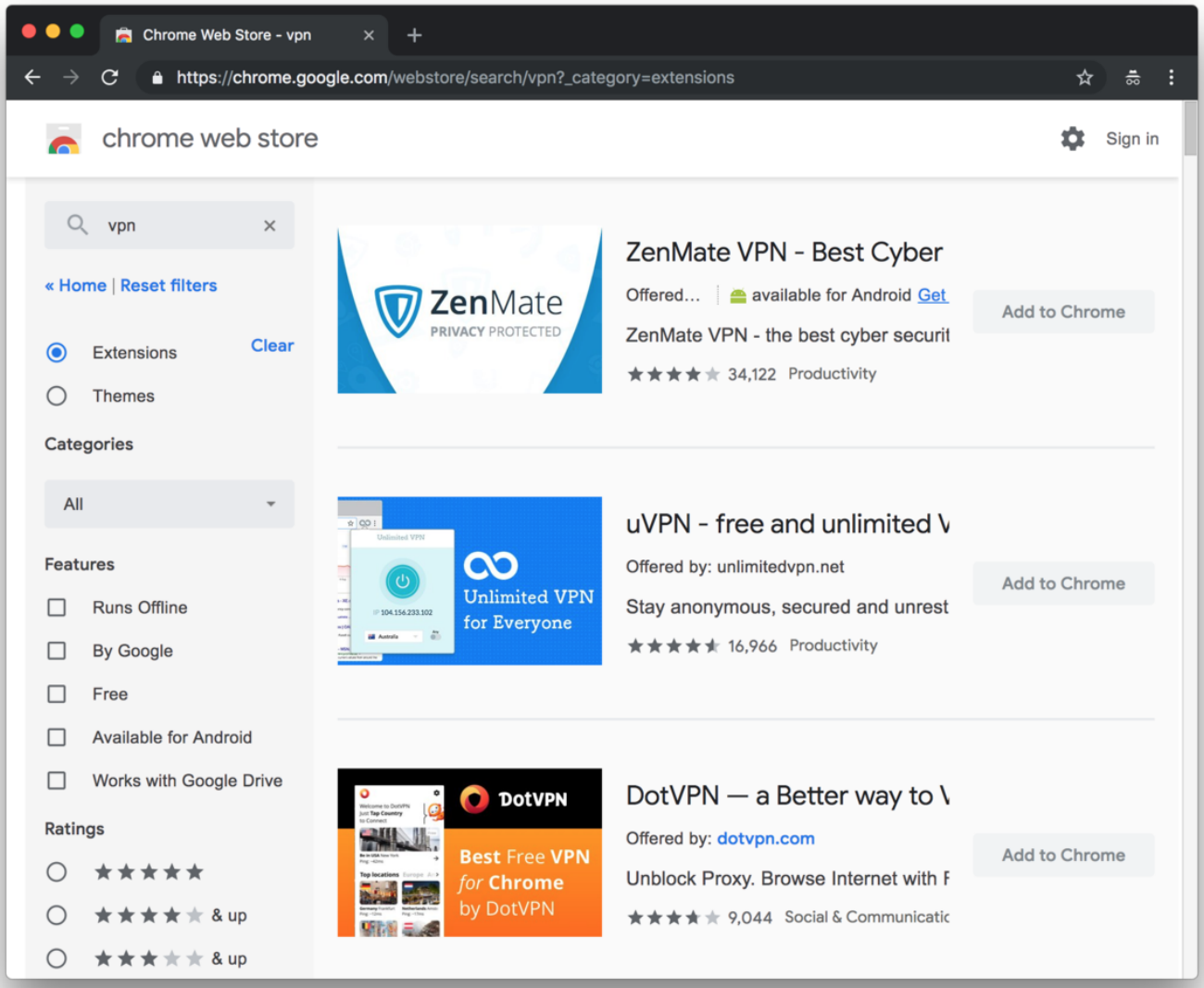
登录

VPN扩展功能的隐私安全问题

Pinging / 2019-03-13 08:14:00 / 浏览数 2147 技术文章 翻译文章 顶(0) 踩(0)

■■■2018■■■■■■■■■■■■■■■■■■■■■

我们不知道VPN的扩展功能是何时流行起来的，但VPN扩展实际上应该被称为代理扩展。其底层不只是VPN还包括代理服务，但VPN生产商却声称其具有很强的安全性与私



经过几次VPN扩展的测试和研究工作，我得出结论，几乎所有的VPN扩展都容易受到IP泄漏和DNS泄漏的影响。
具有讽刺意味的是，尽管大多数漏洞的原因都出自于扩展的错误配置，但浏览器也需要对此负责，因为代理的配置重存在很多陷阱和误导性文档。

