链滴

HTTP 请求头中的 X-Forwarded-For

作者：jaz

原文链接：https://ld246.com/article/1454320869175

来源网站：链滴

###背景
通过名字就知道，X-Forwarded-For 是一个扩展头。HTTP/1.1（RFC 2616）协议并没有对它定义，它最开始是由 Squid 这个缓存代理软件引入，用来表示 HTTP 请求端真实 IP，现在已经成为实上的标准，被各大 HTTP 代理、负载均衡等转发服务广泛使用，并被写入 RFC 7239（Forwarded HTTP Extension）标准之中。
X-Forwarded-For 请求头格式非常简单，就这样：
<pre><code class="lang-shell">X-Forwarded-For: client, proxy1, proxy2
</code></pre>
可以看到，XFF 的内容由「英文逗号 + 空格」隔开的多个部分组成，最开始的是离服务端最远设备 IP，然后是每一级代理设备的 IP。
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 IP、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：
<pre><code class="lang-shell">X-Forwarded-For: IP0, IP1, IP2
</code></pre>
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 IP，IP3 可以通过服务端的 Remote Address 字段获得。我们知道 HTTP 连接基于 TCP 连接，HTTP 议中没有 IP 的概念，Remote Address 来自 TCP 连接，表示与服务端建立 TCP 连接的设备 IP，在个例子里就是 IP3。
Remote Address 无法伪造，因为建立 TCP 连接需要三次握手，如果伪造了源 IP，无法建立 TC 连接，更不会有后面的 HTTP 请求。不同语言获取 Remote Address 的方式不一样，例如 php 是 <code>$_SERVER["REMOTE_ADDR"]</code>，Node 是 <code>req.connection.remoteAddress</code>，但原理都一样。
###问题
有了上面的背景知识，开始说问题。我用 Node 写了一个最简单的 Web Server 用于测试。HTTP 协议跟语言无关，这里用 Node 只是为了方便演示，换成任何其他语言都可以得到相同结论。另外本用 Nginx 也是一样的道理，如果有兴趣，换成 Apache 或其他 Web Server 也一样。
下面这段代码会监听 <code>9009</code> 端口，并在收到 HTTP 请求后，输一些信息：
<pre>JS<code class="lang-js">var http = require('http');

http.createServer(function (req, res) {
res.writeHead(200, {'Content-Type': 'text/plain'});
res.write('remoteAddress: ' + req.connection.remoteAddress + '\n');
res.write('x-forwarded-for: ' + req.headers['x-forwarded-for'] + '\n');
res.write('x-real-ip: ' + req.headers['x-real-ip'] + '\n');
res.end();
}).listen(9009, '0.0.0.0');
</code></pre>

这段代码除了前面介绍过的 Remote Address 和 <code>X-Forwarded-For</code>，有一个 <code>X-Real-Ip</code>，这又是一个自定义头。<code>X-Real-Ip</code> 通常被 HTTP 代理用来表示与它产生 TCP 连接的设备 IP，这个设备可能是其他代理，也可能是真

的请求端。需要注意的是，<code>X-Real-Ip</code> 目前并不属于任何标准，代理和 Web 应用之间可以约定用任何自定义头来传递这个信息。
现在可以用域名 + 端口号直接访问这个 Node 服务，再配一个 Nginx 反向代理：
<pre>NGINX<code class="lang-nginx">location / {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;

proxy_pass http://127.0.0.1:9009/;
proxy_redirect off;

}
</code></pre>

我的 Nginx 监听 <code>80</code> 端口，所以不带端口就可以访问 Nginx 转过的服务。
测试直接访问 Node 服务：
<pre>SHELL<code class="lang-shell">curl http://t1.imquq.com:9009/

remoteAddress: 114.248.23.236
x-forwarded-for: undefined
x-real-ip: undefined
</code></pre>

由于我的电脑直接连接了 Node 服务，Remote Address 就是我的 IP。同时我并未指定额外的定义头，所以后两个字段都是 undefined。
再来访问 Nginx 转发过的服务：
<pre>SHELL<code class="lang-shell">curl http://t1.imquq.com/

remoteAddress: 127.0.0.1
x-forwarded-for: 114.248.28.236
x-real-ip: 114.248.238.236
</code></pre>

这一次，我的电脑是通过 Nginx 访问 Node 服务，得到的 Remote Address 实际上是 Nginx 本地 IP。而前面 Nginx 配置中的这两行起作用了，为请求额外增加了两个自定义头：

