**Large-Scale Analysis of Style Injection by Relative Path Overwrite**

原文链接：<https://seclab.ccs.neu.edu/static/publications/www2018rpo.pdf>

本文发表在2018 International World Wide Web Conference（WWW '18），第一作者是来自美国东北大学的Sajjad Arshad。

1. **主要内容**

跨站脚本攻击一直是最常见的Web攻击手段，但近期有研究表明，在一定条件下，通过注入构造好的CSS样式指令同样可以达到脚本攻击的效果。本文利用RPO（Relative Path Overwrite）向站点中注入样式指令。RPO作为一种新型攻击手段，在此前没有任何关于它的学术研究，对于互联网上RPO漏洞的流行度也不得而知。为此，作者开展了基于RPO的样式注入的大规模研究来分析有多少站点可能会受到该攻击的影响。

最后，作者在Alexa排名前1万的网站中，发现约有9%的站点包含至少一个易受攻击的页面，而这其中又有1/3的站点可以被利用。针对基于RPO的样式注入，作者也提出了一些相对简单的防御手段。

1. **背景知识**
2. **浏览器解析模式**

浏览器在解析CSS时有两种模式：标准模式（Standard Mode）和怪异模式（Quirks Mode）。

**标准模式**：浏览器按照W3C标准解析CSS代码，浏览器仅会解析Content-Type：text/css的css资源文件。

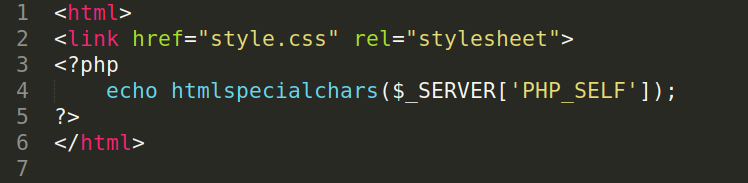
**怪异模式**：主要用于处理网络早期常见的一些编码不良的网站，浏览器使用自己的方式解析CSS指令，不同的浏览器解析执行代码的方式不一样，故称之为怪异模式。

浏览器在解析CSS时选择标准模式还是怪异模式取决于网页中的DTD（Document Type Definition，文档类型定义）声明。

1. **Relative Path Overwrite**

相对路径覆盖（Relative Path Overwrite）是一种新颖的技术，它利用了Web服务器和浏览器在解释相对路径时的差异，使得HTML页面能够将其自身内容作为结果返回给样式表，从而将样式指令注入到站点中。而浏览器在解析CSS资源时的不够严苛的特点将导致注入的样式指令被解析。RPO漏洞可能会出现在使用相对路径来引用css文件或js文件等资源的站点中。浏览器在向服务器请求该类资源之前，会将相对路径展开为绝对路径。

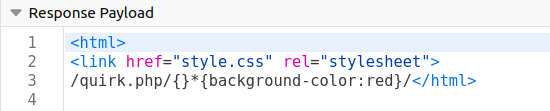
用一个例子来解释下基于RPO的样式注入。现有文件quirk.php如下图所示：



* 1. 攻击者发起请求：

[http://localhost/quirk.php/{}\*{background-color:red}/](http://localhost/quirk.php/%7b%7d*%7bbackground-color:red%7d/)

* 1. 服务器返回与quirk.php对应的HTML文档，内容如下图所示：



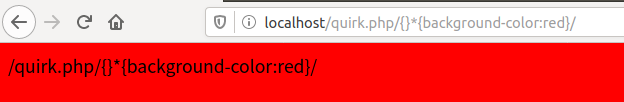
* 1. 该页面中包含一个用相对路径引入的css资源文件，浏览器会扩展该css文件路径，并向服务器发起请求，请求url为：

[http://localhost/quirk.php/{}\*{background-color:red}/style.css](http://localhost/quirk.php/%7b%7d*%7bbackground-color:red%7d/style.css)

* 1. 该路径并不存在对应的css文件，服务器认为{}\*{background-color:red}/style.css为quirk.php的相关参数，故返回如下图所示style.css文档，可以看到quirk.php将自身内容作为结果返回到style.css文件中：