PAC脚本

Chrome和Firefox都提供了用于注册PAC（代理自动配置）脚本的扩展的API。
它是一个`JavaScript`文件，它公开了一个函数`FindProxyForURL■url■host■`，它指示浏览器是否应该将请求转发到代理服务器。
还提供辅助功能以达成条件。 我将在以下内容中介绍有关滥用PAC脚本的常见问题。

拆分通道

常见的VPN扩展会尝试解析请求的主机名，并允许私有地址绕过代理。 这允许用户同时访问内部网络以及代理的因特网。

```
function FindProxyForURL(url, host) {
 let ip = dnsResolve(host);
 if (isInNet(ip, "172.16.0.0", "255.240.0.0"))
   return "DIRECT";
}
```

然而，如果不将DNS泄漏，那么我们无法不可能实现这个功能。
由于调用了dnsResolve，因此将使用本地DNS服务器对每个请求进行DNS查询，本地DNS服务器默认为ISP提供。 这需要以下条件：

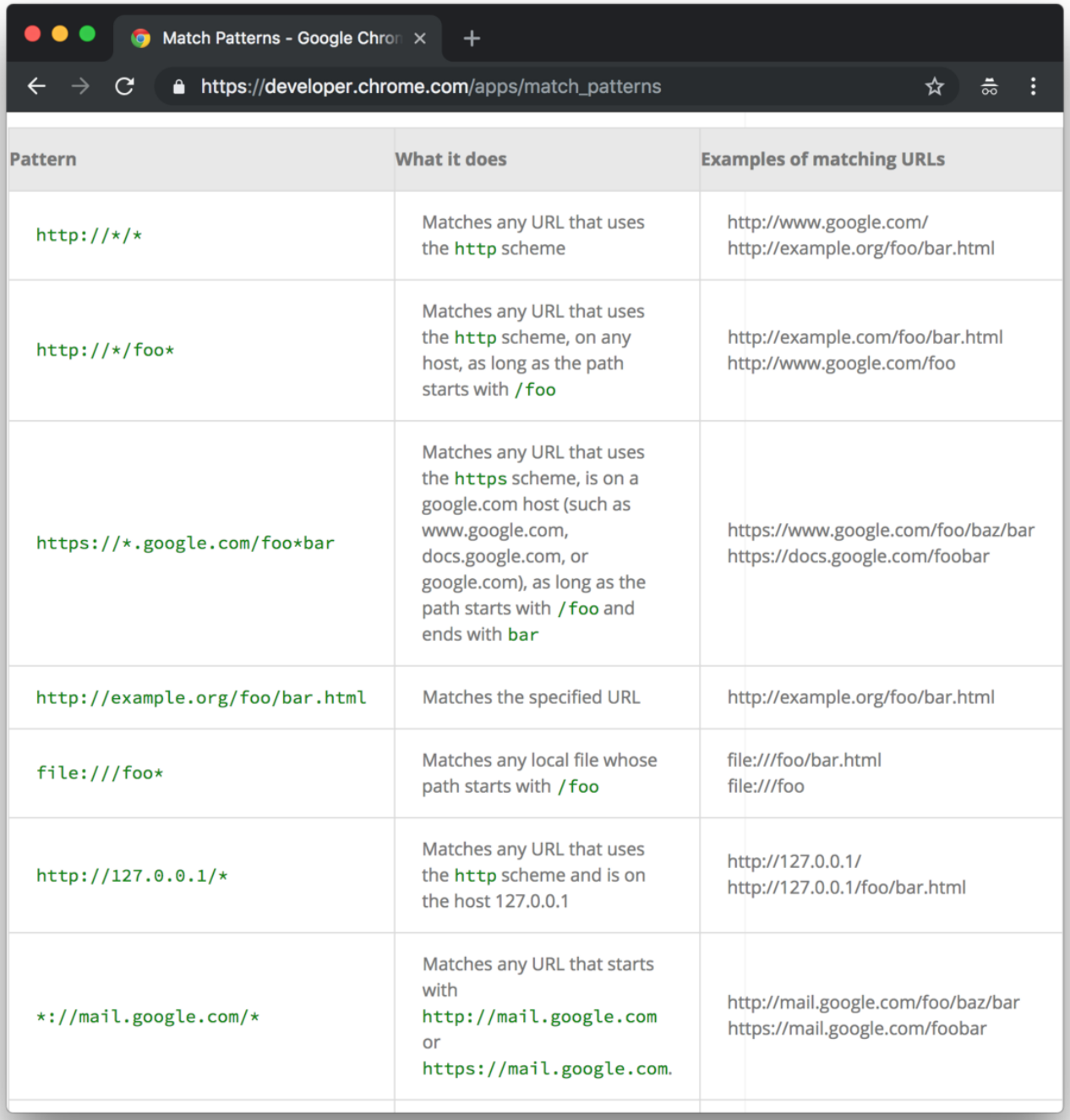
- 一个网站，用于识别用户正在使用的ISP。
- 一个路径上的窃听者（例如ISP），以查看用户正在访问的网站。

