基于TLS协议报文检测NAT网络

[1． 基于TLS协议报文检测NAT网络的目标 2](#_Toc31295)

[2． 基于TLS协议报文进行NAT网络检测的两个概念 3](#_Toc19820)

[2.1 TLS客户端标识 3](#_Toc19400)

[2.1.1 JA3指纹 3](#_Toc11579)

[2.1.2服务端域名，服务端IP 3](#_Toc26304)

[2.1.3 TLS客户端标识的方法 4](#_Toc1606)

[2.2 Session Ticket序列 4](#_Toc21088)

[2.2.1 Session Ticket的生存期和复用 4](#_Toc321)

[2.2.2 Session Ticket序列 4](#_Toc16769)

[3． NAT网络检测方法 5](#_Toc18119)

[3.1 构建TLS客户端对应的Session Ticket序列进行NAT网络检测 5](#_Toc7892)

[3.2 根据Session Ticket序列判定存在NAT网络的标准 5](#_Toc28993)

[4． 实验验证 8](#_Toc29616)

[4.1 实验1 8](#_Toc8947)

[4.2 实验2 8](#_Toc18024)

[4.2 实验结论 9](#_Toc2430)

[5． 总结 10](#_Toc13390)

# 基于TLS协议报文检测NAT网络的目标

通过分析来自同一个客户端IP的流量中TLS协议的客户端发出握手请求的Client Hello报文，检测这个客户端IP的背后是否存在NAT网络。

# 基于TLS协议报文进行NAT网络检测的两个概念

基于TLS协议报文进行NAT网络检测中，涉及到两个重要的概念：TLS客户端标识和Session Ticket序列。

## **2.1 TLS客户端标识**

TLS客户端是指使用TLS协议与服务端通信的运行在用户端的程序。随着TLS协议的广泛应用，大部分运行在用户端的程序都采用TLS协议与服务端通信，PC或手机、平板等个人设备上随时都运行几个或几十个TLS客户端。

不同的应用程序发起的TLS连接是不同的（2.1.1小节会有说明），同一种应用程序的TLS连接是相似的（比如同一种应用程序会访问相同的域名），也就意味着不同应用拥有不同的TLS客户端。找到一种从Client Hello报文标识不同TLS客户端的方法，这样的话就可以区分不同的TLS客户端发出的Client Hello报文了。

### 2.1.1 JA3指纹

不同的TLS客户端的构建方式是不同，因为程序员可以根据OpenSSL等开源库自由的选择使用何种加密套件、启用那些扩展，只要符合TLS协议的标准就行了。这些信息可以从TLS客户端发出来的Client Hello报文中泄露出来。JA3指纹是Client Hello报文中加密套件、启用那些扩展的MD5 HASH值。因此不同的TLS客户端的JA3指纹是不同的，比如常见的几个浏览器的JA3指纹就是不同的：

|  |  |  |
| --- | --- | --- |
| 浏览器名称 | 浏览器版本 | JA3指纹 |
| QQ浏览器 | Windows版10.6.1 | ebf5e0e525258d7a8dcb54aa1564ecbd |
| Chrome | Windows版85.0.4183.102 | b32309a26951912be7dba376398abc3b |
| 360极速浏览器 | Windows版12.0.1458.0 | 795f6df9605d3b68eae714e0a8ba4edf |

因此，JA3指纹可以用于标识不同的TLS客户端。尽管不同的TLS客户端JA3指纹会有差异，但是因为JA3指纹的差异是TLS客户端选择不同的加密套件、启用不同的扩展带来的，而TLS协议中供选择的加密套件、扩展不过几十种，这就导致不同的应用程序的TLS客户端很可能拥有同样的JA3指纹。因此单单只用JA3指纹无法标识出一个应用程序。

### **2.1.2服务端域名，服务端IP**

应用程序若访问的服务端域名或者是服务端IP往往是固定的，不同的TLS客户端所会访问的服务端域名或者是服务端IP是不同的。比如支付宝应用所会访问的服务端域名与微信应用所会访问的服务端域名就会不一样。

这种设定的特例是浏览器，浏览器可以访问互联网中任何存在的域名，对浏览器来说，可以认为一个浏览器中存在很多种应用程序（浏览器访问不同的域名视作这个域名“专用”的应用程序），每一种应用程序对应一种TLS客户端。

