# 打点高频漏洞1------ueditor漏洞利用&源码分析详细版

## 0x01 漏洞简介

大家都说这个漏洞是上传漏洞，其实这个不是上传漏洞，是远程文件下载漏洞。

思路类似于我们打内网的时候远程wget或者certutil直接把payload或者exe下载到目标机器上，然后运行上线。

利用参考

Ueditor上传漏洞复现+环境搭建 - 黑岗0x0001 - 博客园 (cnblogs.com)

这里就不在这里进行复现利用了，网上有很多，这里主要讲源码的部分。

## 0x02 ueditor C#版本源码解析

- 下载好漏洞组件，进行查看


里面这个controller.ashx相当于一个控制器，也就是入口点，类似于mvc架构中的controller，和thinkphp那种模式有点类似。

- 打开看一下这个代码

```
8
9  public class UEditorHandler : IHttpHandler
10 {
11     public void ProcessRequest(HttpContext context)
12     {
13         Handler action = null;
14         switch (context.Request["action"])//应该类比为获取 ?action=xxx这里面的值
15         {
16             case "config":
17                 action = new ConfigHandler(context);
18                 break;
19             case "uploadimage":
20                 action = new UploadHandler(context, new UploadConfig()//这一句话实际上调用了构造函数
21                 {//这里的config应该是本身就存在的config 也就是系统预先写好的config
22                     AllowExtensions = Config.GetStringList("imageAllowFiles"),
23                     PathFormat = Config.GetString("imagePathFormat"),
24                     SizeLimit = Config.GetInt("imageMaxSize"),
25                     UploadFieldName = Config.GetString("imageFieldName")
26                 });
27                 break;
28             case "uploadscrawl":
29                 action = new UploadHandler(context, new UploadConfig()
30                 {
31                     AllowExtensions = new string[] { ".png" },
32                     PathFormat = Config.GetString("scrawlPathFormat"),
33                     SizeLimit = Config.GetInt("scrawlMaxSize"),
34                     UploadFieldName = Config.GetString("scrawlFieldName"),
35                     Base64 = true,
36                     Base64Filename = "scrawl.png"
```

不同的case，就是不同的条件跳转，参数从前台用?action=config等方式传入，然后进入后台，对应不同的case，即进入不同的逻辑。

别的我们先不看，先看漏洞产生的逻辑语句。

- 直接定位到漏洞的逻辑语句

```
62                 break;
63             case "catchimage"://这里条件分支跳转
64                 action = new CrawlerHandler(context);//实例化一个crawlerhander 也就是爬虫处理
65                 break;
66             default:
67                 action = new NotSupportedHandler(context);
68                 break;
69         }
70         action.Process();
```

然后这里看到有个CrawlerHandler，是new出来的

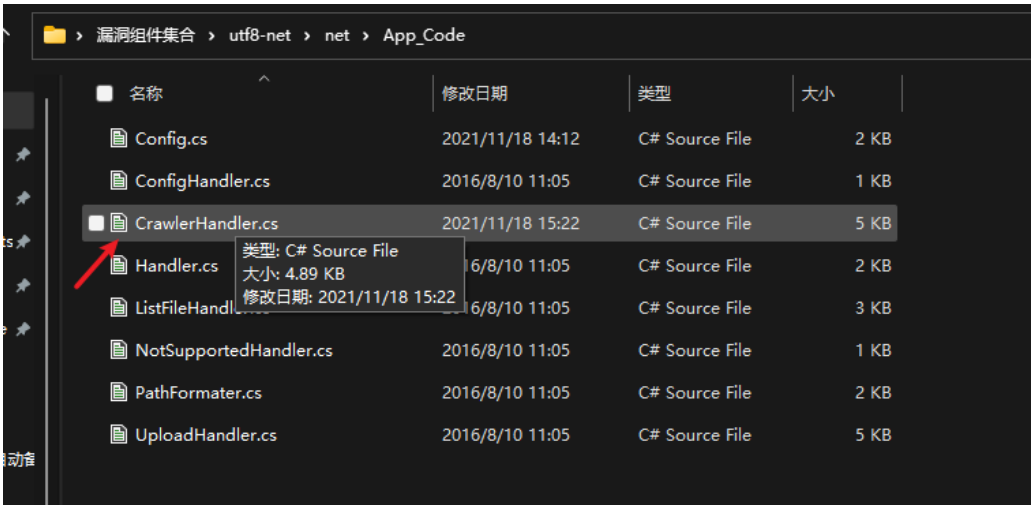
```
            break;
        case "catchimage"://这里条件分支跳转
            action = new CrawlerHandler(context);//实例化一个crawlerhander 也就是爬虫处理
            break;
        default:
            action = new NotSupportedHandler(context);
            break;
    }
```

