SWRhapsody

XSS

于7月 23, 2019由**SWRhapsody**发布

Introduction

想写个对于各种漏洞的总结，这个是第一篇

XSS（Cross Site Scripting）全称跨站脚本攻击，这是一种注入攻击，它将允许攻击者在受害者的浏览器中执行 CSS 和 JavaScript 代码。对于这个漏洞，攻击者通常不直接攻击受害者的浏览器，而是通过把代码注入到存在漏洞的网站或者使用自己的钓鱼网站并诱导用户访问这些网站，攻击者注入的代码成为网站的一部分并在用户的浏览器中执行。

Code

以下代码片段是一个简单的 XSS[2]

```
1 <% String eid = request.getParameter("eid"); %>
2 ...
3 Employee ID: <%= eid %>
```

这是段 JSP 代码，读取 Get 请求中的 eid 的值然后显示，由于没有任何的输入校验，可以很简单的通过 eip 来注入Javascript

```
1 <% String eid = request.getParameter("eid"); %> //eip="<script>alert(0)</script>"
2 ...
3 Employee ID: <%= eid %> // <%= "<script>alert(0)</script>"%>
```

XSS 的危害

XSS 主要会产生以下几种危害：

（1）窃取用户的 Cookie（如果没有设置HttpOnly），获取其他用户信息（BeEF 或 Keylogging）。

（2）挖矿（原来 reddit 上有篇关于用 webassbot 来讨论浏览器挖矿公用运行成本的讨论，不过我现在

SWRhapsody

（4）操作 DOM 节点改变当前的页面。

（5）钓鱼。

当然，取决于网站漏洞的具体情况，XSS 造成的损失也有很大的区别。

XSS 的成因

过分信任用户的输入或过分信赖系统其他组件（之前有个客户端过分信任服务端然后攻击者做了一个假的服务端钓鱼的漏洞，不过现在我找不到了）。

XSS 的分类

XSS主要可以分为一下三类

（1）储存型XSS，注入的代码存在服务器的数据库或者其他地方。

（2）反射型XSS，注入的代码在请求中，一般诱导用户。

（3）DOM型XSS，注入的代码存在用户的浏览器中而不是服务器上。

储存型XSS

CVE-2019-10864 [4]

这是一个 WordPress Plugin 的 XSS 漏洞，存在漏洞的组件是 Wp-Statistics。Wp-Statistics 是一个统计站点访问信息的插件

漏洞出在 Referrers 功能中，这个功能用于统计 Referrer 站点的 IP，Title，地址，不过在获取网站 Title 时由于开发人员的失误导致了漏洞。受影响的为12.6.2以及更低的版本，以下是具体代码

```
1  function wp_statistics_get_site_title( $url ) {
```

SWRhapsody

```
7
8       //Get Page Title
9       if ( class_exists( 'DOMDocument' ) ) {
10          $dom = new DOMDocument;
11          @$dom->loadHTML( $html );
12          $title = '';
13          if ( isset( $dom ) and $dom->getElementsByTagName( 'title' )->length > 0 ) {
14              $title = $dom->getElementsByTagName( 'title' )->item( '0' )->nodeValue;
15          }
16          return ( wp_strip_all_tags( $title ) == "" ? false : $title );
17      }
18      return false;
19
20  }
```

我们可以看到

```
1  return ( wp_strip_all_tags( $title ) == "" ? false : $title );
```

虽然对获取的网站标题进行了过滤但最后函数返回时还是返回了未过滤的变量。这个函数在 top-referrings.php中被调用

```
1  //Get Site Link
2
3  $referrer_html = $WP_Statistics->html_sanitize_referrer( $domain );
4  //Get Site information if Not Exist
5  if ( ! array_key_exists( $domain, $referrer_list ) ) {
6      $get_site_inf             = wp_statistics_get_domain_server( $domain );
7      $get_site_title           = wp_statistics_get_site_title( $domain );
8      $referrer_list[ $domain ] = array(
9          'ip'      => $get_site_inf['ip'],
10         'country' => $get_site_inf['country'],
11         'title'   => ( $get_site_title === false ? '' : $get_site_title ),
12     );
13 }
14 echo "<tr>";
15 echo "<td>" . number_format_i18n( $i ) . "</td>";
16 echo "<td>" . wp_statistics_show_site_icon( $domain ) . " " . $WP_Statistics->get_referrer_lin
17 echo "<td>" . ( trim( $referrer_list[ $domain ]['title'] ) == "" ? $unknown : $referrer_list[ 
18 echo "<td>" . ( trim( $referrer_list[ $domain ]['ip'] ) == "" ? $unknown : $referrer_list[ $do
```