### **2.1.3 TLS客户端标识的方法**

根据之前的分析，可以通过<ja3指纹，服务端域名，服务端IP>这个3元组来标识一个TLS客户端，进而区分不同的TLS客户端发出的Client Hello报文。

## **2.2** Session Ticket**序列**

Session Ticket是一种TLS会话复用机制，在TLS的握手阶段，TLS的客户端可以在Client Hello报文中附带Session Ticket, Session Ticket包含上一次连接的会话密钥等信息(被服务端的专用密钥STEK加密过)，由此尝试恢复上一次的TLS连接，减少TLS连接若需要的时间。

### **2.2.1 Session Ticket的生存期和复用**

TLS客户端可以在Client Hello报文中附带Session Ticket的一个前提是这个重新发起TLS连接的时间减去TLS客户端收到这个Session Ticket的时间没有超过这个Session Ticket的生存期。

根据TLS协议，Session Ticket的生存期是由TLS客户端和服务端共同确定的，因此很难通过被动流量监听的方式确定Session Ticket的生存期。实际上，因为Session Ticket包含了TLS会话中的会话密钥，所以Session Ticket可以视为是随机生成的，因此在流量中的Client Hello报文中只要出现了相同的Session Ticket就意味着发生了复用。反过来，可同一个Session Ticket在流量中第一次出现和最后一次的时间差出现推测TLS客户端的Session Ticket的最小生存期（假定同一TLS客户端的Session Ticket生存期不会发生变化）。

### **2.2.2 Session Ticket序列**

如果一个应用程序在Session Ticket生存期内多次使用同一TLS客户端标识发起TLS连接，也就是发出Client Hello报文，那么这些Client Hello报文内都会带有Session Ticket。这样这个应用程序的网络行为就可以构成一个Session Ticket的序列，并且这个序列因为Session Ticket复用机制，会有相同的值出现。

# NAT网络检测方法

## **3.1 构建TLS客户端对应的Session Ticket序列进行NAT网络检测**

通过被动监听的方式，分析来自同一源IP的流量。先从流量中的Client Hello报文中提取TLS客户端标识，与此把不同的同时构建TLS客户端对应的Session Ticket序列，分析每个TLS客户端对应的Session Ticket序列，来判断是否存在NAT网络。

一般情况下，一个用户同时只会使用同一种应用程序的一个实例，比如用户只会同时使用一个微信聊天，比如用户不会使用不同的浏览器访问同一种应用。也就是说，一般情况下，用户同时只会使用同一种TLS客户端的一个实例。

如果用户同时只会使用同一种TLS客户端的一个实例，那么**同一个TLS客户端的Session Ticket序列中Ticket值的出现顺序应该是有序的（在实验4.1中验证了这一点）**，就是说Ticket A和Ticket B不会交错出现。

## **3.2 根据Session Ticket序列判定存在NAT网络的标准**

如果同一个TLS客户端的Seesion Ticket值的出现顺序是无序的，不同Session Ticket之间交错出现，那么可能存在以下几种可能：

1. 存在NAT网络
2. 用户同时使用了同一种TLS客户端的两个实例（用户同时使用两个微信）
3. 应用程序本身的构建存在问题（比如一个安卓应用的页面上使用相同的TLS客户端发起了连接，但是这些TLS客户端之间并没有共享Ticket，如下表所示，为了便于观察Ticket值做了处理）

|  |  |  |  |
| --- | --- | --- | --- |
| TLS客户端标识 | 客户端IP | 报文出现时间 | Ticket值 |
| catdot.dianping.com | 192.168.11.188 | 15216.4316 | A |
| 203.76.217.13 | 192.168.11.188 | 15216.4316 | A |
| 9b02ebd3a43b62d825e1ac605b621dc8 | 192.168.11.188 | 15217.7051 | B |
|  | 192.168.11.188 | 15219.1714 | A |
|  | 192.168.11.188 | 15219.2063 | A |
|  | 192.168.11.188 | 15219.6401 | C |
|  | 192.168.11.188 | 15219.6407 | B |
|  | 192.168.11.188 | 15274.6121 | D |
|  | 192.168.11.188 | 15306.5473 | D |
|  | 192.168.11.188 | 15336.0636 | D |

