攻防世界-web xff_Referer

原创

Quase7 于 2021-09-11 19:22:18 发布 666 收藏 1

分类专栏： X-CTF 文章标签： http web安全

本文链接：https://blog.csdn.net/m0_53533553/article/details/120241873

xff_Referer

扩展

xff

Romote Address

xff简称X-Forwarded-For

X-Forwarded-For 是一个 HTTP 扩展头部，是用来识别通过HTTP代理，或，负载均衡方式，连接到Web服务器的客户端最原始的IP地址，的HTTP请求头字段。

主要是为了**让 Web 服务器获取访问用户的真实 IP 地址**。

那为什么 Web 服务器只有通过 X-Forwarded-For 头才能获取真实的 IP？

Remote Address代表的是当前HTTP请求的远程地址，**即HTTP请求的源地址**。HTTP协议在三次握手时使用的就是这个Remote Address地址，在发送响应报文时也是使用这个Remote Address地址。**其获取到的 IP 是 Web 服务器 TCP 连接的 IP**。因此，如果请求者伪造Remote Address地址，他将无法收到HTTP的响应报文，此时伪造没有任何意义。这也就使得Remote Address默认具有防篡改的功能。

但是很多用户都通过代理来访问服务器的，那么此时获取到的 IP 就是代理服务器的 IP（不是用户的）。

那么看看一个请求可能经过的路径：

客户端=>（正向代理=>透明代理=>服务器反向代理=>）Web服务器

- 正向代理：很多**企业**会在自己的出口网关上设置代理（主要是为了加速和节省流量）。
- 透明代理：可能是**用户自己**设置的代理（比如为了翻墙，这样也绕开了公司的正向代理）。
- 服务器反向代理：是部署在 **Web 服务器**前面的，主要原因是为了负载均衡和安全考虑。

现在假设几种情况：

- 假如客户端**直接**连接 Web 服务器（假设 Web 服务器有公网地址），则获取到的是客户端的**真实 IP**。
- 假设 Web 服务器前部署了**反向代理**（比如 Nginx），则获取到的是**反向代理设备的 IP**（Nginx）。
- 假设客户端通过**正向代理**直接连接 Web 服务器（假设 Web 服务器有公网地址），则获取到的**正向代理设备的 IP** 。

那么由此可见，因为有了各种代理，才会导致Romote Address获取的IP产生了一定的歧义，为了让 Web 服务器获取到真实的客户端 IP。
X-Forwarded-For 出现了

X-Forwarded-For

这个协议头的格式：X-Forwarded-For: client, proxy1, proxy2

client 表示用户的真实 IP，每经过一次代理服务器，代理服务器会在这个头增加用户的 IP。
注意最后一个代理服务器请求 Web 服务器的时候是不会将自己的 IP 附加到 X-Forwarded-For 头上的，最后一个代理服务器的 IP 地址应该通过Romote Address获取。

举个例子：
如果一个 HTTP 请求到达服务器之前，经过了三个代理 Proxy1、Proxy2、Proxy3，IP 分别为 IP1、IP2、IP3，用户真实 IP 为 IP0，那么按照 XFF 标准，服务端最终会收到以下信息：

X-Forwarded-For: IP0, IP1, IP2

Proxy3 直连服务器，它会给 XFF 追加 IP2，表示它是在帮 Proxy2 转发请求。列表中并没有 IP3，**IP3 可以在服务端通过 Remote Address 字段获得**。

通过 X-Forwarded-For 就能获取到用户的真实 IP，是不是万事大吉了？

伪造X-Forwarded-For

通常可以直接通过修改http头中的X-Forwarded-For字段来仿造请求的最终ip！！！

Burp Repeater 是一个手动修改并补发个别 HTTP 请求，并分析他们的响应的工具。

X-Forwarded-For 安全性

那么很多同学会说，通过 X-Forwarded-For 就能获取到用户的真实 IP，是不是万事大吉了，对于 Web 服务器来说，安全有两个纬度，第一个纬度是 REMOTE_ADDR 这个头，这个头不能伪造。第二个纬度就是 X-Forwarded-For，但是这个头是可以伪造的。

那么谁在伪造呢？，我们分别看下：

> 正向代理一般是公司加速使用的，假如没有特殊的目的，不应该传递 X-Forwarded-For 头，因为它的上层连接是内部 IP，不应该暴露出去，当然它也可以透明的传递这个头的值（而这个值用户可以伪造）。
>
> 透明代理，这个可能是用户自己搭建的（比如翻墙），而且在一个用户的请求中，可能有多个透明代理，这时候透明代理就抓瞎了，为了让自己尽量的正确，也会透明的传递这个头的值（而这个值用户可以伪造），当然一些不法企业或者人员，为了一些目的，会改下这个头的值（比如来自世界各地的 IP 地址）。
>
> 反向代理，Web 服务器前的反向代理服务器是不会伪造的（同一个公司的），一般会原样传递这个头的值。

