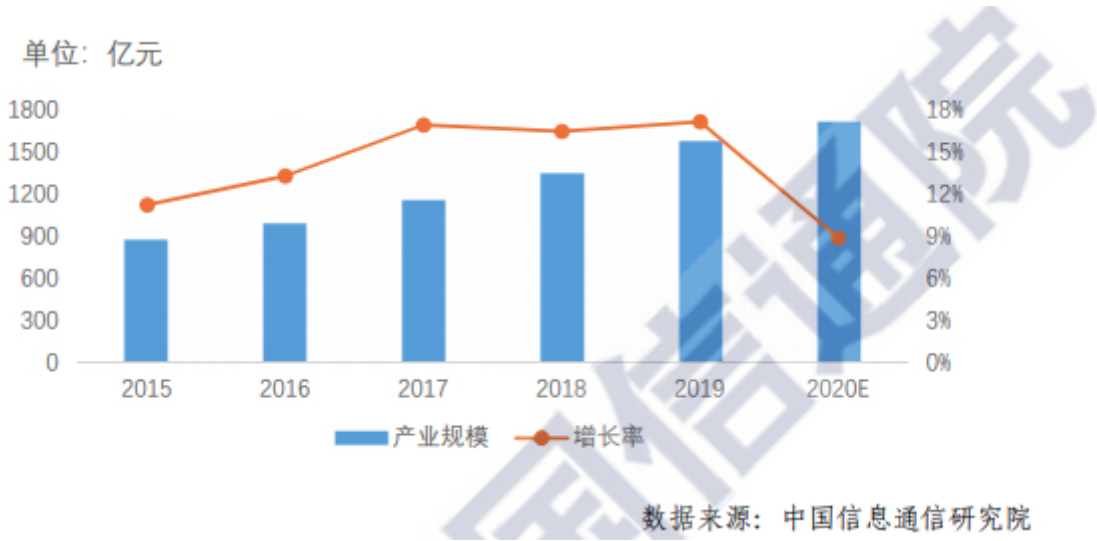
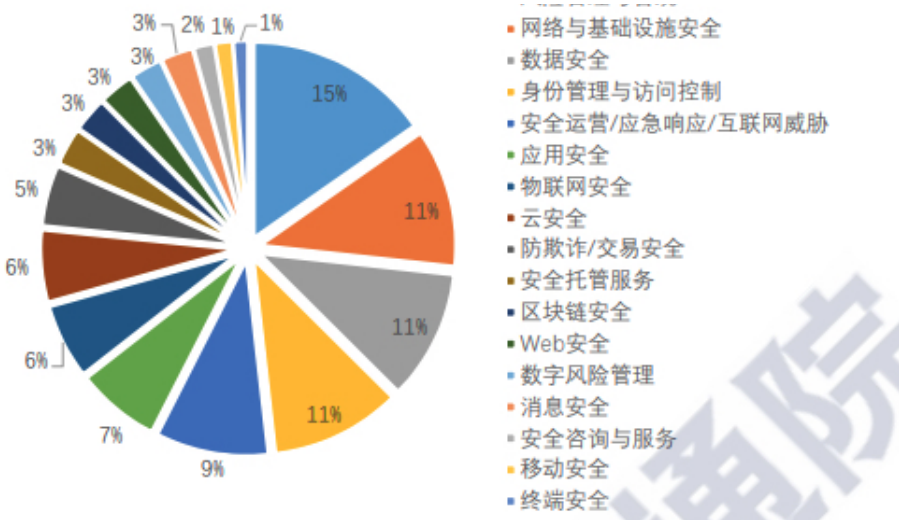
浏览器指纹及其应用

背景

伴随互联网的快速发展，Web 服务已呈现出极速增长的趋势， 人们在使用 Web 服务的同时，也面临着网络安全风险。根据中国信息通信研究院发布的《中国网络安全产业白皮书》报告：我国网络安全产业呈现高速增长态势，2020 年产业规模预计为 1702 亿元。



其中，Web 安全也占有一定比重，Web 站点俨然已经成为网络攻防的重要战场。



数据来源：Momentum Cyber

图 11　2019 年全球网络安全融资领域分布情况

而浏览器指纹在 Web 安全**主动防御**领域扮演着越来越重要的角色。本文主要从浏览器指纹概述、指纹识别原理、应用场景三个方面来介绍浏览器指纹。

浏览器指纹概述

浏览器指纹是由**电子前哨基金会（EFF）**首席科学家 **Peter Eckersley** 在 2012 年提出，它利用浏览器自由传输的一些属性来生成和人类指纹一样具有标识作用的字符串。那么常见的浏览器指纹有哪些呢？可以参考如下表格数据，其中：