* 1. 该css资源对应的Content-Type为text/html，在怪异模式下，浏览器会忽略该声明，将文档中的任何css指令解析执行，最后的效果如图：



1. **设计与实现**

根据上述原理，作者设计了实验，主要分为三个步骤：

1. **Candidate Identification**

第一步是数据集的筛选，原始的数据集是来自于Common Crawler网站在2016年8月爬取下来包含1.6亿个网页的存档。对于该数据集，作者进行了一些筛选：

1. 用一个Java HTML parser来过滤一些仅仅包含内联CSS或CSS是用绝对路径嵌入的网页；
2. 为了实验的可测量性，作者进行了进一步的筛选，分为两个步骤：
   1. 仅保留Alexa中排名前100万的站点页面，这样能让评估的结果更加倾向于流行的站点；
   2. 第二步是根据页面的url进行分组处理，将那些具有相似url和相同文档类型的页面分成一组，在后面的测试时仅从其中选取一个代表页面。

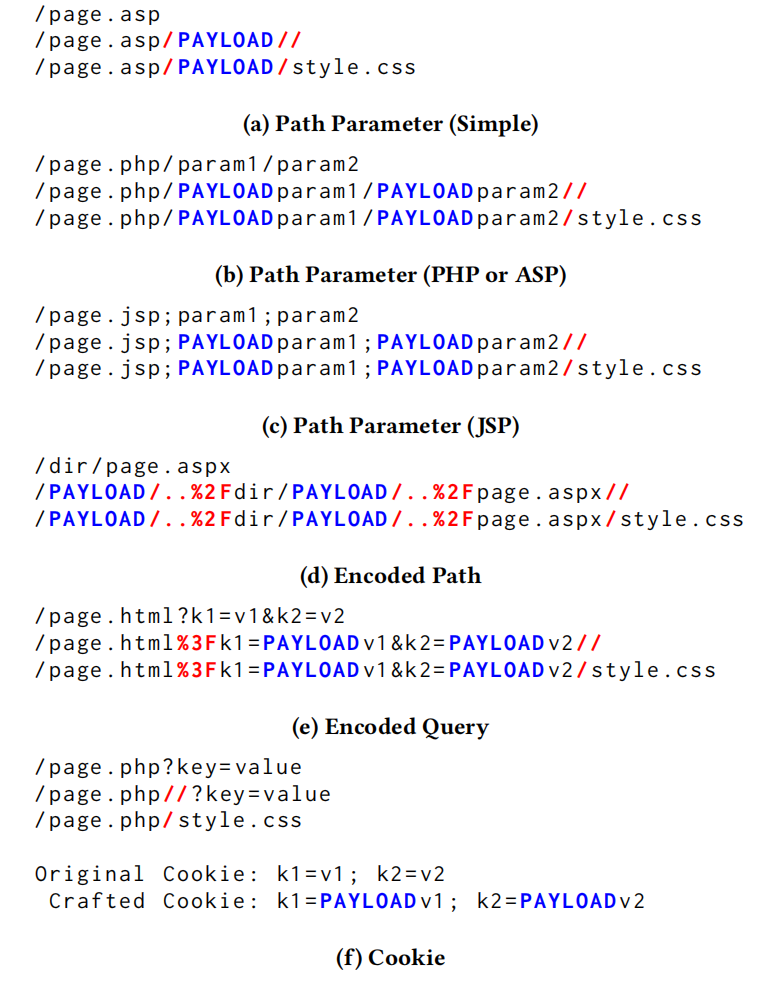
经过层层筛选，最后的数据集是来自222k站点的127m个页面。

1. **Vulnerability Analysis**

这一步的目的是找出数据集中所有存在潜在风险的网页，主要分成以下步骤来进行：

1. 实现一个基于Python Requests模块的轻量的crawler；
2. 从上一步数据集筛选分组后的页面中选出一个代表页面，crawler根据该页面的url特点，通过正则匹配，构造一个具体的、唯一的url；
3. 请求每一个唯一的包含样式指令的url，搜索服务器返回的response中是否包含<base>标签（规定页面中所有相对链接的基准URL），如果包含则认为该页面是安全的，反之，提取出网页中所有相关的样式表路径，构造能造成路径混淆的url，并以此为基础展开样式表请求路径；
4. 检查服务器返回的响应中是否存在注入的样式指令，如果有，则该页面是一个Vulnerable页面。

