一种工控协议识别中的特征字符串挖掘算法

海 洋[1],徐 魁[1],李晓辉[2],曾 涛[3],陶 军[3]
(1. 宝鸡市公安局通信处,陕西 宝鸡 721014;
2. 宝鸡创天清航科技发展有限责任公司,陕西 宝鸡 721000;
3. 东南大学 网络空间安全学院,江苏 南京 210096)

摘 要:对工控协议的识别,是对工控协议开展研究的第一步。而在通信过程中频繁出现的字符串,是对工控协议识别中的重要特征。针对工控协议识别中特征字符串的提取问题,提出了一种自顶向下的频繁字符串挖掘算法,可以直接得到没有冗余的频繁字符串集。同时,对于自顶向下方法中原始数据过于庞大、算法迭代次数较多等问题,借鉴了 N-gram 模型,提出了一种数据划分策略,解决了自顶向下处理时数据过大的问题。此外,在挖掘频繁字符串的过程中,采取了删除重叠项与字符串分裂相结合的方法。实验结果表明,该算法针对多种协议均能识别出其中的特征字符串;同时,利用识别出的字符串作为特征,在协议识别工作中也能取得良好的效果。可以得出结论,该算法能够较好地提取出工控协议中的特征字符串。

关键词:频繁字符串;自顶向下;数据划分;特征提取;数据处理
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2024)01-0200-06
doi:10.3969/j.issn.1673-629X.2024.01.029

A Feature String Mining Algorithm in Industrial Control Protocols Recognition

HAI Yang[1],XU Kui[1],LI Xiao-hui[2],ZENG Tao[3],TAO Jun[3]
(1. Communications Department of Baoji Public Security Bureau,Baoji 721014,China;
2. Baoji Chuangtianqinghang Technology Development Co.,Ltd.,Baoji 721000,China;
3. School of Cyber Science and Engineering,Southeast University,Nanjing 210096,China)

Abstract:The identification of industrial control protocols is the first step in research on industrial control protocols. In the communication process,frequently occurring strings are important features for identifying industrial control protocols. We propose a top-down frequent string mining algorithm that can directly obtain a non-redundant set of frequent strings for feature extraction in industrial control protocols identification. Additionally,to address the issue of large original data and numerous algorithm iterations in the top-down method,we borrow from the N-gram model and propose a data partitioning strategy to solve the problem of processing large data in the top-down approach. Furthermore,to mine frequent strings,we adopt a combination of deleting overlapping items and string splitting. Experimental results show that the proposed algorithm can identify feature strings in multiple protocols and achieve good results in protocol identification by using identified strings as features. It can be concluded that the proposed algorithm can effectively extract feature strings from industrial control protocols.

Key words:frequent strings;top-down;data segmentation;features extraction;data processing

0 引 言

随着 2010 年“震网病毒”[1]的出现,工业控制系统(Industrial Control Systems,ICS)[2]的安全性逐渐受到了全球范围的关注。工业控制系统包括多种类型的控制系统,这些控制系统对关键基础设施的运行至关重要。工业控制系统由许多的子系统以及相关的组件构成,各组件之间传递一些控制信息或者监控数据,这些信息或者数据的传递均是以工控协议的形式传输,各组件对于工控协议的解析与处理影响着工控系统的安全。

收稿日期:2023-03-21 **修回日期**:2023-07-25
基金项目:中国高校产学研创新基金-阿里云高校数字化创新专项(2021ALA03006)
作者简介:海 洋(1984-),男,高级工程师,研究方向为信息检索、专网网络安全;通信作者:陶 军(1975-),男,教授,博士,CCF 高级会员(42637S),研究方向为网络安全、物联网技术等。

无论是对流量进行解析或者其他操作,第一步都是识别出流量所属协议。而一些协议解析工具或者防火墙针对协议的初步识别主要都是基于固定端口号的,然后再对使用该端口号的协议进行下一步的解析,而一旦协议改变端口号,除非解析工具或者防火墙同步改变配置,否则可能会导致无法识别该协议。Moore 等[3]与 Madhukar 等[4]通过实验证实仅通过端口进行协议识别准确率已经降低到 20% 以下。