- 指纹因子：指浏览器对外的公开属性，如 `userAgent` 可以通过 `navigator.userAgent` 获取；`colorDepth` 可以通过 `screen.colorDepth` 获取。

```
> navigator.userAgent
< "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
> screen.colorDepth
< 24
```

- 稳定性：指刷新浏览器而指纹因子对应的取值不会发生改变。如 `colorDepth` 它表示屏幕的颜色深度，在谷歌浏览器中其取值为 24，刷新浏览器后其取值依然是 24。那么它就是**稳定的**指纹因子。
- 独立性：指同一设备上使用不同浏览器，指纹因子的取值不会发生改变。如 `devicePixelRatio` 它表示当前显示设备的物理像素分辨率与 CSS 像素分辨率之比，其取值在同一设备上的谷歌浏览器、火狐浏览器、Edge 浏览器、IE 浏览器、Opera 浏览器和 Safari 浏览器均为同一个值。因此，`devicePixelRatio` 是**独立的**指纹因子。

指纹因子	稳定性	独立性

指纹因子	稳定性	独立性
userAgent	稳定	否
language	稳定	是（大多数时候）
colorDepth	稳定	是
deviceMemory	稳定	是
pixelRatio	稳定	是
hardwareConcurrency	稳定（但是 IE 浏览器不支持）	是（但是 IE 浏览器不支持）
screenResolution	稳定	是
availableScreenResolution	稳定	是
timezoneOffset	稳定	是
timezone	稳定	是
sessionStorage	稳定	否
localStorage	稳定	否
indexedDb	稳定	否
addBehavior	稳定	否
openDatabase	稳定	否
cpuClass	稳定	是
platform	稳定	是（大多数时候）
doNotTrack	稳定	否
plugins	待定	否
canvas	稳定（大多数时候）	否（已经过实际验证）
webgl	稳定（大多数时候）	否（已经过实际验证）
adBlock	稳定（取决于使用时间）	否
hasLiedLanguages	稳定	是
hasLiedResolution	稳定	是
hasLiedOs	稳定	是
hasLiedBrowser	稳定	是

指纹因子	稳定性	独立性
touchSupport	稳定	是
fonts	稳定（大多数时候）	否（大多数时候）
audio	未知	否

> 表格指纹因子数据源：**Browser independent components** & **Stable components**

以上的这些指纹因子我们都可以通过 JavaScript 代码获取其对应取值，通过稳定且独立的指纹因子组合成的浏览器指纹，使得设备识别变得有可能。即通过浏览器指纹我们获取该指纹所在的硬件设备，如个人浏览器或 PC。

你可能会好奇，这些指纹因子可以用来干什么呢？它是如何实现与设备关联的呢？接下来让我们了解一下浏览器指纹识别的背后的原理。

识别原理

信息熵

上面我们介绍了浏览器常见的指纹因子，但是如何度量浏览器中这些可以自由传输的指纹因子呢？了解过信息论的同学应该知道，我们可以通过信息熵来度量信息量，信息熵越高，则能传输越多的信息，熵越低，传输的信息越少。因此，可以将信息熵作为浏览器指纹标识程度的判据，举例来看，对于一个离散型随机变量 X，它的熵 H(X) 为：

$$H(X) = -\sum_{x} p(x) \log_2 p(x)$$

单个指纹因子熵值

上述公式中，使用了以 2 为底的对数函数，其单位为比特。简单来讲，一个指纹因子的信息量可以通过它来量化。举个例子🌰：我们以指纹因子 `doNotTrack` 为例，它的取值结果有两种：

- 开启设置可标记为 1，关闭设置可标记为 0
- 假设用户访问网站的统计结果中，设置了 `doNotTrack` 的概览为 10%，未设置的概览为 90%。那么 `doNotTrack` 这一指纹因子对应的熵为：

$$H(X) = -\frac{1}{10} * \log_2^{\frac{1}{10}} - \frac{9}{10} * \log_2^{\frac{9}{10}} = 0.469bit$$

