这一次,让我们再深入一点 – HTTP的客户端识别

> 这是网络系列的第八篇文章,接下来会有更多精彩内容.敬请期待! 让我们一起乘风破浪!

前言

HTTP作为一个无状态的请求响应协议，几乎没有什么信息用来判断是哪个客户端，也无法记录用户的访问记录。随着业务增长，现在的服务器希望能记录客户端的信息，以便提供个性化服务（收集用户数据）。今天一起来了解下常用的客户端识别方式：

- HTTP首部
- 客户端IP地址
- 用户登录
- 胖URL
- Cookie

HTTP首部

HTTP首部识别，是根据请求首部的相关信息来获取客户端信息。下面是常用的首部：

首部名称	类型	描述
From	请求	用户E-mail地址
User-Agent	请求	用户浏览器软件
Referer	请求	用户是从这个链接跳转过来的
Authorization	请求	用户名和密码
Client-IP	扩展请求	客户端IP地址
X-Forwarded-For	扩展请求	客户端IP地址(详情看这里)
Cookie	扩展请求	服务器产生的ID标签

- `From`首部记录了每个用户的E-mail地址，理想情况下，这个地址可以作为识别用户的标志。但为了防止服务器收集该信息用于垃圾邮件的散发，很少浏览器会发送From首部。
- `User-Agent`首部是用户浏览器的相关信息，可以用来制定和特定浏览器的交互动作，但它并不能用来识别特定的用户信息。
- `Referer`首部提供了用户来源页面的URL。只是标识了用户之前访问了哪个页面。

客户端IP地址

`IP地址`作为用户标识具有一定的前提条件：每个用户有着不同的IP，很少发生变化。但通常情况下，IP地址描述的是一台机器，而不是一个用户；IP地址也会在用户登录服务商时动态获取；服务器可能看到的是HTTP代理的IP地址（这种情况下Client-IP和X-Forwarded-For首部保存了原始IP地址）。

用户登录

`用户登录`，在用户访问需要授权才能访问的网站时，服务器会要求其登录。用户在输入用户名和密码后，浏览器可以通过`WWW-Authentication`首部将相关信息加密后发给服务器，这样服务器就可以标识当前用户。当然，该方式带来一个麻烦就是，用户在访问不同的网站时，需要多次输入不同的用户名和密码。

胖URL

有些服务器会为每个用户生成特定版本的URL，来识别不同的用户身份。通常，会对真正的URL进行扩展，在URL中添加状态信息；这些包含了用户信息的URL就称为胖URL。但该种方式也存在一些问题：

- URL和用户关联，无法共享
- 破坏了缓存，公共的资源由于带有个人信息，无法缓存。
- 除非用户收藏了特定URL，否则用户退出时信息就会丢失。

Cookie

和上面几种识别用户的方式相比，Cookie是最好的方式。

- Cookie的类型
    - 会话Cookie：临时的，记录用户访问站点的设置和偏好，退出浏览器是会删除。
    - 持久Cookie：存储在硬盘，存在时间更长。也是用来存储用户信息。 它们之间的区别是`过期时间`。
- Cookie是如何工作的 Cookie是一个`name = value` 的键值列表，服务器对一无所知的用户的HTTP响应中，使用`Set-Cookie`或`Set-Cookie2`首部设置。浏览器会记住首部的内容，将来在用户返回同一站点时，会将对应的Cookie传给服务器。
- Cookie罐：客户端的状态 Cookie的思想是让浏览器积累一组服务器特有信息，每次访问服务器都会将对应信息提供给服务器。因为浏览器需要负责存储Cookie信息，所以此系统被称为`客户端侧状态`，规范的名称为`HTTP状态管理机制`。当然，不同的浏览器会以不同的方式进行存储，但它们存储的内容大致相同：
    - domain：域，一般为域名。标识浏览器可以将该Cookie发送出去的站点。
    - allh：标识域中所有主机是否都可以获取Cookie。
    - path：域中Cookie相关的路径前缀。控制了用户在访问哪些路径下资源时，才会发送该Cookie。
    - secure：是否只有在使用ssl连接时才发送该Cookie。
    - expiration：过期时间，从格林尼治标准时间开始的秒数。
    - name和value：Cookie的名字和值。
- Cookie的版本
    - Cookie0，也被称为Netscape cookies。 请求头中的大致格式如下：

    ```
    Cookie: name1=value1[; name2=value2][; name3=value3]...
    ```

    响应头中的大致格式为：

    ```
    Set-Cookie: name=value[; name2=value2][; name3=value3]...
    ```

上面格式中提到的name包含以下选项：

```
强制：NAME
可选：Expires、Domain、Path、Secure
```

- Cookie1 Cookie1版本引入了Set-Cookie2（响应首部）和Cookie2（请求首部）首部，可以和版本0的系统相互操作。 响应头中的大致格式为：

  ```
  Set-Cookie2: name=value[; name2=value2][; name3=value3]...
  ```

  上面格式中提到的name包含以下选项：

  ```
  强制：
  NAME，不能以$开头
  Version，cookie的版本。如`Set-Cookie2: Version="1"`
  可选：
  Comment，说明服务器准备如何使用该Cookie。用户可以通过检查此策略来确定是否允许使用带有该Cookie的会话。必须
  CommentURL，详细描述Cookie策略及目的的文档地址。
  Discard，若提供该属性，表示客户端在退出时放弃该Cookie。
  Domain，使用该Cookie的域名标识。
  Max-Age，生存周期，单位秒。客户端根据使用期计算规则进行计算。计算的值大于该值时，该Cookie应该丢弃。
  Path，可以使用该Cookie的路径。
  Port，说明可以使用Cookie的端口号，如：`Set-Cookie2：name="xx"; Port="80,81,82"`。若只有Port没有值
  Secure，是否只有在使用ssl连接时才发送该Cookie。
  ```

  请求头中的大致格式如下：

  ```
  Cookie: name1=value1[; name2=value2][; name3=value3]...
  ```

  在回传匹配Cookie时，需要将其过滤器一同传输，而且保留的关键字需要以$开头。
- Cookie的版本协商 若服务器能理解Cookie2，就使用Cookie2版本。若客户端从同一个服务器既获得了Set-Cookie首部又获得了Set-Cookie2首部，应该忽略老版本。 若客户端支持两个版本的Cookie，但从服务器获得的是版本0，就应该使用Cookie0发送。同时也应该发送Cookie2：$Version="1"告知服务器可以进行升级。

- Cookie与会话跟踪 服务器在合适时候对浏览器发送Cookie，并将浏览器重定向到另一个URL，浏览器会在再次访问时带回Cookie，这样服务器就可以进行会话跟踪。

结语

今天主要了解了服务器识别客户端的相关手段. 希望大家也能有所收获.

我们在享受技术带来的便利的同时，自己的隐私也在丢失。技术本是无罪的，只是某些人的心坏了。你是否也被上面的技术跟踪了....

注

- 部分图片来源于网络，如有侵权，请告知。
- 如有错误，还请指出。共勉!
- 您的喜欢是最大的赞赏。