频繁串寻找使用说明

频繁串寻找算法是基于后缀数据来完成的，该问题被当做“从N个字符串中寻找所有出现次数不小于K的字符子串”来对待。

频繁串算法被封装在SuffixSearch.h的SuffixSearch类里面，该类所有函数都实现为内联函数。

其中SuffixSearch类的构造函数原型为：

SuffixSearch(float \_threshold); 接受一个阈值参数，该参数是一个0-1的浮点数，表示搜索给定的N个数据包里面出现次数大于N \* \_threshold的子串。这里的子串，即可以是ASIIC可打印串，也可以是二进制串。

SuffixSearch类的核心成员函数有两个：

SuffixSearch::feed (unsigned char \* payload\_str, int length)；

int SuffixSearch::calc(vector<string>& patterns,vector<int>& occurence);

其中 调用feed函数用于把待搜索的数据包的载荷的起始位置和载荷的长度丢到类中。每次调用只能注入一个pcap的载荷，这个约束主要是因为feed每调用一次就会认为灌入的数据包个数会增加1。根据分析的目标不同，载荷既可以是应用层的payload也可以是传输层、网络层的payload。

在灌入载荷的时候，每次分析应该把具有相同属性的数据包载荷feed进入。这是因为只有具有相同属性的数据包之间，才更有可能存在频繁串。例如对于TLS握手数据包而言，应该把所有Client Hello放在一起单独分析，而不要把ClientHello和ServerHello混在一起。这主要是因为我们是通过出现次数超过总次数的某个百分比来定义“频繁”的，在ClientHello中以50%的概率出现的子串，在ClientHello和ServerHello混合包中出现的概率就可能降低为25%了。

calc()函数用于在灌入所有所需的数据包后，计算这些数据包里面出现次数超过一定阈值的子串。函数接收两个参数： vector<string> & patterns和vector<int> & occurrence。两个参数都是引用类型。Calc会把计算结果保存到这两个参数中，其中patterns[i]保存寻找到的第i个频繁串，而occurrence[i]则是表示第i串出现的次数。Calc()的返回值为int,表示本次计算找到了多少个串。

**注意！ 所有串是以string的形式保存的，有些串可能是Ansiic不可打印的，因此不能通过标注输入输出去试图把整个串打印。如果要打印整个串，应该遍历这个串的所有元素，按元素输出。**

**main.cpp是一个例程，用于显示如何通过SuffixSearch来寻找mysql连接中的频繁串。**