文件上传XSS另类绕过思路

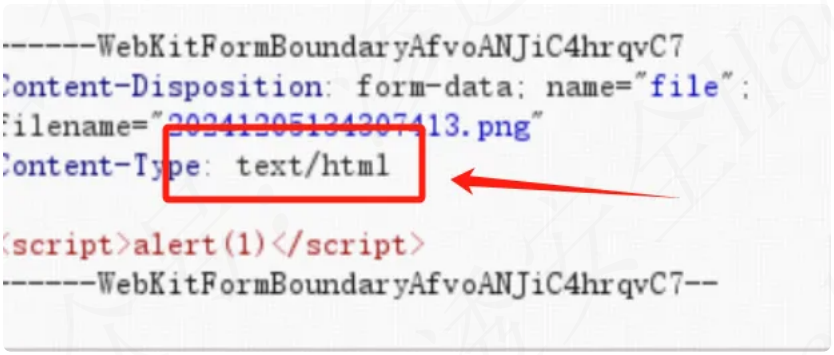
这次主题是xss绕过的一些骚姿势，关于xss，希望大家不要小看这个漏洞。如果xss可以拿到敏感的cookie，劫持关键用户账号的话，实际上达到高危是并不难的。

通常造成xss的有反射xss，和存储xss，存储xss可以有文件上传导致的xss，例如上传html、pdf，相信大家也都知道。

但是如果遇到了后端有进行检测，限制了后缀为html的文件上传（pdf除外，pdfxss似乎并不太好获取cookie，但是可以自定义弹窗）这时候该如何绕过。最近在挖掘SRC的过程中遇到了这个问题。可能同学们觉得你后缀的限制了，还能上传html？当然可以。这里我绕过了两次，所以分两部分进行讲解。

第一次，后端直接进行了后缀检测，只允许上传jpg、png后缀的文件，怎么打文件上传存储XSS?