在某个站点第一次出现在 Referrer 时插件会获取站点的 Title，最后显示在用户的页面上。利用这个漏洞我们需要一个域名与一个 Title 是我们 XSS Payload 的网页。域名最简单可以通过修改主机的 host 文件来获得

```
1  XXX@scp:~$ cat /etc/hosts
2  127.0.0.1   localhost.localdomain   localhost
3  ::1     localhost6.localdomain6 localhost6
4  XXX.XXX.XXX.XXX   www.motherboard.com
5
6  # The following lines are desirable for IPv6 capable hosts
7  ::1     localhost ip6-localhost ip6-loopback
8  fe00::0 ip6-localnet
9  ff02::1 ip6-allnodes
10 ff02::2 ip6-allrouters
11 ff02::3 ip6-allhosts
```

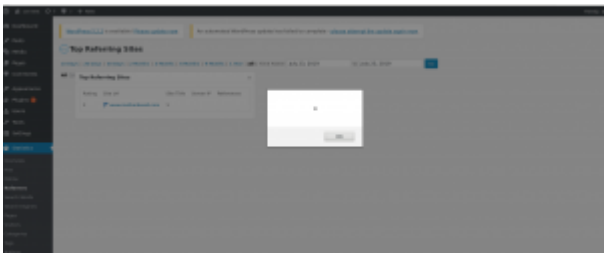
SWRhapsody

```
<head>
  <title>1 <script><script>alert(0)<</script>/script></script> 3</title>
</head>
<body>
  <h1>Hello :)</h1>
  <a href="http://XXX.XXX.XXX.XXX/wordpress/">link</a>
</body>
</html>
```

不过这里我不知道绕过

```
$title = $dom->getElementsByTagName( 'title' )->item( '0' )->nodeValue;
```

要这样写。最后成功触发。



[5] 和 [3] 也是储存型 XSS。 [5] 主要是因为自写的过滤函数只移除一遍各种标签导致可被绕过。 [3] 的XSS 注入位置挺有趣的。

反射型XSS

反射型 XSS 主要出现在网站显示用户输入的地方，[6]是一个 OrientDB 的漏洞，问题出在 HTTP API 的 document 功能中。

OrientDB 是一个 NoSql 的数据库，我们使用如下请求时

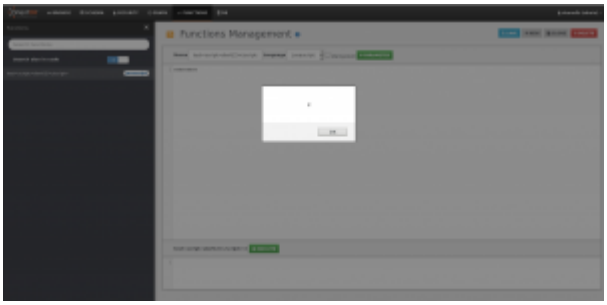
```
POST http://XXX.XXX.XXX.XXX:2480/database/demodb HTTP/1.1
Host: XXX.XXX.XXX.XXX:2480
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: OSESSIONID=OS1563974189707-2375913037217036824
Upgrade-Insecure-Requests: 1
DNT: 1

{"@class":"ofunction","@version":0,"@rid":"#-1:-1","idempotent":null,"name":"test<script>alert
```

会创建一个新的类型为 function 的 document，id 是 -1:-1 在发送完如下请求时登入后台的 function management 时可以看到已经创建了一个新的 function

SWRhapsody

不过这时 Payload 没有触发，随便输入些内容然后点击保存



DOM型XSS

DOM 型 XSS 算是前两种 XSS 的变种，DOM 型 XSS主要是注入到网站的 JavaScript 中，我们看下这段代码

```
function funct3(){
    var pos = document.URL.indexOf("name=")+5;
    var r ='<b>'+document.URL.substring(pos,document.URL.length)+'<b>';
    document.write(r);
}
```

如果请求 `http://xxx.xxx.com/vuln.php#name=alert(1);` 便会触发漏洞。

XSS的防御

对于 XSS 的防御最简单的是过滤掉所有的 Html 和 Javascript 标签，稍微复杂点的可以从前端和后端来防御，并且在 [7] 里面很好列出了几种防御的原则。

前端：

除去过滤输入，前端可以通过使用 `MutationObserver` 来监听[10] [11]

```
function interceptionStaticScript()
{
```