那么对应用程序来说，既然这个值不能完全相信，该怎么办呢？这取决于应用的性质：

假如提供的服务可能就是一些非机密服务，也不需要知道用户的真实 IP，那么建议应用程序或者 Web 服务器对 romote address 做一些限制，比如进行限速等等，也可以放行一些白名单的代理 IP，但是这些白名单 IP 就太难衡量了。

假设你的服务很重要，比如抽奖（一个 IP 只能一次抽奖），这时候你可能想通过 X-Forwarded-For 来获取用户的真实 IP（假如使用 REMOTE_ADDR 则会误杀一片），但是由于 X-Forwarded-For 可能会伪造，所以其实并没有什么好的办法，只能在应用层进行处理了。

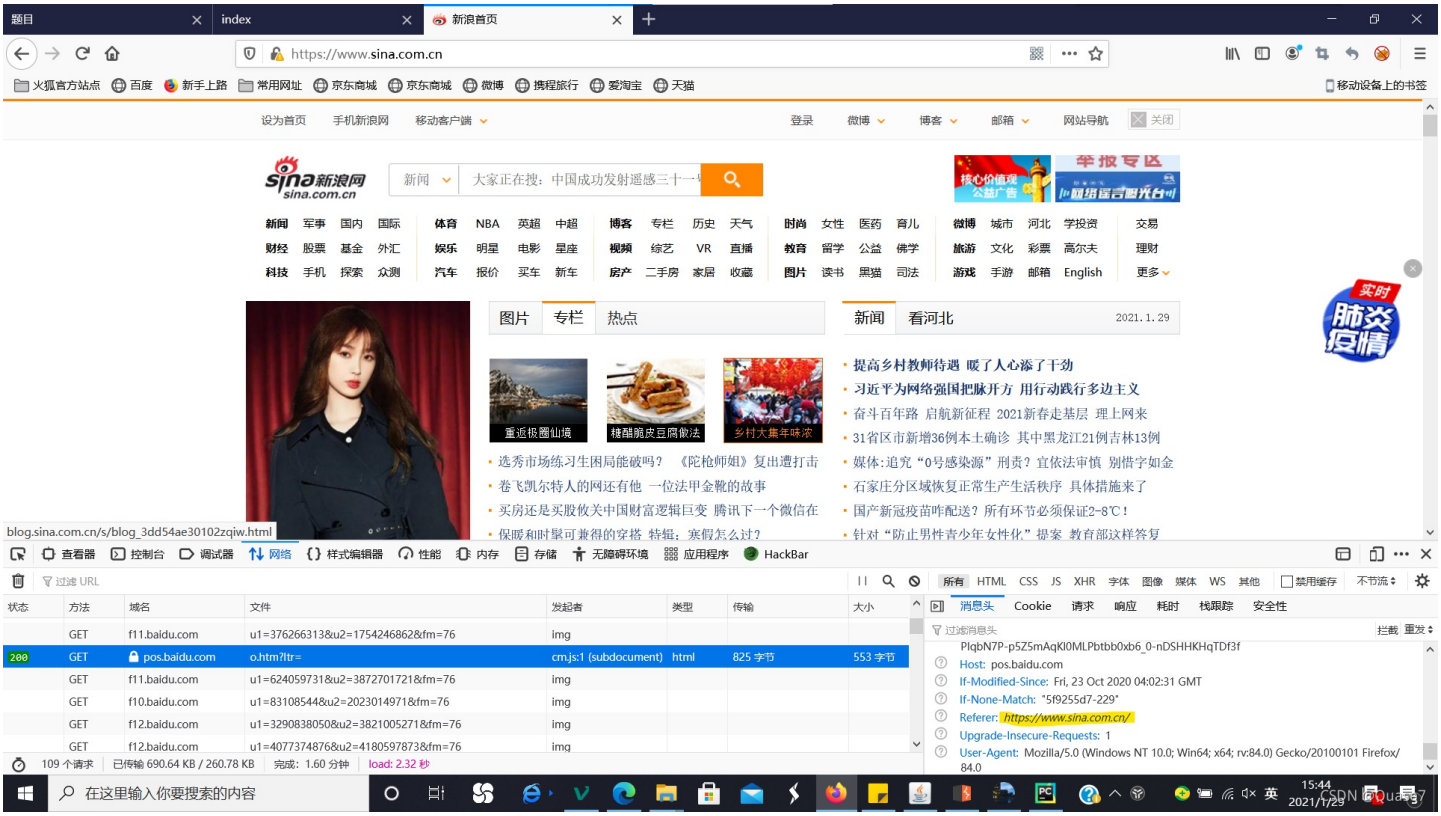
xff

Referer

HTTP Referer是header的一部分，当浏览器向web服务器发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的
Referer，**表示页面的来源**。当浏览器（或者模拟浏览器行为）向web 服务器发送请求的时候，头信息里有包含 Referer 。