修改content-type 为text/html

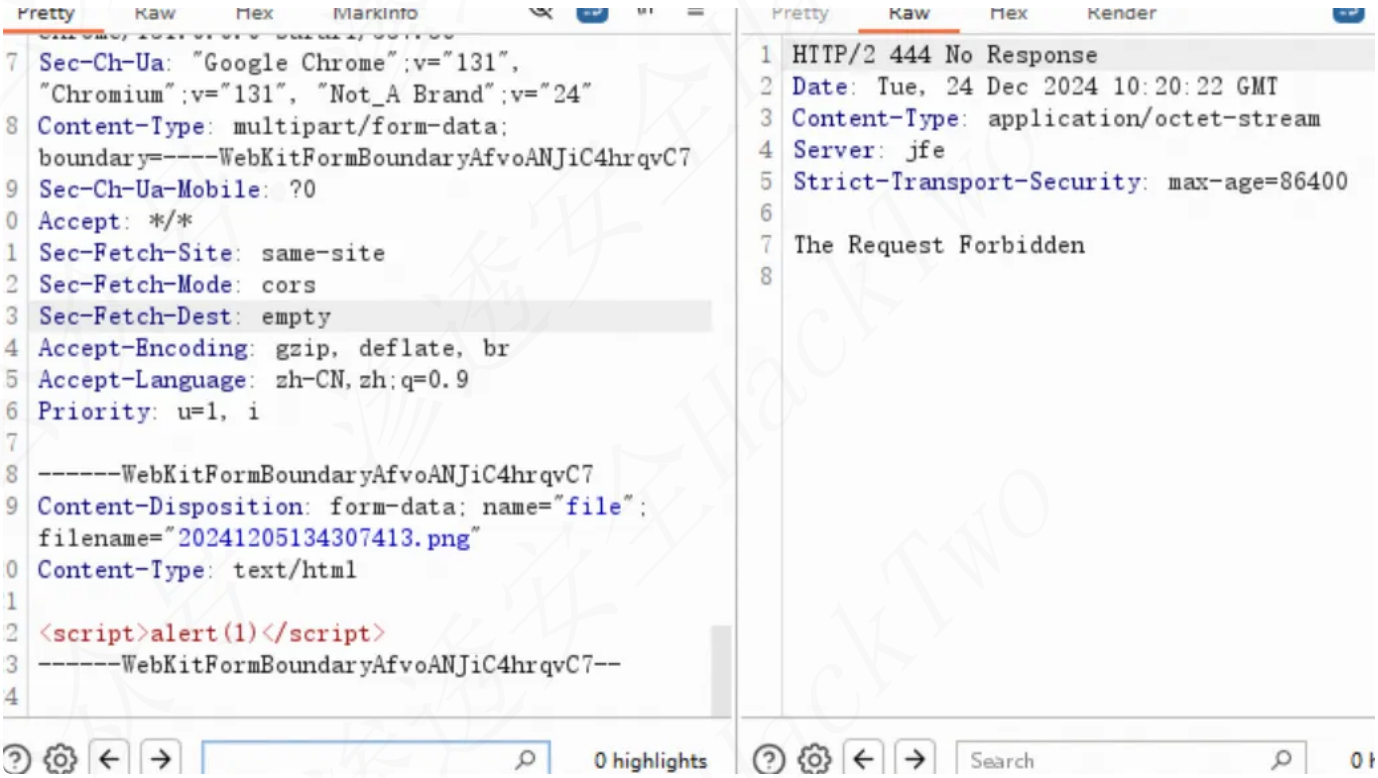


这样他后面解析的时候就会把他当作html来解析了，尽管他的后缀是合法的png也无济于事。当然也不是所有的后端都会这样，大家可以做一个尝试。那么什么时候会这样呢？放在文末来讲解。