白惠文等[5]提出了一种基于 CNN 的针对匿名协议的识别方法。他们将数据包序列预处理为二维向量,并将二维向量中的每个值作为像素值,由此把二维向量转换为图像作为 LeNet-5 网络的输入来训练模型和识别目标流量。对由匿名协议流量和背景流量构成的数据集进行了实验,识别准确率可以达到 96% 以上。

基于有效载荷的识别技术可以得到较好的性能。该方法主要使用深度包检(DPI)[6]技术。DPI 不仅会对数据包的头部字段进行检测,还会对应用层的部分载荷或者所有载荷进行识别,DPI 技术首先会对目标协议进行特征提取,找到协议中的特征字符串等特征,组成协议识别特征库,然后通过特征匹配的方式进行协议的识别,如果网络协议中出现相应的特征串,则可以确定协议的类型。例如,"GET"就是 HTTP 协议的一个特征字符串。

在协议特征准确率较高的情况下,基于有效载荷的协议识别技术是较为准确的一类协议识别方式。因此该方法应用范围广泛,不仅能识别出单协议数据流,还能有效识别出使用端口伪装技术或动态端口技术的流媒体数据以及大多数 P2P 流量。在工业界应用广泛,是高速网络环境部署的最佳选择[7]。

对于公共协议,可以通过查阅协议规范文档获取协议特征。但对于未知协议,早期的特征提取主要采用人工分析的方式寻找,需要具备一定的专业知识。胡梁眉等[8]利用逆向分析的方法,通过网络报文序列采样对比,对相似性极高的报文合并分析。通过统计报文中的关键字节变化率与取值范围等数据特征,对核电工控系统的私有协议进行了识别。

随着技术的发展,许多数据挖掘[9]技术被用于未知协议的特征提取。Wang 等先后提出了 Biprominer[10]和 ProDecoder[11],将数据挖掘用于协议关键字的提取,利用 N-gram 之间的潜在关系来推断协议的格式。Luo 等[12]提出了一种基于字符串频率的关键字提取方法 AutoReEngine,在确定字符串的长度时分别从消息和行的头部、尾部分别进行定位。为了降低二进制逆向协议过程中频繁项提取耗时的问题,Hei 等[13]采用 Apriori 算法生成频繁字节项,然后采用 AC 算法进行频繁项匹配,并通过位置关联来保证特征候选集的完整性。

Agrawal 和 Srikant 提出的 Apriori 算法[14]是频繁项挖掘中最具有影响力的算法,后续许多算法都是基于 Apriori 算法进行改进的。但是其缺点在于得到的结果需要再进行去冗余化处理。

基于此,该文提出了一种自顶向下的频繁字符串挖掘方法。与传统的自底向上式挖掘方法不同,该方法可以直接得到长度最长的频繁字符串,而不用进行去冗余化处理。这一特性,使其能更好地满足利用机器学习方法进行工控协议识别时对于特征字符串的要求。在单协议测试环境和多协议测试环境下,利用识别率、准确率这两个指标对提取的识别特征的正确性和可靠性进行了验证,保证了特征的准确性和完全性。

1　相关概念定义

在进行协议频繁项提取之前,需要给出以下定义。

协议的种类繁多,既有文本型协议,也有二进制协议,但是所有的协议都由整数个字节构成。在此,该文借鉴了 AutoReEngine 中的定义,将构成协议的字节集合 D 表示如下:

$$D = \{\backslash 0x00, \backslash 0x01, \cdots, \backslash 0xFF\} \tag{1}$$

D 中的每一个元素代表一个单字节,而由 D 中的字符构成的序列 M 可以表示为:

$$M = \{\overline{d_1 \cdots d_l \cdots d_n} \wedge d_l \in D, l = 1, 2, \cdots, n_k\} \tag{2}$$