1. 用户交错的使用了不同的应用程序，而这些应用程序使用了相同的TLS客户端发起连接（比如同一厂家的应用程序，淘宝APP和天猫APP）

因此，不能是检测到了同一种TLS客户端的Session Ticket序列出现了无序现象就判定存在NAT网络，要排除掉一些情况。考虑这样一种情况，用户A和用户B在一个NAT网络内部同时使用同一种应用程序，此时同一TLS客户端的不同Ticket出现的交错程度就会很高，如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| TLS客户端标识 | 客户端IP | 报文出现时间 | Ticket值 |
| zhidao.baidu.com | 192.168.11.191 | 85.3489 | A |
| 61.135.185.172 | 192.168.11.207 | 93.8075 | B |
| b32309a26951912be7dba376398abc3b | 192.168.11.191 | 205.3494 | A |
|  | 192.168.11.207 | 213.802 | B |
|  | 192.168.11.191 | 325.3612 | A |
|  | 192.168.11.207 | 333.8024 | B |
|  | 192.168.11.191 | 445.3684 | A |
|  | 192.168.11.191 | 565.3851 | A |
|  | 192.168.11.207 | 573.8061 | C |
|  | 192.168.11.191 | 685.3911 | A |
|  | 192.168.11.207 | 693.8156 | C |
|  | 192.168.11.191 | 805.3997 | A |
|  | 192.168.11.207 | 813.7988 | C |
|  | 192.168.11.191 | 925.6569 | A |
|  | 192.168.11.191 | 928.4066 | A |
|  | 192.168.11.207 | 933.7942 | C |
|  | 192.168.11.191 | 1045.4196 | A |
|  | 192.168.11.207 | 1053.8069 | C |
|  | 192.168.11.191 | 1165.4218 | A |
|  | 192.168.11.207 | 1173.7986 | C |
|  | 192.168.11.191 | 1286.4378 | A |
|  | 192.168.11.207 | 1293.8128 | C |
|  | 192.168.11.191 | 1405.438 | A |
|  | 192.168.11.207 | 1413.808 | C |
|  | 192.168.11.191 | 1525.4524 | A |
|  | 192.168.11.207 | 1533.8018 | C |
|  | 192.168.11.191 | 1645.4603 | A |
|  | 192.168.11.207 | 1655.0475 | C |
|  | 192.168.11.207 | 1656.8206 | C |
|  | 192.168.11.191 | 1765.474 | A |
|  | 192.168.11.207 | 1773.804 | C |
|  | 192.168.11.191 | 1885.475 | A |
|  | 192.168.11.207 | 1894.8306 | C |
|  | 192.168.11.191 | 2006.4919 | A |
|  | 192.168.11.207 | 2013.8077 | C |
|  | 192.168.11.191 | 2125.497 | A |
|  | 192.168.11.207 | 2133.7905 | C |
|  | 192.168.11.191 | 2245.5079 | A |
|  | 192.168.11.207 | 2253.8027 | C |
|  | 192.168.11.191 | 2365.516 | A |
|  | 192.168.11.207 | 2373.8089 | C |
|  | 192.168.11.191 | 2485.5276 | A |
|  | 192.168.11.207 | 2493.9008 | C |
|  | 192.168.11.191 | 2605.5296 | A |

可见在这种情况下，来自两个不同用户的Session Ticket几乎交替出现的。实验结果表明，**只有在存在NAT网络的情况下才会出现同一TLS客户端的Session Ticket序列几乎交替出现的情况（在实验4.2中验证了这一点）。**

为了很好的描述这种情况，定义表达同一个Ticket在一个Session Ticket序列中离散情况的离散指数：

假设Ticket A在某个Session Ticket序列中出现的位置分别是：，假设Ticket A的第次出现和第次出现之间的还出现了个不同的Ticket，那么Ticket A的离散指数是：



实验结果表明（实验4.2），在来自同一源IP的流量中，构建TLS客户端标识对应的Session Ticket序列，在所有的序列中如果存在某个Ticket的离散系数大于20的情况，可以认为来自此源IP的流量中存在NAT网络。

# 实验验证

实验数据集是同一NAT网络内64个源IP的3个小时的网络流量。这些IP背后大多数是单一的Windows设备，也有少部分IP背后是路由器。这些路由器的背后基本只存在两台不同设备，比如一台路由器设备后有一台Windows/Apple电脑和一台Android/Apple手机。并且根据观察这两种不同设备之间的应用程序之间基本不存在着交集，也就意味着这两种不同设备之间的TLS客户端基本不相同，在我们的实验中，他们完全可以视作是单一设备。实验数据集一共有77051条Client Hello报文，去掉TCP重传的报文、不带SNI的报文和不带Session Ticket的报文，实际可用的Client Hello报文有个。