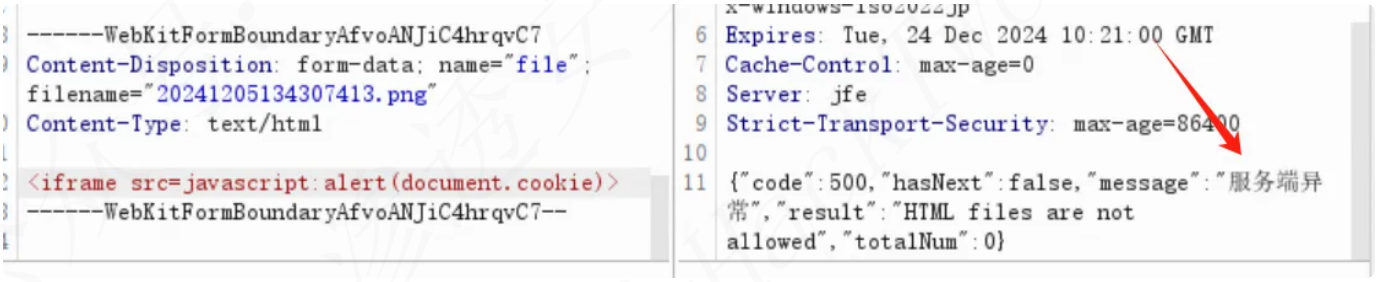
以上就是第一次绕过后缀检测来上传html。

第二次绕过，此时SRC以及让开发修复了，那么我尝试进行一个绕过

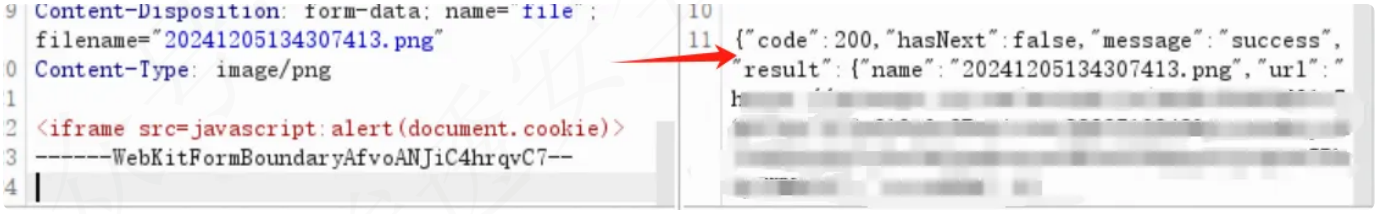
首先我还是尝试之前的上传 返回forbidden 就是被办了



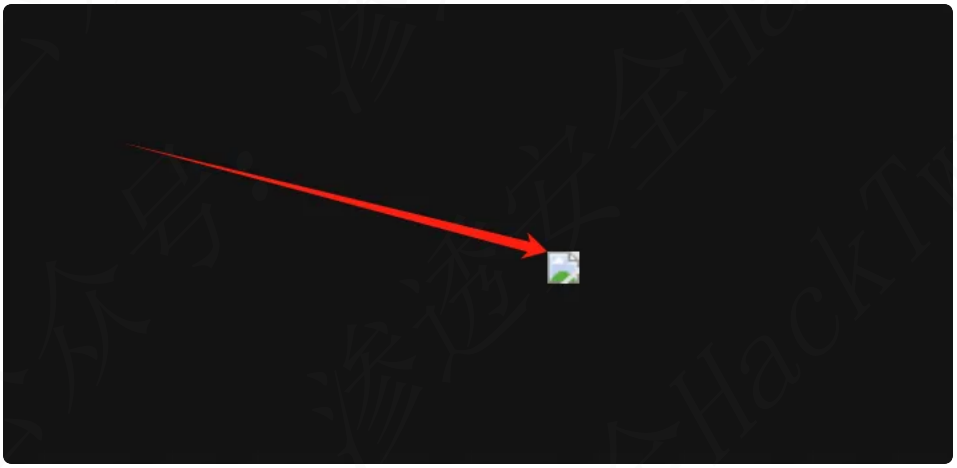
然后我换成弹cookie iframe就没有被ban掉，但是后端返回的服务端异常，推测是后端进行检测 提示 html files are not allowed 也就是html文件不允许上传



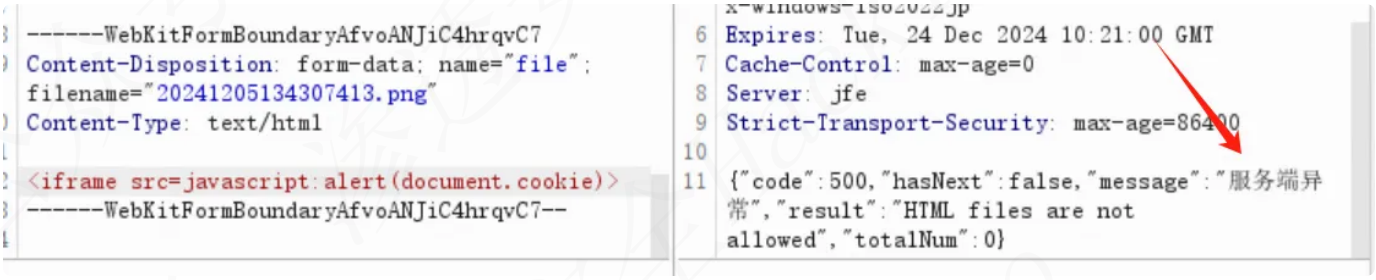
我修改content-type为合法内容时是这样的 会显示success



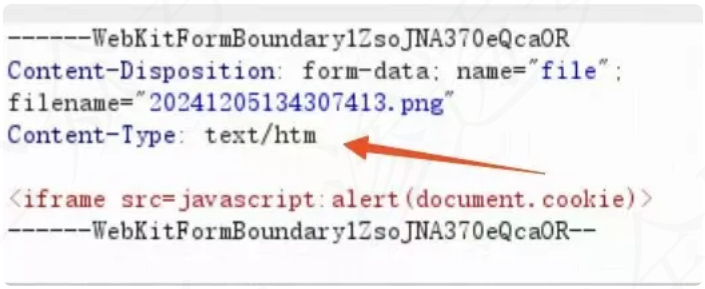
此时访问返回的链接 就会解析为一个图片



然后我再尝试对content-type进行修改，因为第一个提交src之后，我猜测开发会对content-type进行校验了，那么是不是就可以进行绕过检验的尝试呢 前面这样的content-type不行