而由多条同种协议组成的报文集合 Z 表示如下:

$$Z = \langle M_1, M_2, \cdots, M_n \rangle \tag{3}$$

定义 1 支持度:对于一个字符串 str,其支持度 $R_s(\text{str})$ 定义为包含 str 的流量数与数据集 Z 中流量总数的比值,即:

$$R_s(\text{str}) = \frac{\{s \mid \text{str} \in M, M \in Z\}}{|Z|} \tag{4}$$

如果 str 在一条流中多次出现则按一次计算。

定义 2 频繁字符串:在应用层协议中多次出现的字符串序列,位置一般固定,也有部分协议频繁字符串出现的位置不固定。该文将字符串出现频率不低于给定的最小支持度 $R_{\min}$ 的字符串作为频繁字符串。也就是说频繁字符串集合表示为:

$$F = \{\text{str} \mid R_s(\text{str}) \geqslant R_{\min}\} \tag{5}$$

也就是说,如果一个字符串是频繁字符串,那么其在整个数据集中出现的频次需要大于一个阈值 Threshold,阈值为数据集 Z 中的流量总数与最小支持度 $R_{\min}$ 的乘积,即:

$$\text{Threshold} = |Z| * R_{\min} \tag{6}$$

除此之外,将由所有长度为 k 的频繁字符串构成的频繁集定义为 F_k。

定义 3 特征字段:包括协议常量、状态代码、分隔符等,用于标识消息中字段的边界并指示字段语义。例如,在协议 S7 COMM 中,其应用层数据前两个字节总是为 0x\03,0x\00。而第 4 ~ 6 字节总是 0x\02,0x\f0,0x\80。而事实上,这些特征字段正是需要寻找的频繁项。

定义 4 完全字符串:如果字符串 str 是 str′ 的一部分,那么 str 叫做 str′ 的子字符串,对于字符串 str″,如果数据集中没有 str″ 的子字符串,那么 str″ 就叫做完全字符串。

2 算法设计

该文尝试对频繁项挖掘算法进行改进,不再使用自底向上的方式进行频繁项的挖掘与构造,而是以空间换时间,采用自顶向下的思路来查找频繁字符串。改进的算法同样基于以下直觉:如果某个项集是频繁的,那么其所有子集也是频繁的。因此若是能够率先找到长度为 k 的频繁项集 F_k,并对 F_k 中每个频繁字符串的位置信息进行记录,在查找 F_{k-1} 时,对 F_k 中相关位置的序列便不再进行统计。这样的话,既不会导致频繁项出现冗余,也不会导致一些关键信息被忽略。方法描述如下。

2.1 数据划分

在这一步,借鉴了 N-gram 模型[15] 对数据进行划分,N-gram 模型的思想是将分组分解为多个长度相等的序列。首先,需要确定一个初始的频繁项长度 k,然后,将数据集中所有的序列分别划分为 $n-k+1$ 个长度为 k 个字节的子序列。

例如,对于序列 $M=\overline{d_1d_2\cdots d_n}$,可以划分为 $n-k+1$ 个子序列:

$$\{s_1,s_2,\cdots,s_{n-k+1}\mid s_i=\overline{d_id_{i+1}\cdots d_{i+k}}\} \tag{7}$$

2.2 频繁项挖掘

对于已经划分好的数据集,在其中查找长度为 k 的频繁项集 F_k。在这一步需要对数据集中所有长度为 k 的序列统计其出现频次,如果一个序列在一条流中多次出现则按一次计算。为了方便之后的操作,还需要对每个序列出现的位置进行记录。如果某个序列当前出现次数已经大于阈值 Threshold,则将该序列加入频繁项集合 F_k 中,并将该序列及与该序列的多个字节存在重叠的相邻序列删除,然后对剩下的序列执行分裂操作,分为两个长度为 $k-1$ 的子序列,例如序列 $\overline{d_1d_2\cdots d_k}$ 可以分裂为 $\overline{d_1d_2\cdots d_{k-1}}$ 和 $\overline{d_2d_3\cdots d_k}$。特别的,如果要删除的序列位于头部或者尾部,则需要保留未重叠的字段,删除重叠的字段。接下来返回第二步继续查找长度为 $k-1$ 的频繁项。该流程会一直重复直到 $k=1$ 时停止查找。该过程如算法 1 所描述。