SWRhapsody

```
        });
    });
    //为观察者对象传入要观察的对象，并设置config
    //这里的对象为document，配置为subtree+childList，也就表示会在所有的节点的子节点新建或时触发事件
    observer.observe(document, {
        subtree: true,
        childList: true
    });
}
//事件触发时的响应函数
function begininterceptionstatic(mutation)
{
    var nodes = mutation.addedNodes;
    for (var i = 0; i < nodes.length; i++)
    {
        var node = nodes[i];
        //挑选出来script标签与iframe标签
        if (node.tagName === 'SCRIPT' || node.tagName === 'IFRAME')
        {
            //如果iframe标签有srcdoc的话就在dom里删除此标签
            if (node.tagName === 'IFRAME' && node.srcdoc)
            {
                node.parentNode.removeChild(node);
                console.log('Attack detected in iframe. ', node.srcdoc);
            }
            //检测script以及iframe标签的匹配方式，只有通过白名单的才行
            else if (node.src)
            {
                if (!whileListMatch(whiteList, node.src))
                {
                    node.parentNode.removeChild(node);
                    console.log('Attack detected. ', node.src);
                }
            }
            //对iframe以及script标签的内容也就是innerHTML进行黑名单检测
            else if (blackregmatch(jswordblacklist,node.textContent))
            {
                node.parentNode.removeChild(node);
                console.log('Attack detected. '+node.textContent);
            }
        }
    }
}
```

也可以用钩子来过滤处理各个注入点，innerhtml，iframe，image 等的src 和 document.write()

```
//首先获得这个函数的对象
var old_write = window.document.write;
//然后重写此函数
window.document.write = function(string)
{
    //基于函数的参数进行自定义的操作
    if (blackregmatch(wordblacklistDocumetnWrite, string))
    {
        console.log('Attack detected. ', string);
        return;
    }
    //接着运行函数原生代码：使用函数对象.apply(document, arguments)这种写法
    old_write.apply(document, arguments);
}
```

除了使用像 htmlpurifier，OWASP Java Encoder 这样的工具，最主要的防御手段还是过滤输入，比如像富文本编译器这样的场景，需要支持一部分的 html 标签的可以使用白名单和黑名单的过滤机制 [13]

CSP与各种属性：

CSP与各种属性我想单独写一篇。

Reference

[1] https://excess-xss.com/
[2] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[3] https://blog.ripstech.com/2019/wordpress-csrf-to-rce/

[4] https://medium.com/@aramburu/cve-2019-10864-wordpress-7aebc24751c4

[5] https://www.alienvault.com/blogs/security-essentials/discovering-cve-2018-11512-witycms-0.6.1-persistent-xss

[6] https://www.exploit-db.com/exploits/46517

[7] https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html

[8] https://blog.qualys.com/technology/2017/03/06/smart-dom-xss-detection-in-qualys-was

[9] https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

[10] http://cauc.me/2017/09/19/XSS%E5%89%8D%E7%AB%AF%E9%98%B2%E5%BE%A1%E2%80%94%E2%80%94%E2%80%94%E5%AF%B9WHCTF%E4%B8%ADXSS%E9%A2%98%E7%9B%AEfilter-js%E7%9A%84%E5%88%86%E6%9E%90/

[11] http://fex.baidu.com/blog/2014/06/xss-frontend-firewall-3/

[12] https://juejin.im/post/5bac9e21f265da0afe62ec1b

[13] https://www.leavesongs.com/PENETRATION/xsshtml.html

分类： ARTICLE COLLECTION

SWRhapsody

0 条评论

发表评论

名称 *

电子邮件 *

网站

在想些什么?

发表评论

近期文章

携程Apollo YAML 反序列化

CVE-2020-5410

CodeQL部分源码简读

服务器与网关的不一致

SWRhapsody

近期评论

文章归档

2020年8月

2020年6月

2020年5月

2020年3月

2020年1月

2019年12月

2019年11月

2019年8月

2019年7月

2019年5月

2019年4月

2019年1月

2018年11月

2018年10月

2018年9月

2018年4月

2018年3月

2018年2月

2018年1月

分类目录

SWRhapsody

cryptography

Exercise

Exploit

HackTheBox

Penetration Test

Uncategorized

相关文章

ARTICLE COLLECTION

各种文章收集

DDOS CC攻击原理及防范方法: https://www.cnblo 阅读更多…

ARTICLE COLLECTION

CSP

Introduction Content Security Polic 阅读更多…

SWRhapsody