那么试试这样的，也就是text/htm，而不是html，就可以上传成功了。(成功图片没保存)



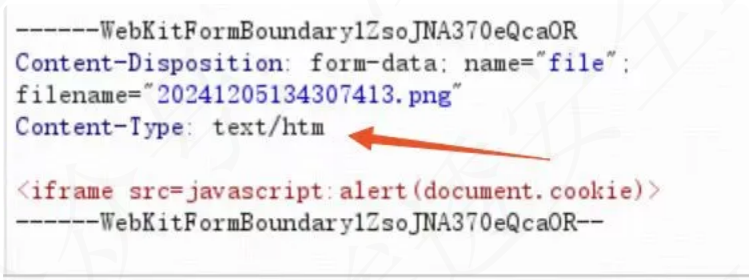
这样做是因为很多后端会将htm也解析为html，而后端可能只会匹配完整的html如下，这样的校验，那么就可以用htm进行绕过了

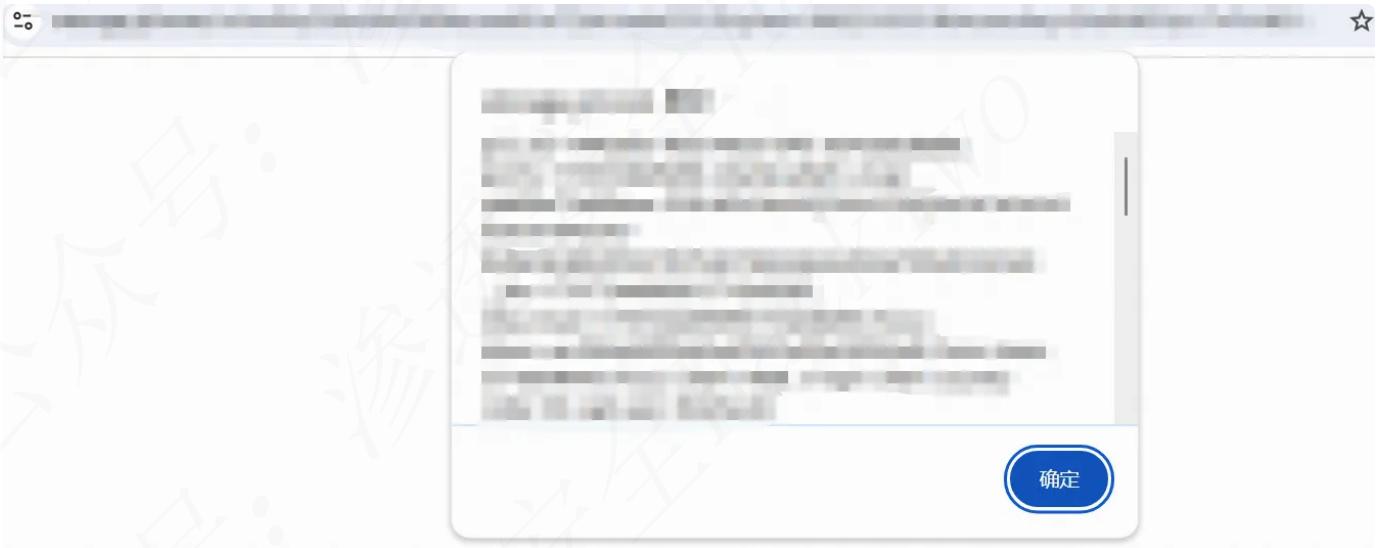
YAML

```
// 校验 Content-Type 是否包含 "html"
    if (contentType != null && contentType.contains("html")) {
        // 处理包含 HTML 的请求
        response.getWriter().write("HTML files are not allowed");
    } else {
        // 处理不包含 HTML 的请求
     ..........................................
    }
```

