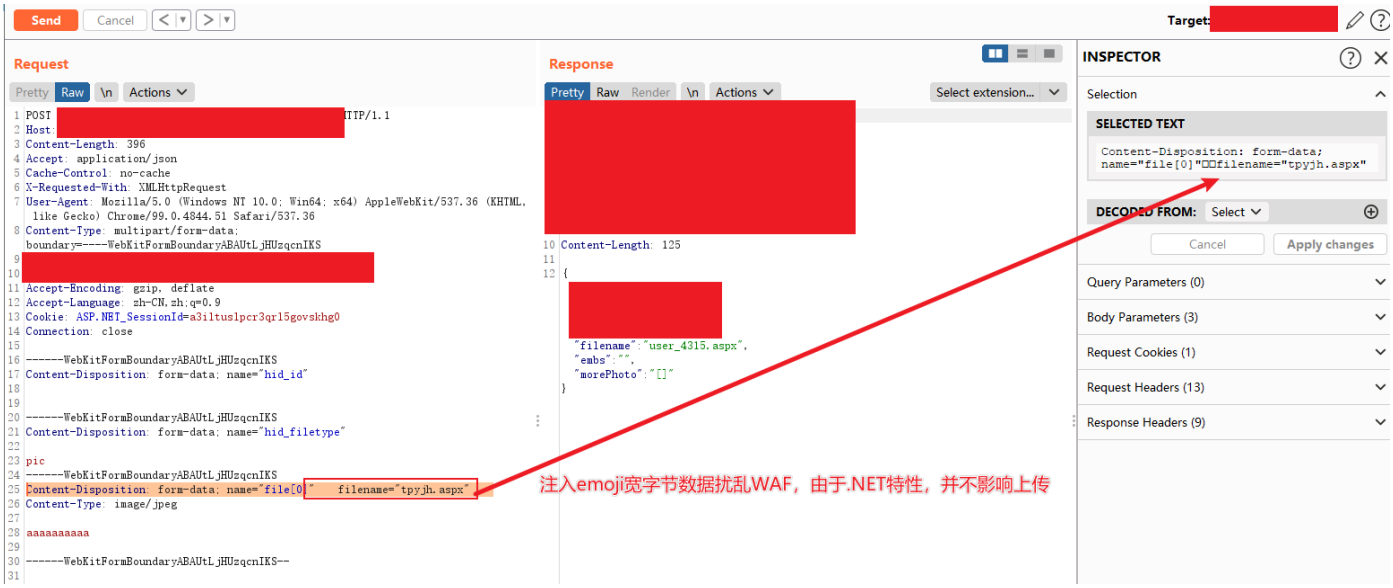
下面简述一下对抗思路

0x03 文件名检测——根据.NET上传匹配文件名特性绕WAF(骗)

.NET中常用**context.Request.Files**处理文件上传表单

其在匹配上传文件名时只匹配**Content-Disposition:**后的**filename=xxxx**，这就给了我们很大的绕WAF操作空间，可以直接在Content-Disposition中注入脏数据来扰乱WAF的检测

这里我去掉分号，并使用emoji宽字节数据扰乱waf



如图，使用无害数据成功上传aspx文件

0x04 文件内容检测——双图片夹恶意代码(偷袭)