辅助函数的错误使用

另一个非常常见的问题是扩展误解了辅助函数的工作原理。

```
function FindProxyForURL(url, host) {
 if (shExpMatch(url, "*://api.vpn.com/*") ||
   shExpMatch(host, "192.168.*.*") ||
   dnsDomainIs(host, "vpn.com") ||
   isPlainHostName(host)
 )
   return 'DIRECT';
}
```

在Chrome中，有一种称为匹配模式的功能，其用于定义扩展程序的URL。 它使用URL格式的通配符。



Pattern	What it does	Examples of matching URLs
`http://*/*`	Matches any URL that uses the `http` scheme	http://www.google.com/ http://example.org/foo/bar.html
`http://*/foo*`	Matches any URL that uses the `http` scheme, on any host, as long as the path starts with `/foo`	http://example.com/foo/bar.html http://www.google.com/foo
`https://*.google.com/foo*bar`	Matches any URL that uses the `https` scheme, is on a google.com host (such as www.google.com, docs.google.com, or google.com), as long as the path starts with `/foo` and ends with `bar`	https://www.google.com/foo/baz/bar https://docs.google.com/foobar
`http://example.org/foo/bar.html`	Matches the specified URL	http://example.org/foo/bar.html
`file:///foo*`	Matches any local file whose path starts with `/foo`	file:///foo/bar.html file:///foo
`http://127.0.0.1/*`	Matches any URL that uses the `http` scheme and is on the host 127.0.0.1	http://127.0.0.1/ http://127.0.0.1/foo/bar.html
`*://mail.google.com/*`	Matches any URL that starts with `http://mail.google.com` or `https://mail.google.com`.	http://mail.google.com/foo/baz/bar https://mail.google.com/foobar