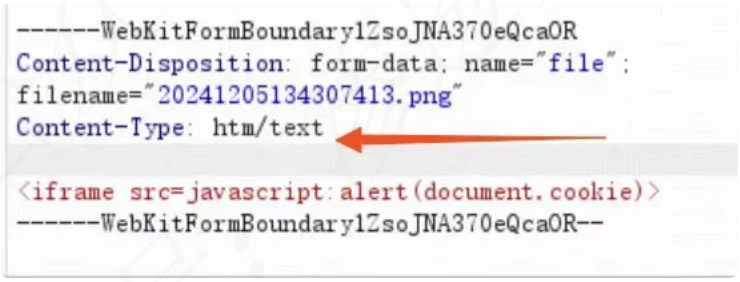
但是很可惜，后端将其解析为纯文本了，也就是text 并没有解析成html

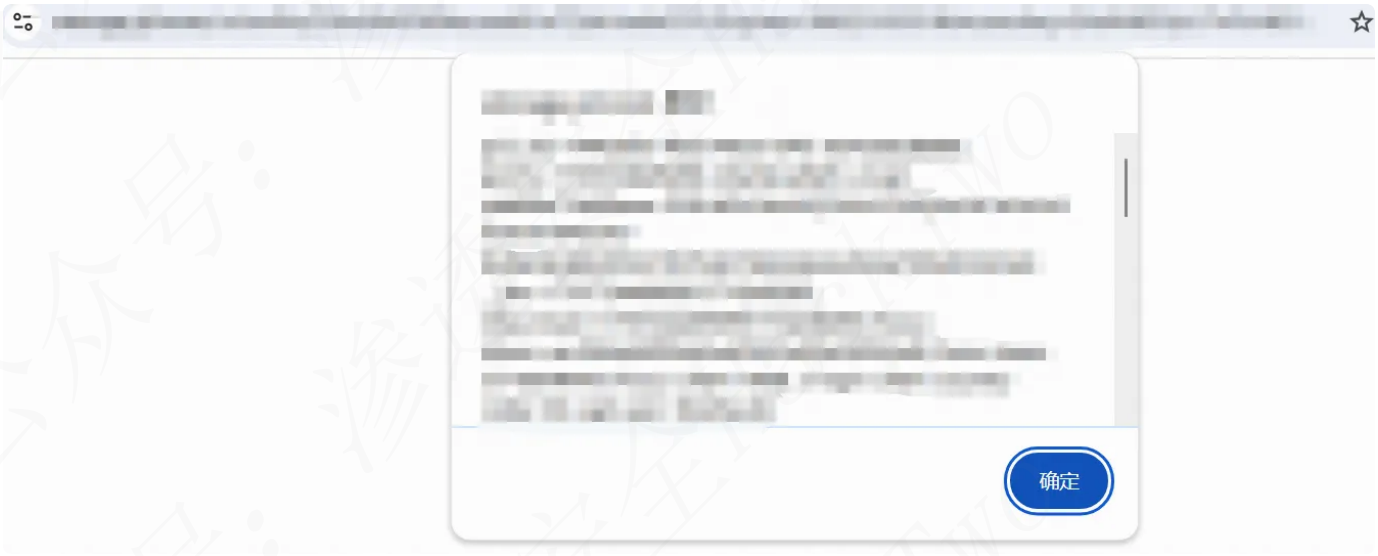
那么我将text与htm交换位置呢，很好绕过了防护 这是为什么呢？因为我猜测他会不会有个顺序？例如优先解析为text再解析为html（个人猜测，有知道的可以告诉我），所以换位置就可以，把htm换前面去 直接弹窗





那么我将text直接删了，只剩下一个htm，也绕过了防护 直接探测





最好回到前面的问题，什么情况下可以做这样的尝试呢？也就是上传png或者jpg，通过修改content-type而解析成html打存储xss。

我个人觉得只要是上传之后返回的路径是以链接的形式返回的文件路径，而不是xxx.jpg或者xxxx.png。

例如 你正常上传之后，返回的路径是这样http://a.b.com/file/xss.png这样就不会通过content-type来渲染为html，而是看后缀，直接视为图片，但是返回的路径是这样的http://a.b.com/asdfasdfqewrqeq是这样一个随机链接，他不包含xxx.png/xxx.jpg就后端就会根据content-type来进行解析，而非后缀。

以上就是全部内容，希望大家在挖掘SRC过程中多做尝试。去揣摩后端校验的逻辑来进行绕过。