前面既然直接传免杀Shell不行，我又测试了加入图片头，尝试绕过检测

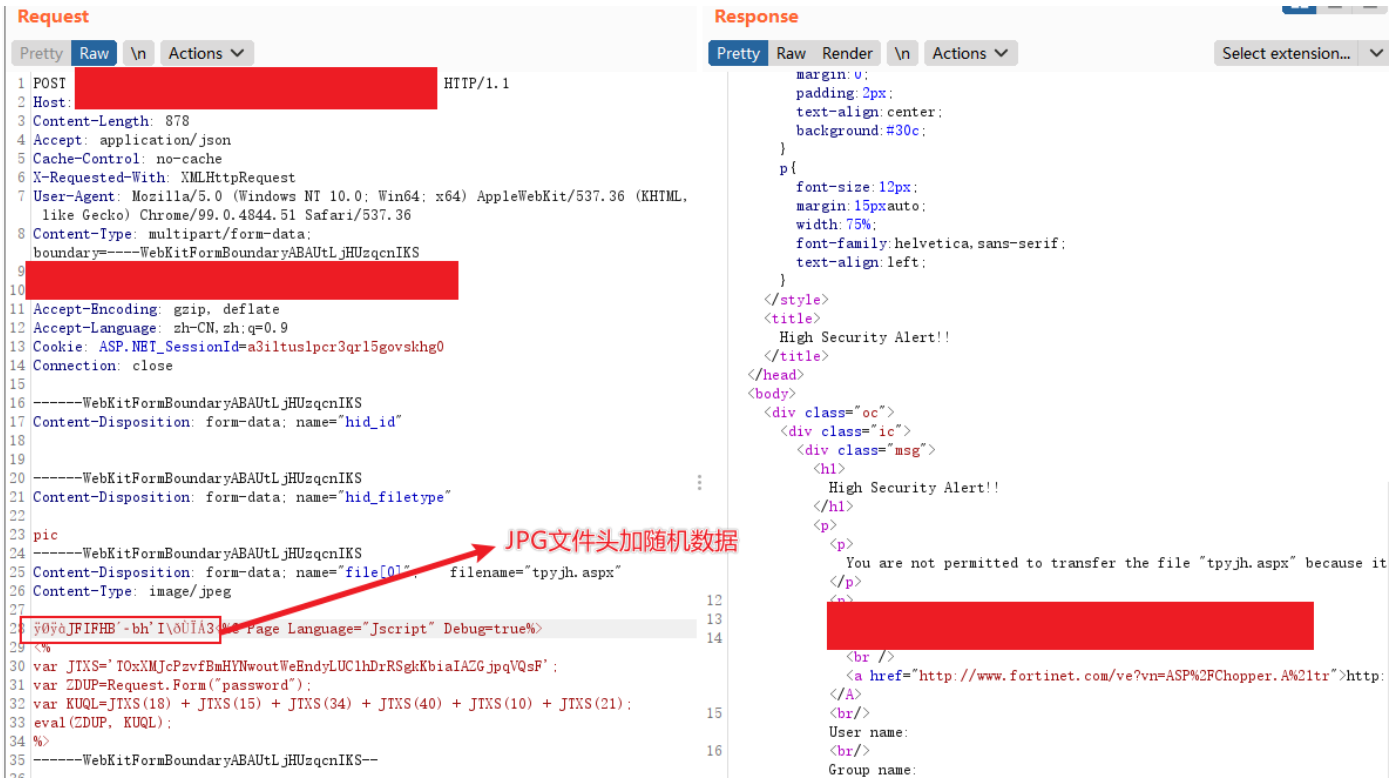
例如：

```
JPEG (jpg)，文件头：FFD8FF

PNG (png)，文件头：89504E47

GIF (gif)，文件头：47494638
```