算法 1:查找频繁项集 F

Input:dataset Z, length k, minimum support $R_{\min}$

Output:frequent items set F

1: $F=\{\}$, Threshold $=R_{\min}*|Z|$
2:FOR all $M_i\in Z$ DO
3: 将 $\boldsymbol{M}_i$ 划分为 $\boldsymbol{n}-\boldsymbol{k}+1$ 个长度为 k 的序列 $\boldsymbol{S}_i$
4:WHILE $k\geqslant 1$ DO
5: $F_k=\{\}$,dict={}
6: FOR $j\in[0,S_i.\text{size})$
7: dict. $S_i[j]+1$,dict. $S_i[j].j+1$
8: IF dict. $S_i[j]>$ Threshold
9: update $F_k.S_i[j]=$ dict. $S_i[j]$
10: END IF
11: END FOR
12: 从所有 $\boldsymbol{S}_i$ 删除相关位置项并将其分裂为长度为 $k-1$ 的项
13: ADD F_k to F
14: $k=k-1$
15:END WHILE

相关流程示意如图 1 所示。在图中,以字节为单位,首先取 $k=3$ 个字节划分序列,假设序列'000001'为长度为 3 个字节的频繁项,那么删除以及拆分后的结果如下。可以看到,所说的删除并不是将相关序列从列表中移除,而是将这些序列置为空字符串,因为这样处理不会打乱每个序列在原数据中的位置。最终得到的每一个频繁项的表示如下:

```
{
Frequent item:
[
Total times,
{position 1: times[,position 2:times]}
]
}
```

其中,Frequent item 表示频繁字符串,Total times 表示该频繁字符串总共在多少个数据包中出现过,position n 代表该字符串在第 n 个位置出现了 times 次。

例如,当 $k=9$ 时,CIP PCCC 协议的一个频繁项如下:

{"4b0220672401074d00":[824,{40:418,46:406}]}

表示"4b0220672401074d00"这个字段总共出现了 824 次,其中在第 40 个字节出现了 418 次,第 46 个字节出现了 406 次。

2.3 合并频繁项集

在确定了初始的 k 值后,通过第二步便可以得到长度不同的频繁项集 $F_k,F_{k-1},\cdots,F_1$,那么,初始的完全频繁项集合如式 8 所示:

$$F = \bigcup_{i=k}^{1} F_i \tag{8}$$

这样一来,整个频繁项的拆分删除工作可以用图 1 来表示。

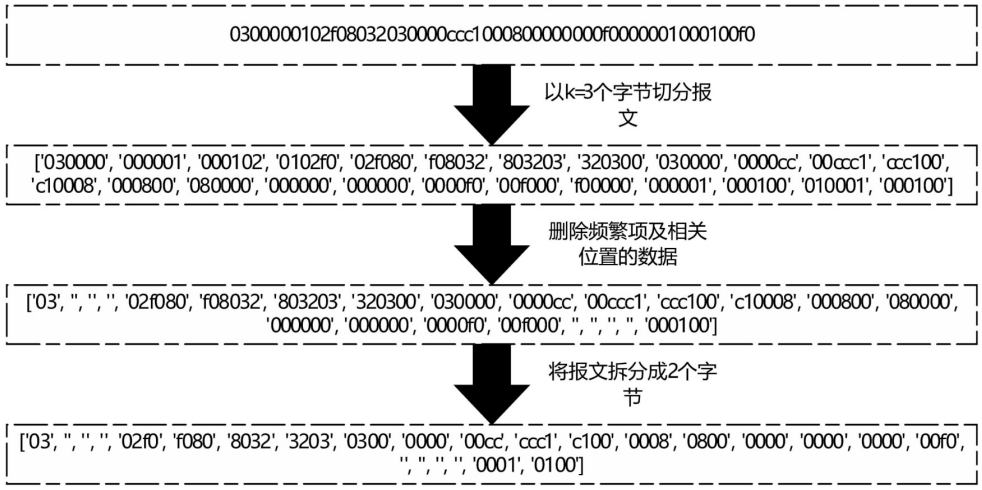


图 1　频繁项删除及拆分示意图

2.4　初始长度的确定

上述所有工作中,一切前提都是建立在初始长度 k 已知。然而实际情况下,初始的 k 值还需要人工确定。那么应该如何确定初始的 k 值? 显然, k 的取值不能过大也不能过小,因为如果初始的 k 值设置过大,那么在找到第一个频繁项之前,将花费大量无意义的时间在序列的统计与拆分之上,而一旦 k 的初始值设置过小,则会导致之前提到的数据冗余的问题出现。因此需要找到合适的初始值。该文使用改进的二分查找法来进行初始 k 值的确定,其思路如下:

(1)对于数据集 Z 中的所有消息进行扫描,选取其中最短的消息序列的长度 $l_{\min}$ 作为 k 取值的右边界 R ,以 1 作为左边界 L 。

(2)取 $k = \dfrac{L+R}{2}$ 开始进行数据划分,并统计长度为 k 个字节的频繁字符串。

(3)如果当前未找到频繁字符串,则更新右边界 $R = k - 1$,回到第 2 步。

(4)如果当前长度的频繁项数目大于等于 2 且部分频繁项之间存在交叠,则更新左边界 $L = k + 1$,同时为了减少查找次数,可以将存在交叠字段的频繁项进行合并,并以合并后的最长序列长度 l_m 更新右边界 R ,然后回到第 2 步。

例如,如果查找出长度为 4 的频繁集为 [′1234′, ′2345′, ′3456′, ′3214′] ,那么合并后的结果为 [′123456′, ′3214′] ,那么此时 R 更新为 6, L 更新为 5。

(5)而如果长度为 k 的频繁集只包含一个元素或者多个无交叠字段的元素,那么当前 k 值便是最终的 k 值,算法结束。过程如算法 2 所描述。

算法 2:确定初始长度 k

Input: dataset Z , minimum support $R_{\min}$

Output: Initial k value

1. Initiation: $L = 0, R = l_{\min}$, Threshold = $|Z| * R_{\min}$
2. WHILE $L <= R$ DO
3. $k = \dfrac{L+R}{2}$, dict = {}
4. FOR $M \in Z$ DO
5. 将 M 划分为多个长度为 k 的序列 Seq
6. FOR seq ∈ Seq DO
7. dict[seq]+1
8. IF dict[seq]>=Threshold THEN
9. update F_k[seq] = dict[seq]
10. END IF
11. END FOR
12. END FOR
13. IF F_k. size == 0 THEN
14. $R == k - 1$
15. ELSE IF F_k. size > 1 or THEN
16. 合并频繁项,计算合并后序列长度 l_m
17. let $L = k + 1$, $R = l_m$
18. ELSE
19. BREAK
20. END IF
21. END WHILE

在完全频繁项集 F 中,由于在挖掘过程中便考虑到了去冗余问题,因此不存在冗余问题,但是还是可能存在不可靠的频繁字符串,例如,在二进制协议中长度为 1 的字节 $0x\backslash 00$ 几乎在每条流量中都有出现,然而出现的位置却不一定固定,甚至可能有很大偏差,而类似于这样的字段显然不适合作为特征字段来进行协议的识别。因此,需要过滤该类弱特征,该文考虑了频繁字符串在所有流量中的位置信息,进而从频繁项集 F 中推断特征字段。

前文提到,频繁项集中记录了各个频繁字符串在各个位置出现的频次,需要据此来进行频繁项的过滤,筛选出用于识别的特征字段。可以确定的特征字符串如下。

如果该频繁字符串在某个位置出现次数大于阈值 Threshold ,那么该字符串为位置固定的一个特征字段。

如果某个字符串出现位置虽然不固定,但是各个位置出现次数之和等于或者十分接近该字符串出现总次数,那么该字符串也是特征字符串,虽然其出现位置不固定,但是一般只有有限几个位置,且很少在一个数据包里重复出现。

如果频繁字符串虽然出现位置多变,但是自身长度较长,该文也将其保留对于其他较短的频繁字符串,通过其在数据包中出现位置的方差来进行保留或者过滤。即考虑字符串位置到包含该字符串的消息开头的偏移量,将其表示为 P 。对于需要保留的关键字,其方差 P 应当非常小。