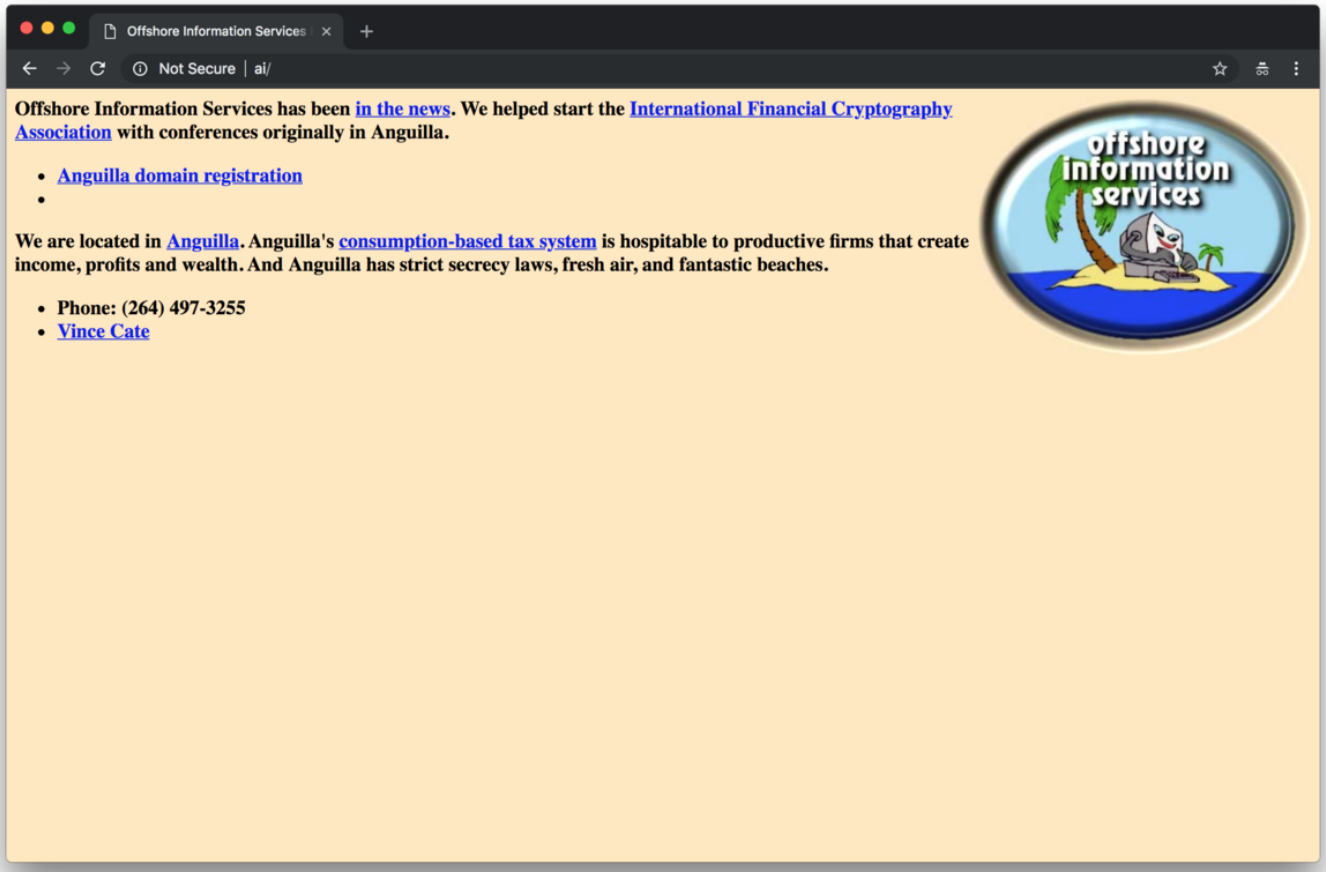
当然，开发人员认为shExpMatch应该以相同的方式工作，因为它也支持相同的通配符。 但是由于它不支持URL，所以表达式与匹配模式有所不同。
例如，http://evil.com/://api.vpn.com/绕过代理，因为它匹配表达式*://api.vpn.com/*。
同样，192.168.evil.com也能过绕过代理，因为它的主机名匹配192.168.*.*。 网站可以通过让浏览器向这些URL发出请求来泄露用户的IP地址。

下面是dnsDomainIs的表述：

■■■■■■■■■■■■■■■true■

类似于函数直接的比较，我们期望看到两个参数相等两个参数相等。 事实上，一些例子也表明情况就是如此。 描述中实际上称为主机名的只是子域部分。
例如，dnsDomainIs("api.vpn.com", "vpn.com")返回true，因为api.vpn.com是vpn.com的子域。
仅此一项不会引入任何安全问题，但Chrome有一个预期的实现错误，它只匹配尾部。 这允许攻击者注册域evilvpn.com以传递dnsDomainIs(host, "vpn.com")并泄漏用户的IP地址。

对isPlainHostName的分析也是相当有趣。 当主机名不包含点时，它返回true。 没有点的主机名表示它属于内部网，因此让它绕过代理似乎是合理的。
除了并非总是如此。 某些顶级域名（如http://ai）可通过互联网访问，因此可以绕过代理。
幸运的是，利用这一种方法进行攻击是不可行的，因为攻击者需要拥有TLD。
值得一提的是，Chrome更进一步排除了IPv6地址，因为它们也是无点（例如[::1]），这会带来另一个绕过方法。



弱匹配

另一个常见问题是扩展不使用提供的辅助函数。 这可能是由于开发人员不了解提供的帮助程序功能或Firefox不支持它们。

通常可用于PAC文件的全局会使函数（isPlainHostName()，dnsDomainIs()）不可用。

在许多情况下，本机JavaScript函数可以直接使用，也可以作为polyfill使用。

```
function FindProxyForURL(url, host) {
 if (host.indexOf("localhost") !== -1 ||
   /^127\./.test(host) ||
   isPlainHostName(host) ||
   url.substring(0, 4) !== 'http'
 )
   return 'DIRECT';
}
function isPlainHostName(host) {
 return host.search('\\.') === -1;
}
```