<pre><code class="lang-nginx">proxy_set_header X-Real-I
$remote_addr;
proxy_set_header X-Forwarded-For <span class="hljs-variab
e">$proxy_add_x_forwarded_for;
</code></pre>
实际上，在生产环境中部署 Web 应用，一般都采用上面第二种方式，好处多多，具体是哪些不
本文重点不写了。这就引入一个隐患：很多 Web 应用为了获取用户真正的 IP，从 HTTP 请求头中获
IP。
HTTP 请求头可以随意构造，我们通过 curl 的 <code>-H</code> 参数构造&nb
p;<code>X-Forwarded-Fox</code> 和 <code>X-Real-Ip</code>，再来测试一
。
直接访问 Node 服务：
<pre>SHELL<code class="lang-shell">curl http://t1.imquq
.com:9009/ -H 'X-Forwarded-
or: 1.1.1.1' -H 'X-Real-Ip: 2.2.2.2'

remoteAddress: 114.248.23
.236

x-forwarded-for: 1.1.1.1</s
an>

x-real-ip: 2.2.2.2

</code></pre>

对于 Web 应用来说，<code>X-Forwarded-Fox</code> 和 <code>X-Real-Ip
/code> 就是两个普通的请求头，自然就不做任何处理原样输出了。这说明，对于直连部署方
，除了从 TCP 连接中得到的 Remote Address 之外，请求头中携带的 IP 信息都不能信。
访问 Nginx 转发过的服务：
<pre>SHELL<code class="lang-shell">curl http://t1.imquq
.com/ -H 'X-Forwarded-For: 1.1.1.1' -H <span class="hljs-s
ring">'X-Real-Ip: 2.2.2.2'

remoteAddress: 127.0.0.1</
pan>

x-forwarded-for: 1.1.1.1</s
an>, 114.248.238.236</spa
>

x-real-ip: 114.248.238.236<
span>

</code></pre>

这一次，Nginx 会在 <code>X-Forwarded-For</code> 后追加我的 IP；并用
的 IP 覆盖 <code>X-Real-Ip</code> 请求头。这说明，有了 Nginx 的加工，<code
X-Forwarded-For</code> 最后一节以及 <code>X-Real-Ip</code> 整个
容无法构造，可以用于获取用户 IP。
用户 IP 往往被使用在跟 Web 安全有关的场景上，例如检查用户登录地区，基于 IP 做访问频率
制等等。这种场景下，确保 IP 无法构造更重要。经过前面的测试和分析，对于直接面向用户部署的
eb 应用，必须使用从 TCP 连接中得到的 Remote Address；对于部署了 Nginx 这样反向代理的 Web
应用，在正确配置了 Set Header 行为后，可以使用 Nginx 传过来的 <code>X-Real-Ip</co
e> 或 <code>X-Forwarded-Ip</code> 最后一节（实际上它们一定等价）。
/p>
那么，Web 应用自身如何判断请求是直接过来，还是由可控的代理转发来的呢？在代理转发时

加额外的请求头是一个办法，但是不怎么保险，因为请求头太容易构造了。如果一定要这么用，这个定义头要够长够罕见，还要保管好不能泄露出去。
判断 Remote Address 是不是本地 IP 也是一种办法，不过也不完善，因为在 Nginx 所处服务上访问，无论直连还是走 Nginx 代理，Remote Address 都是 127.0.0.1。这个问题还好通常可以忽，更麻烦的是，反向代理服务器和实际的 Web 应用不一定部署在同一台服务器上。所以更合理的做是收集所有代理服务器 IP 列表，Web 应用拿到 Remote Address 后逐一比对来判断是以何种方式问。
通常，为了简化逻辑，生产环境会封掉通过带端口直接访问 Web 应用的形式，只允许通过 Nginx 来访问。那是不是这样就没问题了呢？也不见得。
首先，如果用户真的是通过代理访问 Nginx，<code>X-Forwarded-For</code> 最后节以及 <code>X-Real-Ip</code> 得到的是代理的 IP，安全相关的场景只能用这个但有些场景如根据 IP 显示所在地天气，就需要尽可能获得用户真实 IP，这时候 <code>X-Forwarded-For</code> 中第一个 IP 就可以排上用场了。这时候需要注意一个问题，还是拿之的例子做测试：
<pre>SHELL<code class="lang-shell">curl http://t1.imquq.com/ -H 'X-Forwarded-For: unknown, <>"1.1.1.1'
remoteAddress: 127.0.0.1
x-forwarded-for: unknown, <>"1.1.1.1, 114.248.238.236
x-real-ip: 114.248.238.236
</code></pre>
<code>X-Forwarded-For</code> 最后一节是 Nginx 追加上去的，但之前部分都来自 Nginx 收到的请求头，这部分用户输入内容完全不可信。使用时需要格外小心，符合 IP 格式才能使，不然容易引发 SQL 注入或 XSS 等安全漏洞。
<h3>结论</h3>

直接对外提供服务的 Web 应用，在进行与安全有关的操作时，只能通过 Remote Address 获取P，不能相信任何请求头；
使用 Nginx 等 Web Server 进行反向代理的 Web 应用，在配置正确的前提下，要用 <code>X-Forwarded-For</code> 最后一节 或 <code>X-Real-Ip</code> 来获取IP（因为 Remote Address 得到的是 Nginx 所在服务器的内网 IP）；同时还应该禁止 Web 应用直对外提供服务；
在与安全无关的场景，例如通过 IP 显示所在地天气，可以从 <code>X-Forwarded-For</code> 靠前的位置获取 IP，但是需要校验 IP 格式合法性；

PS：网上有些文章建议这样配置 Nginx，其实并不合理：
<pre><code class="lang-nginx">proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $remote_addr;
</code></pre>
这样配置之后，安全性确实提高了，但是也导致请求到达 Nginx 之前的所有代理信息都被抹掉无法为真正使用代理的用户提供更好的服务。还是应该弄明白这中间的原理，具体场景具体分析。</>