至此,便可以得到协议识别的特征字段。

```
<feature protocol = " cippccc" >
    <bytes>
      <content>0x4b0220672401074d00</content>
        <position>40</position>
        <position>46</position>
        <offset>9</offset>
    </bytes>
    <bytes>
        <content>0x0000000000000000</content>
        <position>20</position>
        <offset>8</offset>
    </bytes>
    <bytes>
        <content>0x0200</content>
        <position>30</position>
        <offset>2</offset>
    </bytes>
    <content>0x00</content>
            <position>1</position>
            <offset>1</offset>
</feature>
```

图 2 CIP PCCC 协议特征字符串

图 2 为 CIP PCCC 协议的识别特征中特征字符串的表示。其中< feature protocol = " cippccc" > </feature >表示一个 CIP – PCCC 识别特征的开始与结束。< bytes> </bytes>之间表示具体位置的字节内容,offset 表示该特征字段包含多少字节,position 表示特征字段起始位置,当同一个<bytes> </bytes>之间出现多个 position 字段时,只要该特征字符串与其中任意一个位置匹配成功即可。如果一条流量与特征中所有字段都匹配成功,便认为这条流属于 CIP PCCC 协议。

3 实验验证

3.1 评估指标

为了验证算法的正确性与性能,主要针对 Omron Fins,Modbus,S7 COMM 等 10 种协议进行了分析,对其进行了特征字符串的提取。并将提取出来的特征字符串作为识别特征,进行了协议识别工作,通过识别率、准确率对识别的效果进行了验证。

定义 5 识别率:在单协议测试环境下,使用提取得到的某种工控协议的识别特征识别的流量占该协议总流量的百分比。该指标能够度量挖掘的协议识别特征的完全性。

定义 6 准确率:在单协议识别环境中,被识别为某种工控协议的流量中真正是该协议的流量所占的百分比。准确率主要用来验证挖掘的协议识别特征的正确性。

3.2 特征字符串提取效果

针对 Omron Fins,Modbus,S7 COMM 等 10 种协议的特征字符串提取结果如表 1 所示。

表 1 不同协议的特征字符串

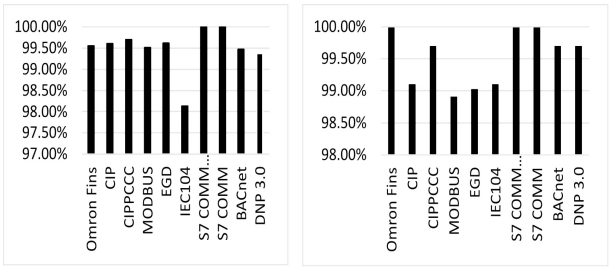
协议	特征字符串	起始位置
Omron Fins	0x46494e53	0
	0x00000000	12
CIP	0x0000000000000000	20
	0x2000	30
	0x00	1
	0x4b0220672401074d00	(40,46)
CIPPCCC	0x0000000000000000	20
	0x2000	30
MODBUS	0x00	1
EGD	0x0000	2
IEC104	0x0d01	0
S7 COMM PLUS	0x0d01	0
	0x0300	0
S7 COMM	0x02f08072	4
	0x0300	0
	0x02f08032	4
BACnet	0x0000	9
DNP 3.0	0x81	0
MODBUS	0x0564	0

可以看到,对于每种协议,都提取出了相应的特征字符串。

3.3 协议识别效果

利用提取出的特征字符串进行协议识别工作,识

别结果如图 3 所示。



(a)协议识别的识别率　　(b)协议识别的准确率

图 3　协议识别的结果

对协议识别的结果,对其识别率和准确率进行了统计。识别率如图 3(a)所示。可以看到,对于选取的 10 种协议,其识别率均能达到 98% 以上。其中除了对 IEC104 协议的识别率在 98% 左右,其余 9 种协议的识别率均达到了 99% 以上,对于 S7COMM 和 S7COMMON PLUS 协议甚至可以达到接近 100% 。