> Note：点击**信息论基础**进行穿越复习。

多个指纹因子概率

知道了如何计算单个指纹因子的信息熵，我们就可以进一步对浏览器的多个指纹因子进行信息熵的合并计算。根据 **Peter Eckersley** 发布的论文 **How Unique Is Your Web Browser** 结果表明：通过 `userAgent`、`fonts`、`screenResolution` 和 `plugins` 等八个指纹因子来生成浏览器指纹:

Variable	Source	Remarks
User Agent	Transmitted by HTTP, logged by server	Contains Browser micro-version, OS version, language, toolbars and sometimes other info.
HTTP ACCEPT headers	Transmitted by HTTP, logged by server	
Cookies enabled?	Inferred in HTTP, logged by server	
Screen resolution	JavaScript AJAX post	
Timezone	JavaScript AJAX post	
Browser plugins, plugin versions and MIME types	JavaScript AJAX post	Sorted before collection. Microsoft Internet Explorer offers no way to enumerate plugins; we used the PluginDetect JavaScript library to check for 8 common plugins on that platform, plus extra code to estimate the Adobe Acrobat Reader version.
System fonts	Flash applet or Java applet, collected by JavaScript/AJAX	Not sorted; see Section 6.4.
Partial supercookie test	JavaScript AJAX post	We did not implement tests for Flash LSO cookies, Silverlight cookies, HTML 5 databases, or DOM globalStorage.

Table 1. Browser measurements included in Panopticlick Fingerprints

其统计结果显示这 8 个指纹因子至少包含 18.1 bit 的信息熵，通过自信息计算公式：

$$I(e) = -\log_2 p(x)$$

我们可以推导出，其出现同一个指纹所需要的概率：

$$p(x) = 2^{-I(e)}$$

通过计算可以得出：随机挑选一个浏览器，在 286777 个浏览器中才会出现与其相同的一个浏览器指纹。从而说明**浏览器指纹具有非常高的标识性**，而且伴随着指纹因子维度的增加，出现同一个浏览器指纹的概率会越来越低，其识别准确率也会越来越高。

小结

综上，我们可以通过计算出浏览器指纹因子对应的熵值，再结合自信息计算其出现概率，从而根据指纹取值和其概率推断出所在设备，了解了浏览器指纹生成原理。那么你也应该想了解它的应用场景有哪些吧，我们接着看。

应用场景

浏览器指纹常见的应用场景包含：

恶意爬虫的主动防御

目前针对 Web 站点，网络上有各种自动化爬虫，进行数据采集，常见如刷票、点评、爬取隐私数据等，现阶段主要防御措施有：被动检测与防御、 IP 地址检测与封锁、浏览器指纹&主动防御等。

扫描器识别与拦截

除了网络上等爬虫工具，还会有各种专业的 Web 漏洞扫描工具，如：**AWVS**、**APPScan**、**Xray** 等，扫描器在主动识别 Web 漏洞的同时，也会给线上 Web 服务带来潜在安全问题。针对扫描器的拦截，可以根据其浏览器指纹的特异性，在其扫描指定站点时，进行识别与拦截。

Web 攻击的追踪与溯源

在我们的浏览器上运行着各种各样的脚本，利用浏览器脚本对 Web 攻击进行溯源有绝对优势，可以通过浏览器指纹特征信息提取和关联，对攻击者进行追踪与溯源。

个性化广告推送

当你在网上浏览或者查找相关商品时，是不是经常会出现与之相关的广告页面？这时你很有可能正在被跟踪，因为你的屏幕分辨率、时区和表情符号集这些信息都暴露在了网上。甚至可以在你使用无痕浏览模式时跟踪你，有兴趣可以查看这里。

以上是目前浏览器指纹涉及的部分应用场景，从中我们可以看出：浏览器指纹既可以是广告方推送的利器，也可以是防御方识别的判据。

总结

本篇文章主要从浏览器指纹概述、指纹识别原理和应用场景三个方面带领大家认识了浏览器指纹。针对常见的浏览器指纹进行了介绍，对浏览器指纹和设备的关联性进行了原理解析，最后简要介绍了目前浏览器指纹的一些使用场景。下一篇文章中，我们将设计一个简单的指纹追踪模型，通过实战，了解浏览器指纹的用途。敬请期待。

参考链接

- xprize
- fingerprintjs
- 信息论基础
- How do trackers work?
- 基于 50 万个浏览器指纹的新发现
- How Unique Is Your Web Browser?
- Learn how identifiable you are on the Internet
- Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints