Clickjacking攻防

milkfr / 2018-03-19 16:23:02 / 浏览数 18226 技术文章 技术文章 顶(0) 踩(0)

0x00 概述

- 公司渗透检查要求点击劫持，发现Chrome不支持`X-Frame-Options: ALLOW-FROM uri`，JS防御可以被绕过，然后对点击劫持又查了一些资料，记录一下
- 本文主要讲解点击劫持攻击，包括漏洞简介、威胁场景、修复方式、深入攻防和一个经典案例
- 漏洞简介用实验简单描述了点击劫持，深入攻防部分开始通过实验讲了用js防御点击劫持漏洞的绕过方式和可能更好的防御方式
- 本文主要参考《白帽子讲Web安全》和斯坦福大学论文《Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites》，推荐去看原文哦

0x01漏洞简介

点击劫持是一种视觉上的欺骗手动，攻击者可以使用一个透明的、不可见的iframe，覆盖在一个网页上，然后诱使用户在该网页上进行操作，通过调整iframe页面的位置，可

之所以叫点击劫持（Clickjacking），是因为它劫持了用户的登录态，并诱导了点击等页面操作

作为一个前端漏洞，它相对与XSS和CSRF来说，因为需要诱使用户与页面产生交互行为，因此实施攻击的成本更高，在网络犯罪中比较少见，但仍然可能被利用在钓鱼、欺诈

做个实验、举个例子

我们修改hosts文件，使一台远程主机被解析为`example.com`，本机被解析为`hacker.com`，注意一些浏览器的有DNS缓存，注意清除

我们在远程主机上运行开启一个简单http服务应用，当我们访问时会出现如下一个文件下载目录页面，页面中只有一个文件链接，我们把它看成付钱功能的按钮

我们为这个页面设置cookie

```
resp.set_cookie('session-cookie', '123')
resp.set_cookie('third-part-cookie', '456', expires='Sun, 18-Mar-2019 10:05:05 GMT')
```

之后，我们在本机随意开启一个Web应用，返回一个HTML页面，页面代码如下，它将`example.com`放在一个`iframe`标签内，将iframe页面虚化透明，在iframe页面下层

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Clickjack</title>
   <style>
       iframe {
           width: 1440px;
           height: 900px;

           position: absolute;
           top: -0px;
           left: -0px;
           z-index: 2;

           -moz-opacity: 0.2;
           opacity: 0.2;
           filter: alpha(opacity=0.2);
       }

       button {
           position: absolute;
           top: 100px;
           left: 50px;
           z-index: 1;
           width: 120px;
       }
   </style>
</head>
<body>
   <iframe src="http://www.example.com" scrolling="no"></iframe>
   <button>Click Here!</button>
</body>
</html>
```

页面的实际显示如下图所示

点击后会发送请求到example.com，并带上example.com的cookies，也就是劫持了用户的身份

一般我们把cookies分为session cookie（临时cookie，断开连接后消失）和third-part
cookie（本地存储的cookie，expires会设置过期时间，浏览器会本地保存到过期），如上分别使用了这两种类型的cookie。某些版本的浏览器（IE7、IE8）出于安全考虑对
cookie，目前主流浏览器包括IE高版本都会发送，所以基本只要带有cookie都能劫持

页面可以完全虚化透明至看不见，如果hacker.com的页面精美一些，可以更好诱导点击操作，甚至可以做个小游戏，在用户存在登录态的情况下，便可以使用用户的登录态

0x02 威胁场景

事实上我们虽然说点击劫持是劫持了用户的登录态，但实际上这是一种视觉攻击，因此也会有一些视觉效果上的攻击变种和配合

- 诱使用户登录站点，发送其他含有iframe的链接给用户，在用户不知情情况下诱使用户完成一些操作
- 通过类似flash游戏改变用户鼠标点击的位置，完成一些较为复杂的操作
- 搭配输入框填写表单
- 图片覆盖攻击(Cross Site Image Overlaying,
  XSIO)，覆盖原有站点的图片，诱骗用户点进钓鱼网站，这里原本原网站处于点击按钮前面，而想办法跳转到钓鱼站点则是将原网站处于点击按钮后方，然而实际上这种方
  123px; top:123px;")，但其实没有什么意义，可以不管
- 拖拽劫持和数据窃取：浏览器支持Drag&Drop的API，比如JS的event.dataTransfer.getData('Text'))等接口，拖拽使操作更加简单。浏览器中的拖拽对象可以
  API支持下，这个攻击过程非常隐蔽，突破传统ClickJacking只能诱骗用户操作，不能获取数据的局限，有更大破坏力，所以一些浏览器直接不允许拖拽接口
- 手机触屏上的点击劫持，劫持用户触屏操作，而且手机为了节约屏幕空间，经常隐藏地址栏，更容易造成视觉欺骗

0x03 修复方式

首先要明确业务是否真的需要iframe嵌套，如无必要，不要使用，如要使用，最好同源。推荐以下方式中X-Frame-Options和JS防御结合

HTTP头部的X-Frame-Options标签

比较推荐这种修复方式，主流浏览器基本支持这个头部标签，可以在MDN文档上找到浏览器的支持情况

最推荐的方式：在HTTP头部上加上X-Frame-Options■DENY头部

其次推荐：在HTTP头部上加上X-Frame-Options■SAMEORIGIN头部，这时，站点间安全性就是同源站点页面中安全性最弱的页面了，如果某个页面可以被篡改，就失去

最后，X-Frame-Options: ALLOW-FROM uri这个头部目前还不被Chrome支持，若是用户群体大而非内部使用的，不推荐使用

JS防御

用JS判断当前页面是否被其他页面嵌套，如果是，跳转到自己的域名下

建议代码，目前没有绕过方式，暂时比较安全，首先隐藏页面，当确认没有被iframe包裹时显示，否则重定向

```
<style>
   body {display: none;}
</style>

<!-- ■■JS■■body■■■■■ -->
<script>
if (self == top) {
   document.getElementsByTagName("body")[0].style.display = 'block';
} else {
   top.location = self.location;
}
</script>
```

如果在网上搜索点击劫持修复方案，很多博客会给如下代码，不推荐使用，会被绕过

```
if (top.location != location) {
   top.location = self.location;
}
```

多因素认证

在重要功能按钮点击前加入多因素认证，如付款的二次密码或者某些隐私问题等。这种方式算很有效，但较麻烦，不重要的问题使用也会变复杂

不以cookie作为登录态

不推荐，cookie机制还是比较安全可靠的，自定义HTTP头部或者其他方式需要开发本身了解其他方面安全机制和浏览器策略来确保安全，没有必要

CSP策略

FireFox有"CSP"策略，但是不是所有浏览器支持，不推荐使用，以后若普及可以使用

0x04 深入攻防

前面说到JS防御点击劫持的方式，事实上，点击劫持刚被提出时，几乎没有站点做了防御，之后有很多站点使用JS的方式防御，但是JS的防御代码存在被绕过的可能性，前面

JS防御代码的主要绕过方式有4种

- JS接口函数的hook
- 检查代码的绕过
- 部分浏览器特性导致的绕过
- JS禁用

JS接口函数的hook

站点防御方式

```
if (top.location != location) {
   top.location = self.location;
}
```

方法一：onBeforeUnload事件触发：在即将离开当前页面(刷新或关闭)时执行自定义函数

攻击方式

```
<script>
   window.onbeforeunload = function() {
       return "■■5■■■■■";
   }
</script>
<iframe src="http://www.example.com">
```

跳转时诱惑用户点击留在此页，如下图所示，跳转操作就不会触发。此方法在IE浏览器上仍然有效，Chrome已经无效，应该做了相应的规则处理

方法二：重写top.location，此方法在IE和Chrome上都无效，某些版本Safari有效

攻击方式，重写top.location，使该接口无效

```
<script> var location = "clobbered";</script>
// ■■
<script>window.__defineSetter__("location", function() {});</script>

<iframe src="http://www.victim.com"></iframe>
```

检查代码的绕过

这种方式具有特殊性，需要站点对点击劫持的检查代码本身有逻辑错误，可以绕过，而绕过方式就是安全里比较常规的编码、特殊符号、大小写等fuzz手段

方法一：双重框架

当站点防御方式为

```
if (top.location != location) {
   parent.location = self.location;
}
```

使用双层框架，攻击者先嵌入一个页面，再在那个页面里嵌入受害站点，某些版本浏览器会在第一跳转后parent对象仍然不变，导致防御失败。时间有限我没有尝试