发现并没有什么卵用，都GG了

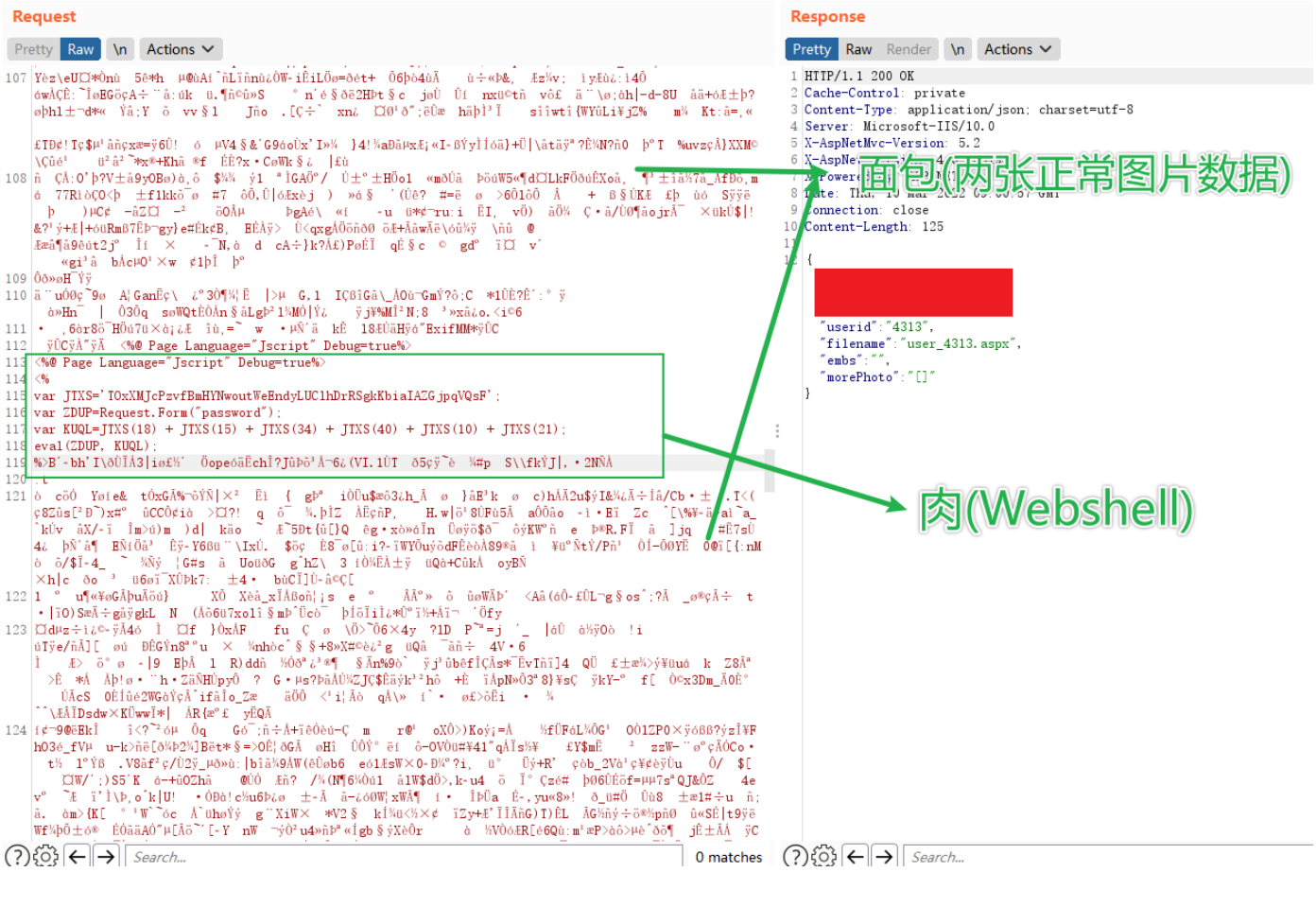


后面根据.NET特性，使用了UTF-16编码，也被拦截了，看来飞塔对编码这块有所防范。

今天是肯德基疯狂星期四，突然一个想法，在我脑中闪过——“我们能不能在万绿丛中夹带一点红呢”

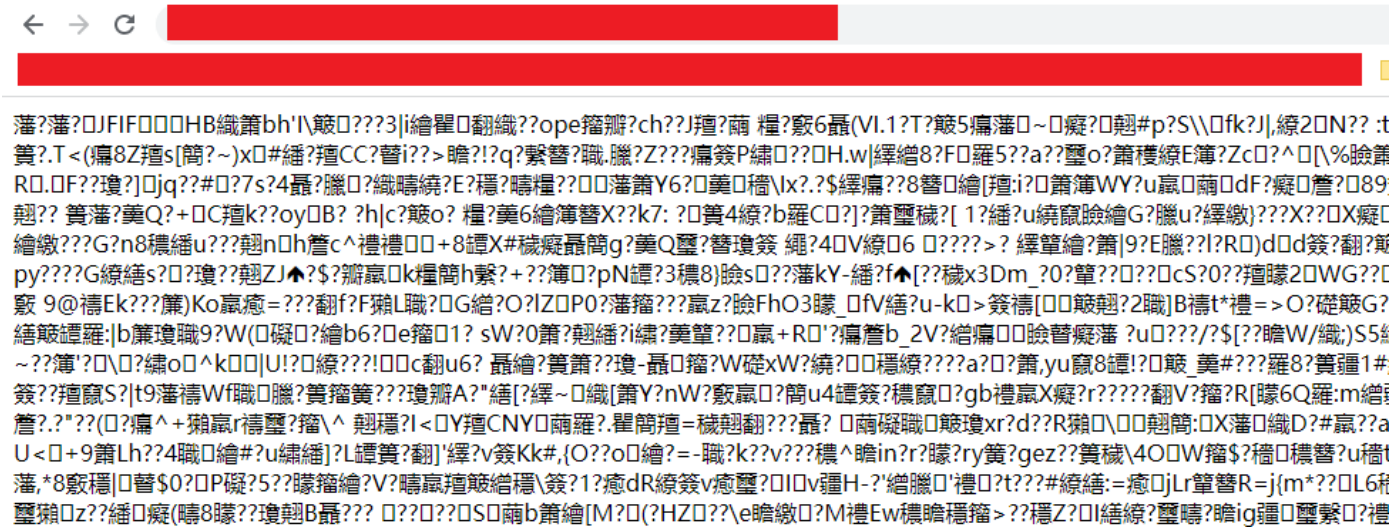
我们来试试“汉堡包法”

选取两张正常图片数据，中间夹带上我们的恶意代码



如图，成功上传带有恶意代码的aspx文件，飞塔就此告破~

翻了一下后台的图片路径，成功拼接出shell的url路径



蚁剑加密一下流量，成功Getshell

0x05 总结

善用编码、宽字节，熟知每种开发语言的特性，是绕过WAF不可或缺的基本功

特殊情况下，思路灵活可以出奇制胜~