XSS Dynamic Detection Using PhantomJs

笑然 / 2016-10-30 05:26:00 / 浏览数 2358 安全技术 漏洞分析 顶(0) 踩(0)

```
[+] Author: fridayy
[+] Team: n0tr00t security team
[+] From: http://www.n0tr00t.com
[+] Create: 2016-10-29
```

XSS 是典型的浏览器端漏洞，由于用户的输入未经转义直接输出到页面中，恶意代码在用户的浏览器中被解析，从而造成危害。传统的反射型 XSS 可以通过判断页面源码是否含有特定字符串来检测。但由于 Web 2.0 的快速发展交互越来越复杂，DOM-XSS 也层出不穷，导致传统的检测方案的漏报率很高。本文主要介绍了如何利用 PhantomJS + Python 完成动态检测。

0x01 PhantomJS

既然是动态检测，那么就需要一个浏览器，但普通的浏览器在渲染页面上花费了太多的资源和时间，并不适用。怎么办？当然开源世界早有解决方案：PhantomJS、PyQt、等等。对比了一下上手难易程度、文档丰富程度等，我选择了 PhantomJS 进行开发。

PhantomJS 是无界面的 Webkit 解析器，提供了 JavaScript API 。由于去除了可视化界面，速度比一般 Webkit 浏览器要快很多。同时提供了很多监控和触发接口，可以方便的操作页面 DOM 节点，模拟用户操作等。

0x02 漏洞判别标准

XSS 漏洞说到底还是用户输入被当成页面代码解析了，解析的结果可能是执行了JS代码，也可能是在页面中创建/修改了某个 DOM 节点(有部分过滤，无法执行JS代码的情况下)。所以我们将 Payload 大概分为两类：

- 第一类，执行了指定的JS代码（`alert(1)`）
- 第二类，创建了新的DOM节点（`<xsstest></xsstest>`）。

根据这两种 Payload ，简化的漏洞判别标准如下：

1.页面弹窗（在PhantomJS中重载`window.alert`）

2.新节点（解析玩页面后，判断`document.getElementsByTagName('xsstest')`是否为空）。

```
page.onAlert = function (message) {
   if(message == xss_mark) {
       xss_exists = 1;
       ret = "Success, xss exists";
       phantom_exit(ret);
   }
   console.log('Alert: ' + message);

   return true;
};

function check_dom_xss_vul(){
   return document.getElementsByTagName(dom_xss_mark).length;
}
```

为了验证检测代码，编写一个简单存在XSS漏洞的页面。

```
<?php
echo $_GET['test'];
?>
```

经测试，访问 `http://127.0.0.1:8000/xss.php?test=<img src=1 onerror=alert(1)>`，我们的检测代码成功检测到了弹窗，并返回了正确的结果。但是，如果是下面这种情况呢？

```
<?php
$click = $_GET['test'];

echo "<div onclick=$click></div>";
?>
```

0x03 执行事件代码

很明显，我们需要执行`onclick`中的代码，才能检测到漏洞。首先我们想到的是触发事件，仅仅是触发 click 事件：`document.getElementsByTagName('div')[0].click()`。但是 Javascript 也就仅仅提供了 click 事件的触发函数而已。既然代码直接输出在了

onclick/onmouseover 之类的属性里，我们遍历所有节点的属性，针对 onxxxxx 的属性值，直接调用 eval 方法，执行对应的代码就可以了。

```
var nodes = document.all;
for(var i=0;i<nodes.length;i++){
   var attrs = nodes[i].attributes;
   for(var j=0;j<attrs.length;j++){
       attr_name = attrs[j].nodeName;
       attr_value = attrs[j].nodeValue;
       if(attr_name.substr(0,2) == "on"){
           console.log(attrs[j].nodeName + ' : ' + attr_value);
           eval(attr_value);
       }
   }
}
```

访问 http://127.0.0.1:8000/xss.php?test=alert(1) 成功执行代码，但新的问题很快出现：并不是所有的JS代码都是以内联的形式写入到 HTML 代码中的，程序猿们往往更喜欢通过 document.addEventListener 或者 jQuery 中的 $('dom').click 直接绑定事件。例子如下：

```
<script type="text/javascript" src="http://apps.bdimg.com/libs/jquery/2.1.1/jquery.min.js"></script>

<div class="link-area"></div>

<?php
echo '<script>$("#image").click(function(){$(".link-area").html("'.$_GET['test'].'")});</script>';
?>
```

0x04 触发事件

所以我们现在需要这样的接口：能够触发某个 DOM 节点的某个事件，包括但不仅限于 click 事件。 PhantomJS 和 JavaScript 都可能存在这样的接口，但是找遍了 PhantomJS 的接口，也只是发现了触发 click 事件的接口。所以聚焦点重新回到 Javascript 上来。很快我发现了 dispatchEvent 这个函数。

```
// phantom_finish.js
var evt = document.createEvent('CustomEvent');
evt.initCustomEvent(click, true, true, null);
document.getElementsByTagName("div")[0].dispatchEvent(evt);
```

成功执行了 click 事件，但是如何能获取到所有节点的绑定事件呢？有两种方法：

- 遍历所有节点，获取每个节点绑定的事件
- 在dom节点加载前，重写addEventListener方法，并将所有的绑定的事件及节点记录下来。

方法一在遇到 jQuery 绑定事件的时候扑街了。方法二明显比方法一节省资源，并且测试通过：

```
// phantom_init.js
_addEventListener = Element.prototype.addEventListener
Element.prototype.addEventListener = function(a,b,c) {
       save_event_dom(this, a);      // ■■■■■■■■■■■■■■■■■
       _addEventListener.apply(this, arguments);
};
```

这样，我们的 JS 代码也算告一段落，PhantomJS 组件能够执行内联代码及触发所有的绑定事件。万事具备，只欠一个调度系统了~

0x05 调度系统

XSS 扫描是 URL 粒度扫描，针对网站的每一个链接(去重后)都要进行测试。XSS检测系统的输入值包括：

```
* URL ■■■http://127.0.0.1:8000/xss.php?a=1&b=2■
* method
* post_data
* headers
```

调度系统的功能就是处理这个URL，拼接对应的payload，并调用 PhantomJS 组件，检测是否含有 XSS 漏洞。举个例子，当payload为 <img src=1 onerror=alert(1)> 时，需要调用两次 PhantomJS 组件，输入的URL分别为：

```
http://127.0.0.1:8000/xss.php?a=<img src=1 onerror=alert(1)>&b=2
http://127.0.0.1:8000/xss.php?a=1&b=<img src=1 onerror=alert(1)>
```

当然 Payload 不止一个，会有很多种玩法，简单提供几个基础 Payload ：

```
'"><img src=1 onerror=alert(1)>
'"><script>alert(1)</script>
```

```
';alert(1)//
";alert(1)//
'" onmouseover=alert(1)
javascript:alert(1)
'"></script><img src=1 onerror=alert(1)>
"'></textarea><xsstest>
```

0x06 更多思考

采用了 Webkit 解析器来检测XSS漏洞，提高了检测的覆盖率，也大幅降低了误报率。但有些仅在 IE
下有效的漏洞，就无法覆盖到了。上述种种，已经基本将动态XSS检测的思路分析透彻。XSS有很多种玩法，在payload中可以带进一些有意思的攻击代码，比如钓鱼、打Co

最后，再次欢迎对 XSS 利用有各种猥琐想法的同学来交流，微博 @Fr1day

上一篇：安全扫描自动化检测平台建设（Web... 下一篇：二进制入门漏洞分析思维导图

1. 0 条回复
   - 动动手指，沙发就是你的了！