crtl键按住查看

```
4
5   using System.Web;
6
7   public class CrawlerHandler : Handler
8   {
9       public CrawlerHandler(HttpContext context);
10
11      public override void Process();
12  }
```

非常好，啥也没有，这里我对c#也不是很了解，于是我又打开原本的那个文件来看

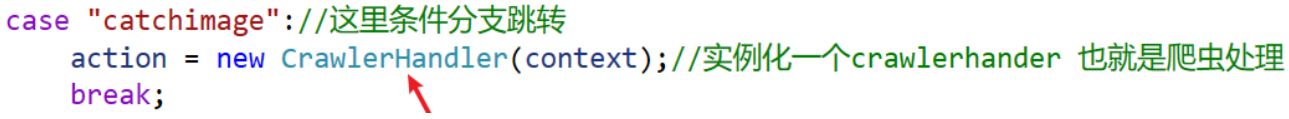

这里就找到了对应的cs文件




cs文件就是c#文件的源码，c sharp = cs，有点东西，我是猜的，不知道是不是这么来的这个文件名。

然后打开看一下


看不明白，有点晕，但是结合之前




这一行，我明白，他在new一个东西，new东西的时候通常要调用一个构造方法，也就是和CrawlerHandler同名的一个构造方法。

这个构造方法在这里

```
/// Crawler 的摘要说明
/// </summary>
public class CrawlerHandler : Handler//实例化一个类
{
    private string[] Sources;//定义sources
    private Crawler[] Crawlers;//定义爬虫
    public CrawlerHandler(HttpContext context) : base(context) { }

    public override void Process()
    {
        Sources = Request.Form.GetValues("source[]");
        if (Sources == null || Sources.Length == 0)
        {
            WriteJson(new
            {
                state = "参数错误：没有指定抓取源"
            });
            return;
        }
        Crawlers = Sources.Select(x => new Crawler(x, Server).Fetch()).ToArray
```

