Markdown中的XSS

惊鸿一瞥最是珍贵 / 2019-03-10 09:00:00 / 浏览数 2537 安全技术 WEB安全 顶(0) 踩(0)

前言

最近我测试一个web网站的时候发现，我可以通过markdown编辑器和渲染包来触发跨站点脚本（XSS）漏洞。这是我第一次遇到这种类型的漏洞，我发现它特别有趣，因为

何为Markdown？

Markdown是一种用于编写和格式化内容的简单语言。简单地说，编写者只需要掌握少量的语法，就可以写出简洁美观的内容。从GitHub上的Gists和readme文件，到您正
标准化的语法允许不同的markdown处理器以不同的方式显示相同的文档。标题始终是标题，但处理器可以选择应用哪种字体和权重、将标题放置在何处以及标题在目录中的
看看一个例子

视觉效果也非常棒！但是，Medium并不存储HTML和CSS的网页，而是存储一个标记文件。

```
![The goodest boy](https://images.unsplash.com/the_good_boy.png)
```

非常实用！Medium读取这一行，由于markdown的语法，Medium知道这是一个分享出来供大家阅读的图像。Medium处理此行并生成构成本文的HTML。

怎么在markdown中触发XSS?

上一段的重点在最后一行。Medium读取markdown中的行，然后生成HTML。敲黑板！如果不能安全地实现这一点，我们可以在markdown中写入恶意JavaScript代码，因
在我测试的Web应用程序中，我知道触发XSS不是很容易的一件事。它是一个Angular应用程序，默认情况下会清除页面上渲染的所有内容。而且，基于对API的测试，我知
但是，我认为，如果在web应用程序或API上没有正确地对这些代码进行清理，markdown就会是一个突破口。

再探Markdown

Markdown中的另一个例子是链接，它的语法与图像相同，但是没有前缀'!'。

```
[Click Me](https://www.example.com/)
```

当由Medium处理时：

```
<a href="https://www.example.com/">Click Me</a>
```

如果我们能够在markdown中精心设计，我们就应该能够修改生成的HTML。

exploit

最初的漏洞利用十分简单，从上面的代码来看，我们可以从href属性中转义，并添加一些在DOM事件上触发的脚本。或者，我们直接将代码放在href中。

```
<a href="javascript:alert('XSS')">Click Me</a>
```

将payload放在括号中

```
[Click Me](javascript:alert('Uh oh...'))
```

果不其然，这奏效了。现在我们有了一个链接，当我们点击它时，它会弹出一个警告。这表明前端和后端都没有将markdown视为XSS向量，或者没有正确地进行处理。

这是就完了吗？

首先，在执行JavaScript之前，用户必须实际单击该链接。理想情况下，我们希望仅通过访问页面来执行它。其次，一个恶意链接没有什么效果，那这次攻击就毫无意义。
我们需要在页面加载并在用户不知道的情况下，悄无声息地利用漏洞展开攻击。这让我们将视角切回到图像文件。如果我们可以创建一个图像并将脚本设置为在加载图像时

再进一步！

回到markdown中的图像语法

```
![The goodest boy](https://images.unsplash.com/the_good_boy.png)
```

这是最终的HTML：

```
<img src="https://images.unsplash.com/the_good_boy.png" alt="The goodest boy">
```

当然，我们可以将相同的payload放在括号中触发XSS。但这并没有什么意义。
很重要的一点是：

> markdown如何渲染为HTML，因markdow不同而异。

在Markdown中将JavaScript注入图像代码的最佳方式

```
![Uh oh...]("onerror="alert('XSS'))
```

这是我发现的第一个payload。当JavaScript代码直接放置在src或alt属性中时，似乎无法执行，但我可以关闭src属性并添加更多属性。这个过程为:

```
<img src="" onerror="alert('XSS') alt="Uh oh...">
```

由于src值为空，因此加载图像将导致代码执行错误。所以我这样构造：

```
![Uh oh...](https://www.example.com/image.png"onload="alert('XSS'))
```

事实证明，我们仍然可以添加源链接并添加onload属性，该属性在页面加载后执行。

谢谢阅读！

总结

聪明的开发者面对这种情况也做了充分的准备，做好渗透测试同样也非常重要，bug
bounty计划也为应用程序的安全保驾护航，但开发人员也要时刻保持警醒，满足功能需求的同时也必须考虑安全性。当你在外面进行测试的时候，不要忘记markdown,这里

```
https://blog.usejournal.com/exploiting-xss-via-markdown-72a61e774bf8
```

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板