比如我在www.google.com 里有一个www.baidu.com 链接，那么点击这个www.baidu.com ，它的header 信息里就有：Referer=http://www.google.com

由此可以看出来吧。它就是表示一个**来源**。看下图的一个请求的 Referer 信息。



referer的作用

1.防盗链。

刚刚前面有提到一个小 Demo 。

我在www.google.com 里有一个www.baidu.com 链接，那么点击这个www.baidu.com， 它的header信息里就有：Referer=http://www.google.com

那么可以利用这个来防止盗链了，比如我只允许我自己的网站访问我自己的图片服务器，那我的域名是www.google.com， 那么图片服务器每次取到Referer来判断一下是不是我自己的域名www.google.com， 如果是就继续访问，不是就拦截。

将这个http请求发给服务器后，如果服务器要求必须是某个地址或者某几个地址才能访问，而你发送的referer不符合他的要求，就会拦截或者跳转到他要求的地址，然后再通过这个地址进行访问。

2.统计网站流量

统计文章有多少次是来自谷歌搜索结果，多少次来自百度搜索结果等。
比如从我主页上链接到一个朋友那里，他的服务器就能够从HTTP Referer中统计出每天有多少用户通过点击我主页上的链接访问他的网站。

3.防止恶意请求。

比如静态请求是*.html结尾的，动态请求是*.shtml，那么由此可以这么用，所有的动态请求，其Referer 必须 为我自己的网站。

Referer同样是不可信的！！！

空Referer

空Referer是怎么回事？什么情况下会出现Referer?
首先，我们对空 Referer 的定义为， Referer 头部的内容为空，或者，一个 HTTP 请求中根本不包含 Referer 头部。

那么什么时候 HTTP 请求会不包含 Referer 字段呢？根据Referer的定义，它的作用是指示一个请求是从哪里链接过来，那么当一个请求并不是由链接触发产生的，那么自然也就不需要指定这个请求的链接来源。

比如，直接在浏览器的地址栏中输入一个资源的URL地址，那么这种请求是不会包含 Referer 字段的，因为这是一个“凭空产生”的 HTTP 请求，并不是从一个地方链接过去的。

那么在防盗链设置中，允许空Referer和不允许空Referer有什么区别？
允许 Referer 为空，意味着你允许比如浏览器直接访问，就是空。

xff和referer区别

x-forwarded-for 和 referer的区别: x-forwarded-for 用来证明ip的像是“127.0.0.1”这种，而referer是用来证明“域名”的

XFF构造来源IP 例如127.0.0.1

Refer构造来源浏览器 例如https://www.google.com

origin

Host
描述请求将被发送的目的地，包括，且仅仅包括**域名和端口号**。
在任何类型请求中，request都会包含此header信息。

Origin
用来说明请求从哪里发起的，包括，且仅仅包括**协议和域名。**
这个参数一般只存在于CORS跨域请求中，可以看到response有对应的header：Access-Control-Allow-Origin。

Referer
告知服务器请求的原始资源的URI，其用于所有类型的请求，并且包括：**协议+域名+查询参数**（注意，不包含锚点信息）。
因为原始的URI中的查询参数可能包含ID或密码等敏感信息，如果写入referer，则可能导致信息泄露。

origin主要是用来说明最初请求是从哪里发起的；

origin只用于Post请求，而Referer则用于所有类型的请求；

origin的方式比Referer更安全点吧。

攻略

burpsuite抓取页面，发送至repeater，在消息头中添加x-forwarded-for:123.123.123.123

Burp Repeater 是一个手动修改并补发个别 HTTP 请求，并分析他们的响应的工具。

response中提示.innerHTML="必须来自https://www.google.com";

在已添加x-forwarded-for:的基础上，继续添加Referer:https://www.google.com，发送请求，response中得到flag。

并分析他们的响应的工具。

response中提示.innerHTML="必须来自https://www.google.com";

在已添加x-forwarded-for:的基础上，继续添加Referer:https://www.google.com，发送请求，response中得到flag。