也没啥用

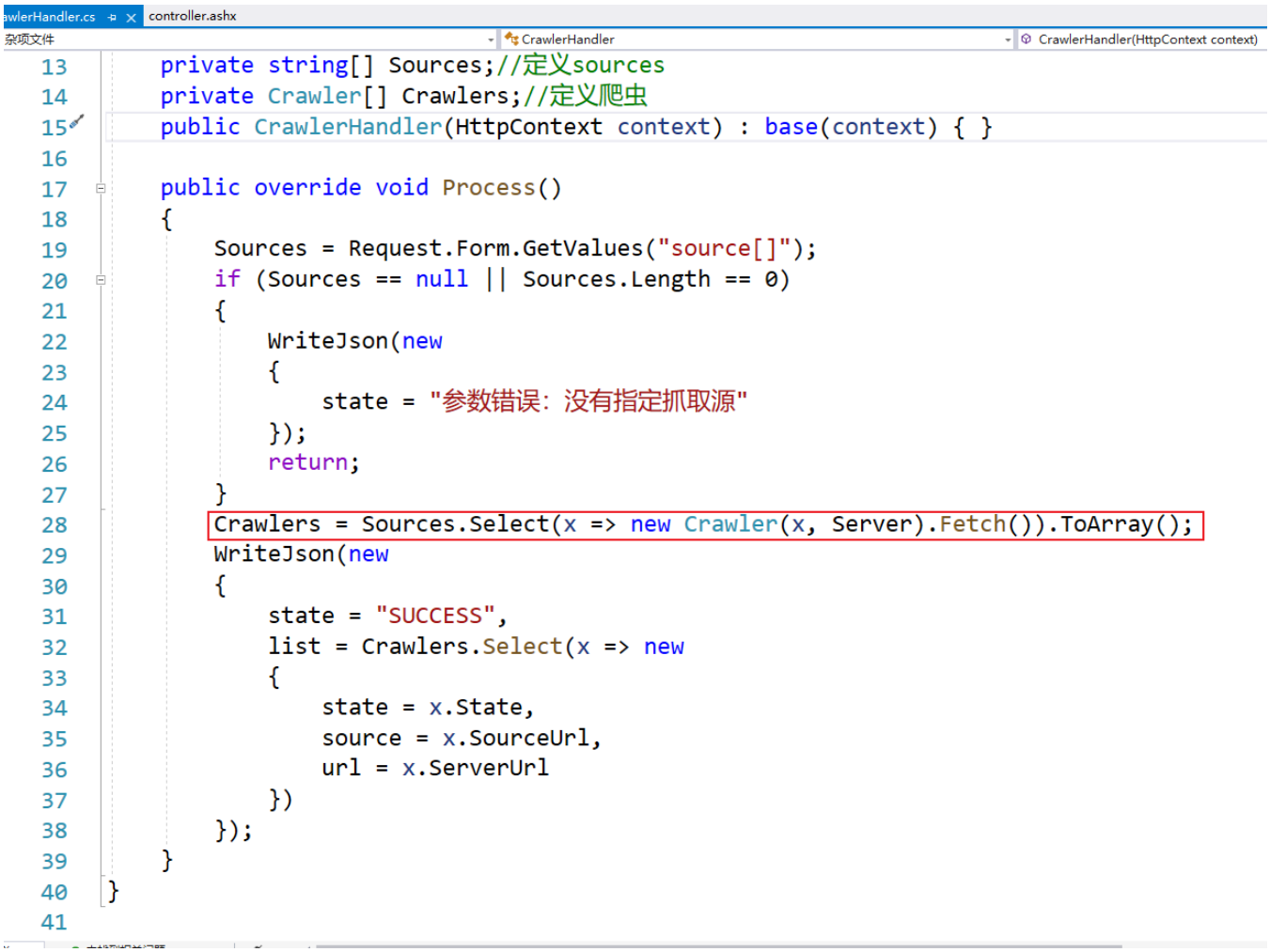
然后又看回去

发现一行

```
            case "listimage":
                action = new ListFileManager(context, Config.GetString("
                break;
            case "listfile":
                action = new ListFileManager(context, Config.GetString("
                break;
            case "catchimage"://这里条件分支跳转
                action = new CrawlerHandler(context);//实例化一个crawlerh
                break;
            default:
                action = new NotSupportedHandler(context);
                break;
        }
        action.Process();
    }

```

这里调用了一个Process方法，我理解每一个对应的cs文件中都有一个Process方法，前台参数是谁，就传给谁。




这里有个fetch方法，跟一下

```
public Crawler Fetch()
{
    if (!IsExternalIPAddress(this.SourceUrl))
    {
        State = "INVALID_URL";
        return this;
    }
    var request = HttpWebRequest.Create(this.SourceUrl) as HttpWebRequest;
    using (var response = request.GetResponse() as HttpWebResponse)
    {
        if (response.StatusCode != HttpStatusCode.OK)
        {
            State = "Url returns " + response.StatusCode + ", " + response.StatusDescription;
            return this;
        }
        if (response.ContentType.IndexOf("image") == -1)
        {
            State = "Url is not an image";
            return this;
        }
        ServerUrl = PathFormatter.Format(Path.GetFileName(this.SourceUrl), Config.GetString("catcherPa
        var savePath = Server.MapPath(ServerUrl);
        if (!Directory.Exists(Path.GetDirectoryName(savePath)))
        {
            Directory.CreateDirectory(Path.GetDirectoryName(savePath));
        }
        try
```