而协议识别的准确率则如图 3(b)所示。可以看到对于选取的 10 种协议,准确率基本都可以达到 99% 以上。对 CIP,MODBUS,EGD,IEC104 等协议的准确率在 99% 左右,而对 Omron Fins,S7COMMOM,S7COMMON PLUS 等协议的准确率接近 100% 。

综合识别率和准确率来看,利用文中算法所提取出的频繁字符串作为特征进行协议识别,可以得到非常准确的结果。对于 S7COMMON,S7COMMON PLUS 等特征字符串较多的协议,甚至可以得到接近 100% 的识别率和准确率。这说明文中算法所提取出的频繁字符串确实是协议的特征字符串。

4　结束语

在对工控协议的识别工作中,基于有效载荷的识别方法逐渐成为主流。在这种趋势下,对工控协议中识别特征的选择和提取工作尤为重要。频繁字符串由于其出现频率较高,作为协议识别中的特征字符串极为合适。针对工控协议中的频繁字符串,设计了一种自顶向下的提取算法,使提取出来的字符串无需再进行去冗余化处理。同时针对传统自顶向下算法中的初始数据过大、步骤较多等问题进行了优化。实验数据表明,该算法对大多数协议都能提取出有效的频繁字符串。

参考文献:

[1] FARWELL J P,ROHOZINSKI R. Stuxnet and the future of cyber war[J]. Survival,2011,53(1):23-40.

[2] STOUFFER K,FALCO J,SCARFONE K. Guide to industrial control systems (ICS) security[R]. [s. l.]:NIST,2011.

[3] MOORE A W,ZUEV D. Internet traffic classification using bayesian analysis techniques[C]//Proceedings of the 2005 ACM SIGMETRICS international conference on measurement and modeling of computer systems. Banff:ACM,2005:50-60.

[4] SANDERS C. Practical packet analysis,3E:using wireshark to solve real-world network problems[M]. San Francisco:No Starch Press,2017.

[5] 白惠文,马雪婧,刘伟伟,等. 基于深度学习的匿名协议流量识别技术研究[J]. 计算机仿真,2021,38(7):360-365.

[6] ANTONELLO R,FERNANDES S,KAMIENSKI C,et al. Deep packet inspection tools and techniques in commodity platforms:challenges and trends[J]. Journal of Network and Computer Applications,2012,35(6):1863-1878.

[7] BUJLOW T,CARELA-ESPAÑOL V,BARLET-ROS P. Independent comparison of popular DPI tools for traffic classification[J]. Computer Networks,2015,76:75-89.

[8] 胡梁眉,李　实,朱航潇,等. 核电工控系统的协议解析与行为审计[J]. 工业控制计算机,2021,34(6):94-96.

[9] KLEBER S,MAILE L,KARGL F. Survey of protocol reverse engineering algorithms:decomposition of tools for static traffic analysis[J]. IEEE Communications Surveys & Tutorials,2018,21(1):526-561.

[10] WANG Y,LI X,MENG J,et al. Biprominer:automatic mining of binary protocol features[C]//2011 12th international conference on parallel and distributed computing,applications and technologies. Gwangju:IEEE,2011:179-184.

[11] WANG Y,YUN X,SHAFIQ M Z,et al. A semantics aware approach to automated reverse engineering unknown protocols[C]//2012 20th IEEE international conference on network protocols (ICNP). Austin:IEEE,2012:1-10.

[12] LUO J Z,YU S Z. Position-based automatic reverse engineering of network protocols[J]. Journal of Network and Computer Applications,2013,36(3):1070-1077.

[13] HEI X,BAI B,WANG Y,et al. Feature extraction optimization for bitstream communication protocol format reverse analysis[C]//2019 18th IEEE international conference on trust, security and privacy in computing and communications/13th IEEE international conference on big data science and engineering (TrustCom/BigDataSE). Rotorua: IEEE,2019:662-669.

[14] AGRAWAL R,SRIKANT R. Fast algorithms for mining association rules[C]//Proc. 20th int. conf. very large data bases. Santiago de Chile,:VLDB,1994:487-499.

[15] BERMUDEZ I,TONGAONKAR A,ILIOFOTOU M,et al. Towards automatic protocol field inference[J]. Computer Communications,2016,84:40-51.