尝试将某些主机名列入白名单的扩展程序很常见，但我们并不了解起准确方式。
例如，他们只在主机或主机的开头查找子字符串（127.localhost.evil.com传递host.indexOf("localhost") !== -1和/^127\./.test(host)）。
有时，会导致RegExp错误（例如，不会转义.）。

如前所述，Firefox不支持辅助函数。 因此，Firefox扩展必须为isPlainHostName等函数实现polyfill。
看起来，根据文档显示，它只需要检查主机名是否为无点。 他们解决的是上述IPv6问题。
在这里，攻击者可以通过让浏览器向IPv6主机发出请求来泄漏用户的IPv6地址。

有时，扩展不希望处理非HTTP流量，因此它们允许不以http开头的URL绕过代理（url.substring(0, 4) !== 'http'）。
这中方法为网站提供了通过强制其浏览器发出非HTTP请求来泄漏用户IP地址的机会。 它们可以是FTP（ftp://）和WebSocket ws://■wss://）。

主机名白名单

一对扩展程序拥有代理绕过的白名单。 它们通常是公司的域（*
.vpn.com），DNS环回服务（例如http://lvh.me），Google服务和带宽密集型服务（例如CDN和流媒体站点）。 访问白名单网站的用户将泄露其IP。

未加密的代理协议

某些扩展使用被认为不安全的协议。

```
function FindProxyForURL(url, host) {
 return "PROXY http.vpn.com; HTTP http.vpn.com; SOCKS socks4.vpn.com; SOCKS4 socks4.vpn.com; SOCKS5 socks5.vpn.com;";
}
```

PAC脚本支持四种代理协议。 HTTP（代理和HTTP），HTTPS，SOCKS4（SOCKS和SOCKS4）和SOCKS5。
由于TLS、HTTPS隧道是安全的，但是而HTTP和SOCKS不支持加密。 这意味着路径上的窃听者可以轻松拦截流量，就像没有VPN或代理一样。

DNS预取技术

Chrome使用了一种DNS预取的技术：

> DNS预取是尝试在用户关注链接之前解析域名。 这是使用计算机的正常DNS解析机制完成的。没有使用Google的连接。

Chrome会自动为以下网址预取DNS：

- 多功能框中的项目类（地址栏）
- HTTP页面中的超链接或选择DNS预取的站点

最重要的是，即使启用了代理，默认情况下也会启用此功能，如下所示。

这会影响使用PAC脚本的扩展功能，并且会导致DNS泄漏。 Opera的内置VPN也受到影响。

更新：所有Chrome VPN扩展都受到影响

唯一的缓解措施是用户手动禁用此功能：

1 导航到chrome://settings/

2 在“搜索设置”中输入“预测”

3 禁用选项“使用预测服务来帮助完成在地址栏中输入的搜索和URL”和“使用预测服务更快地加载页面”