这里仅对文件response的contenttype做了判断，只要是image即可，因此通用poc用的都是图片马。

poc中有个特殊的的点，网上很多文章没讲清楚

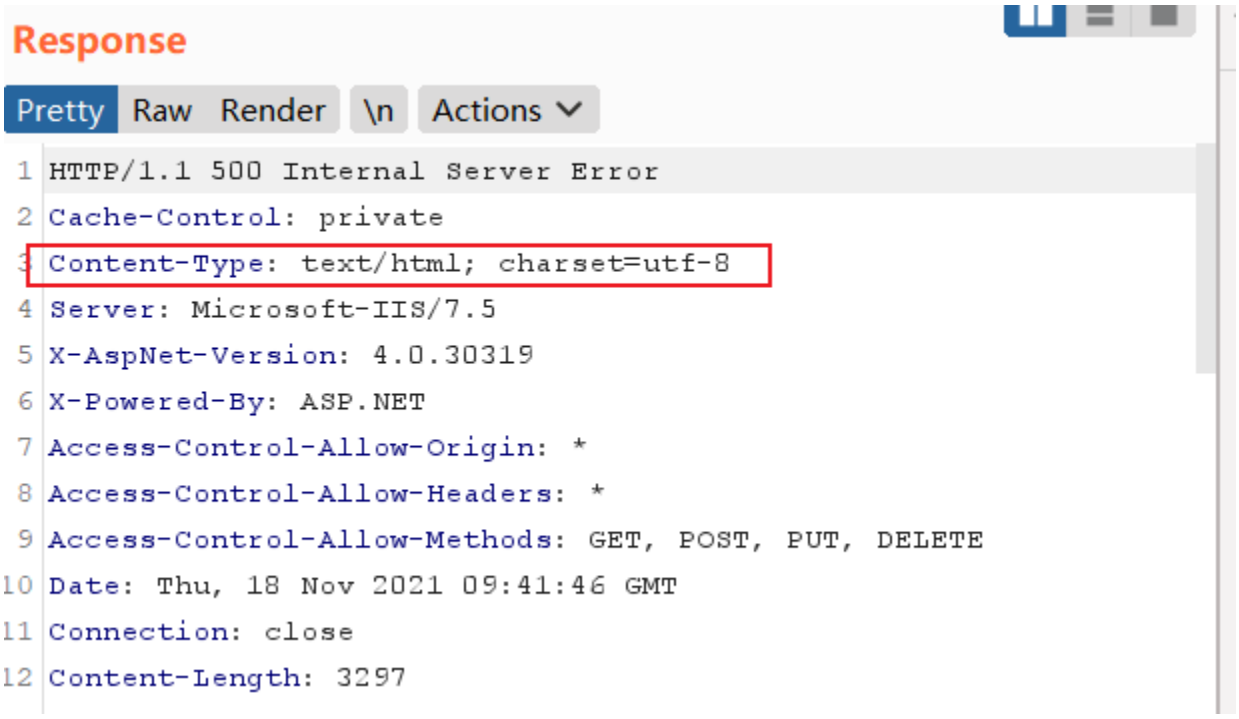
传过去的参数是

1.gif?.aspx

之所以为什么是1.gif?.aspx才能生效，这里还得看源码

首先这里写明了response中的content-Type是image

也就是返回包的部分




所以上传的东西必须是一个具有image文件头的东西

其次后台对于文件名的判定是通过.来判定

也就是这一行代码

```
75              return this;
76          }
77          ServerUrl = PathFormatter.Format(Path.GetFileName(this.SourceUrl), Config.GetString("catcherPathFo
78          var savePath = Server.MapPath(ServerUrl);
79          if (!Directory.Exists(Path.GetDirectoryName(savePath)))
80          {
81              Directory.CreateDirectory(Path.GetDirectoryName(savePath));
82          }
83          try
```

就用的是这个GetFileName方法

这个方法经过我的测试


这个payload会被当成一个名为1.gif?的文件，但是后缀还是.asp，也就是按照asp来做解析

而这个文件，在前台1.gif?xxxxx按照浏览器的逻辑，?后面就是参数，因此前台解析这个http://127.0.0.1/xxx/1.gif?.asp

是当成图片文件来做解析的

真是有意思 原理已经剖析完毕了 今天先写到这里

我还有个大胆的想法，就是那个response.ContentType.Index()究竟是判断什么东西，如果是判断文件头，那么直接改文件内容就可以了，文件名依旧是.asp文件，到时候有空再试一试，嘿嘿，算是另外一种poc。