```
Attacker top frame:
<iframe src='hacker.com'></iframe>
Attacker sub-frame:
<iframe src='example.com'></iframe>
```

双层框架还有另一种情况，即比如微软允许google嵌套的页面，检查的top.location是google则放过，但google没有点击劫持的防御，那我们可以嵌套google的页面，则

方法二：referrer检查

有的站点后端对referrer检查，不符合就不返回响应包或者返回错误信息，一般使用正则匹配，不是专门有安全方面意识的情况下，写的匹配语句可能被绕过

方法三：top.location检查绕过

防御方式：

```
if (filter(top.location)) {
   top.location = self.location;
}
```

以上filter函数可能对站点名称做一些匹配，可以尝试使用fuzz的方式绕过，匹配规则不完全的情况下可以绕过

方法四：手机端一类站点绕过

简单讲就是example.com加了，但是和example.com功能一样的m.example.com没加，属于疏漏

部分浏览器特性的绕过

这部分实验比较复杂，且出现情况特定，了解一下就好，详情可以看论文

- IE8和Chrome引入了XSS过滤器，我们在可以写一段恶意代码，利用浏览器XSS过滤器的匹配规则，禁用iframe标签页面里的JS代码或者禁用部分代码，这段恶意代码的
- OnBeforeUnload-204冲刷，很多浏览器（IE7、IE8、Chrome、FireFox）可以让一个攻击者通过修改onBeforeUnload来重复提交location到204页面，不断冲刷会取
- IE对iframe有一个`security='restricted'`属性，Chrome对iframe有一个`sandbox`属性，这些可以用来禁用被嵌套页面的JS，导致防御代码无效，但IE的这个属性也
- IE和FireFox实现的`document.designMode`可以在框架页面中被开启，会禁用JS，可以传送Cookie

禁用JS

不管怎么用JS防御，被浏览器或者用户使用插件禁用JS就没有作用了

0x05 案例与总结

最后，使用facebook的一个案例做个总结

facebook知道JS代码可以被绕过被禁用的特性，为了真正防御点击劫持，在某个版本使用了以下的防御代码：这段代码添加了一个黑色9999*9999的像素块，50%透明，所

```
if (top != self){
   window.document.write("<div style='background: black;
                  opacity: 0.5;
                  filter: alpha(opacity=50); position: absolute;
                  top: 0px; left: 0px; width: 9999px; height: 9999px; z-index: 1000001'
                  onClick='top.location.href=window.location.href'</div>")
}
```

上面防御代码可以用以下方式绕过

```
<body style="overflow-x: hidden; border: 0px; margin: 0px;">
   <iframe width="21800px" height="2500px" src="http://facebook.com/" frameborder="0" marginheight="0" marginwidth="0"></ifram
   <script> window.scrollTo(10200, 0);</scirpt>
```

scrollTo function动态移动frame到中心，清除了点击会跳转的div，防御代码失效

精巧如facebook的代码仍然被绕过，再回头看我们推荐的防御方式

X-Frame-Options头部存在以下问题

- 改策略需要为每个页面指定，这可能会使部署复杂化，可以在一些重要页面部署，比如支付和登录，当然，框架或Web容器部署则无问题
- 代理在添加和删除头部方面是臭名昭著的，可能会去掉头部X-Frame-Options导致不能防护，就像通信通信给HTTP页面嵌广告，HTTPS则无问题

JS的防御代码：我们上面推荐的JS防御代码只是已知还未存在绕过方式的代码，特点如下

- 页面加载时，样式表会隐藏页面上所有内容，禁用js，页面保持为空
- 类似的，如果页面被框架，将保持空白或者frame bust
- frame bust代码被阻止，如被hook，页面会保持空白
- 该脚本仅显示内容页面不在框架中运行的文档
- 注意用户使用NoScript也会不显示页面，需要后备机制
- 实验中没有发现任何用户可感知的渲染上的延迟
- 是目前最安全的方案而不是一定没有问题的方案

建议重要功能还是加上多因素认证

注意漏洞修复方式，不要以为修复了实际上没有用，所以对漏洞和业务的理解都很重要，很多情况下需要安全去积极了解业务或者业务主动询问安全来确定更好的修复方案

0x06 参考资料

《白帽子讲Web安全》

《Busting Frame Busting:
a Study of Clickjacking Vulnerabilities on Popular Sites》

点击收藏 | 2 关注 | 1

上一篇：跟我一步一步审计.net程序 下一篇：CrySiS勒索病毒变种分析

1. 0 条回复
   - 动动手指，沙发就是你的了！