服务器修复

除了PAC脚本之外，Chrome还允许扩展程序设置固定代理服务器。 这类似于PAC脚本，只有return语句。 它确实支持使用匹配模式的简单绕过列表。

错误文档

绕过列表的文档指出：

> 匹配本地地址。 如果主机是“127.0.0.1”，“:: 1”或“localhost”，则地址是本地的。
> 示例：“<local>”</local>

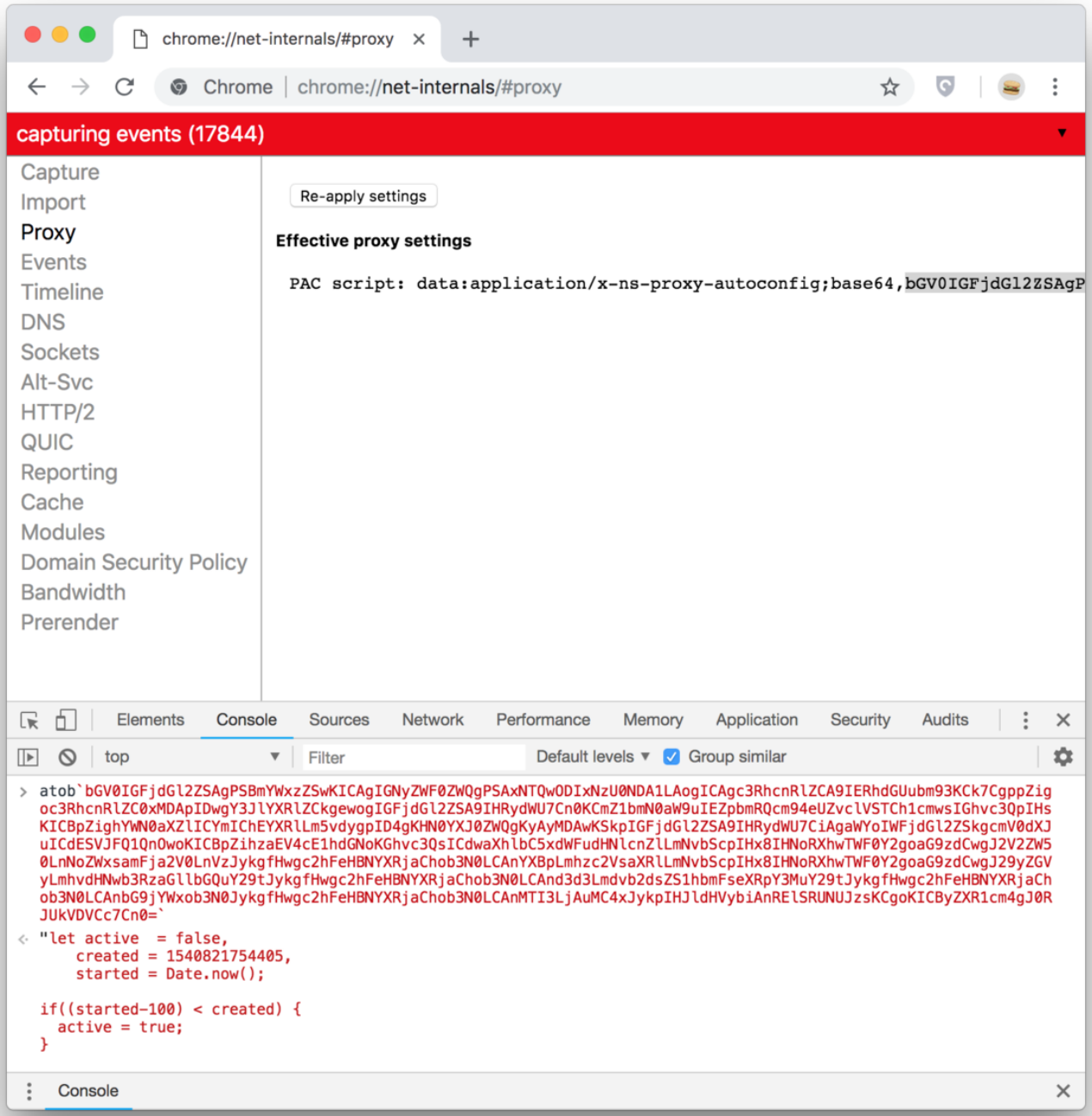
因此，这个列表使得环回地址绕过代理非常简单。浏览Chromium的源代码显露出其余的内容：

```
class BypassLocalRule : public ProxyBypassRules::Rule {
public:
 bool Matches(const GURL& url) const override {
   const std::string& host = url.host();
   if (host == "127.0.0.1" || host == "[::1]")
     return true;
   return host.find('.') == std::string::npos;
 }

 std::string ToString() const override { return "<local>"; }

 std::unique_ptr<Rule> Clone() const override {
   return std::make_unique<BypassLocalRule>();
 }
};
```

·Matches·方法不仅为环回地址（127.0.0.1和[:: 1]）返回true，而且还返回任何没有.的主机名。 这与isPlainHostName完全相同。 这也导致了IPv6的泄漏。

自身检查

在Chrome中，我们可以访问chrome://net-internals■proxy查看有效的代理设置。
要提取PAC脚本，复制base64之后的所有内容，并在DevTools控制台中运行atob■“PASTE_HERE”■。 除了提取源代码之外，Firefox没有简单的方法。



结论

本文中提到的事件是由于VPN供应商以及浏览器设置不协调导致的。我向受影响的各方报告了这些问题，但没有太大进展。

在我看来，VPN扩展非常适合用来绕过geoblocking，但对于匿名性和隐私性来说，这不是一个很好的处理方法。

■■■■■■■■■■■■https://blog.innerht.ml/vpn-extensions-are-not-for-privacy/

点击收藏 | 1 关注 | 1

上一篇：从5道堆题看堆利用 下一篇：从5道堆题看堆利用

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