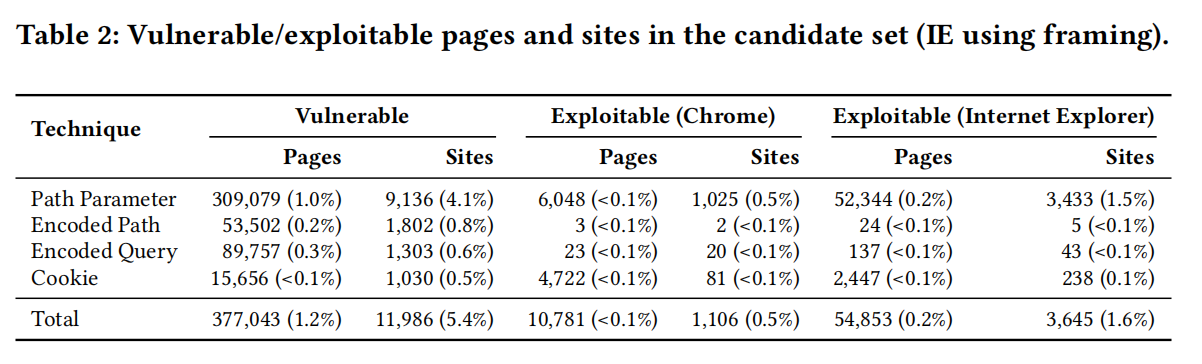
下图是实验中所用的payload，其中PAYLOAD=**%0A{}\*{background:NONCE}**，其中NONCE是随机生成的一串字符串，用来唯一标识一个样式指令。



1. **Exploitability Analysis**

经过上一步的测试，还需要在真实的浏览器上检查这些攻击是否能够生效，为了实现这个目的，作者设计了如下实验：

1. 建立一个基于Google Chrome的crawler，用Remote Debugging Protocol驱动浏览器并记录相应的HTTP请求和响应；
2. 向从上一步中选取的Vulnerable Page中注入样式指令，注入的指令使用随机生成的图像url来为HTML页面加载背景图像，通过检查是否存在向外部请求图像的请求流量来判断样式指令注入是否成功；
3. 在IE浏览器中重复此实验，IE浏览器允许frame中的页面继承父页面的doctype，父页面会将子页面的渲染模式强制转化为quirks mode。针对这个问题，作者在本地创建了页面，在这些本地页面中用iframe来加载这些victim page，检查注入的指令是否被解析；
4. 为了避免可能会带给网站的破坏性副作用，不会进行网站登录、表单提交，按钮点击等操作。
5. **实验评估**



实验发现，在所有数据集中，有大约1.2%的页面时存在潜在风险的（Vulnerable），受到影响的站点占4.5%。通过向Path Parameter中注入payload是最有效的方式。

根据实验结果，作者探讨了一个Vulnerable页面被成功利用的因素：

* 1. **Document Type**

Document Type在基于RPO的样式注入攻击中发挥着重要的作用，浏览器渲染页面时选择standard mode还是quirks mode与文档类型声明直接相关，错误的doctype声明拼写或是老旧的doctype都会导致浏览器进入quirks mode。

* 1. **Internet Explorer Framing**

IE浏览器会使frame中的子页面继承父页面文档类型的特性使得Vulnerable页面在IE浏览器中被成功利用的数量是在Chrome浏览器的5倍之多。

* 1. **Anti-Framing Technique**

因为IE浏览器中存在的framing问题，作者又探讨了Anti-Framing技术，与之相关的http响应头字段时X-Frame-Options。浏览器会根据该字段值来决定是否要加载一个iframe中的页面，为了防止rpo攻击，应当将这个字段的值设置为deny。

在Vulnerable Pages中有4999个页面正确使用了该报头，从而防止了攻击。

* 1. **MIME Sniffing**

为了防止浏览器通过MIME Sniffing特性将文档的Content-Type识别为text/html，从而导致在quirks mode下文档中的css指令被解析，应当将X-Content-Type-Options的值设为nosniff来禁止浏览器进行类型嗅探。

针对页面中存在的RPO漏洞，作者提出了一些防御手段，如在页面中使用<base>标签，利用doctype将文档类型声明为标准模式等，但最有效的方式莫过于直接使用绝对路径引入css等资源文件。

1. **总结评价**

本篇论文中的实验所用技术并不复杂，实验方法就是利用爬虫来对每个页面进行分析，但本篇论文的亮点就在于它展示了现实网络站点中可能会受到的RPO攻击影响的一种情况。

RPO作为一种新型的Web攻击方式，本文的作者是利用它来进行样式指令注入，css的语法要求没有那么严格，允许很多脏字符的存在，并且即使有不符合css语法的地方，也会被浏览器忽略，继续向下执行，直到找到合法的css语句。与之相比，注入js脚本就困难得多了，因为js有更严格的语法要求，并且不能包含脏字符，但同时，如果能找到一种注入js脚本的方式，造成的危害也会更大，这也可以成为我们未来研究的方向。