## **4.1 实验1**

验证3.1小节中的假设“一般情况下，同一个TLS客户端的Session Ticket序列中Ticket值的出现顺序应该是有序的”是否成立。

对数据集中64个源IP的流量独立分析，构建TLS客户端标识和每个TLS客户端标识对应的Session Ticket序列。一共得到了3164个不同的TLS客户端标识，这些TLS客户端标识对应的Session Ticket序列长度的最大值为431，Session Ticket序列长度的最小值为1。Session Ticket序列平均长度为5.36，Session Ticket序列长度大于3个的有1128个。Session Ticket序列长度的分布为：

在这1128个Session Ticket序列长度大于3的TLS客户端标识中，仅有32个TLS客户端标识对应的Session Ticket序列发生了交错现象，占Session Ticket序列长度大于3的TLS客户端标识的2.81%，占TLS客户端标识的

## **4.2 实验2**

验证3.2小节中的假设“只有在存在NAT网络的情况下才会出现同一TLS客户端的Session Ticket序列几乎交替出现的情况”是否成立。

先按照实验1的方式实验。一共得到了6796个不同的Session Ticket，其中复用次数最多的Session Ticket被复用了404次，复用次数最少的Session Ticket被复用了1次，每个Session Ticket平均被复用了2.459次，被复用次数两次及以上的有2123个Session Ticket。在这6796条Session Ticket中，出现了交错出现的Session Ticket有52条，占全部Session Ticket的0.76%。这些发生了交错的Ticket中，最大离散指数是3，前10大离散指数是：3，2，2，2，2，2，2，2，2，2，离散指数平均值是1.19.

然后，再把数据集中64个源IP的流量当成是从一个源IP中发出来的流量进行分析，构建TLS客户端标识和每个TLS客户端标识对应的Session Ticket序列。一共得到了2487个不同的TLS客户端标识，其中85个TLS客户端标识对应的Session Ticket序列发生了交错现象。一共得到了6790个不同的Session Ticket，出现了交错出现的Session Ticket有303条，占全部Session Ticket的4.46%。这些发生了交错的Ticket中，最大离散指数是20.52857143，前40大离散指数是：20.52857143，20，17.04571429，17，12.1974359，9.333333333，9，8，7.921428571，7.488970588，7，7，7，7，7，7，7，7，7，6.848484848，6.666666667，6.666666667，6.666666667，6.666666667，6.666666667，6.666666667，6.435185185，6，6，6，6，6，6，6，6，6，6，6，6，5.855952381，5.833333333，5.666666667，5.666666667，5.666666667，5.666666667，5.666666667，5.666666667，5.666666667，5.4，5.384615385，5，5，5，5，5。

## **4.2 实验结论**

根据实验1、实验2可以证实3.1和3.2中假设全部成立：

一般情况下，同一个TLS客户端的Session Ticket序列中Ticket值的出现顺序应该是有序的，实验观测得到有序的概率在98.99%。

只有在存在NAT网络的情况下才会出现同一TLS客户端的Session Ticket序列几乎交替出现的情况。实验观测表明，在实验数据集上，不存在NAT网络时，Ticket的最大离散指数不超3，存在NAT网络时Ticket的最大交错指数为20.52857143，且有40个Ticket的离散指数大于5.

因此，在来自同一源IP的流量中，构建TLS客户端标识对应的Session Ticket序列，在所有的序列中如果存在某个Ticket的离散系数大于20，可以认为来自此源IP的流量中存在NAT网络。

# 总结

本此实验的核心思路是通过被动监听的方式，分析来自同一源IP的流量。先从流量中的Client Hello报文中提取TLS客户端标识，与此同时根据TLS客户端标识把流量中的Client Hello报文分开，构建TLS客户端标识的Session Ticket序列。根据Session Ticket序列中Session Ticket应该是基本有序，不会交错这一特点入手（因为TLS客户端标识可对应一种应用程序，用户一般只会使用一种应用程序的一个实例），提出了Ticket的离散指数这一指标。

经过实验后得出结论：如果存在某个Ticket的离散系数大于20，可以认为来自此源IP的流量中存在NAT网络。