基于特定模式树的用户行为关联规则挖掘算法①

The Algorithm of users behavior associate rules mining Based on Specific Pattern Tree

戴 臻 (中南大学信息科学与工程学院 长沙 410075)
(湖南科技职业学院软件学院 长沙 410118)
费洪晓 李 俊 (中南大学信息科学与工程学院 长沙 410075)
谢文彪 (中南大学信息科学与工程学院 长沙 410075)
(长沙理工大学电气与信息工程学院 长沙 410076)
肖新华 (中南大学信息科学与工程学院 长沙 410075)

摘要:Apriori 算法是关联规则挖掘的通用算法,它能满足绝大多数的应用,但是在某些方面,如入侵检测中挖掘用户活动记录等具有特定模式的记录时,计算最大频繁集会产生大量冗余的、无趣的规则。论文在 Apriori 算法的基础上针对上述情况提出了一种基于特定模式树的算法,消除无趣项的产生,通过递归挖掘模式树获得最大频繁集。整个过程只需要扫描一次数据库,进一步提高了算法效率。

关键词:关联规则 特定模式树 最大频繁集 入侵检测

1 引言

在互联网日益发达的今天,计算机网络提供了人们方便快捷的生活方式,成为了信息社会发展的的重要保证。同时安全问题也成了不容忽视的重要课题。传统的基于手工和经验的入侵检测(Intrusion Detection System,IDS)不能适应当前不断更新的攻击手法,而基于数据挖掘的 IDS 能很好地解决这一问题[1]。

数据挖掘本身是一项通用的知识发现技术,其目的是要从海量数据中提取出人们感兴趣的数据信息[2]。将数据挖掘技术应用到入侵检测领域,构建基于数据挖掘的入侵检测系统,其主要特点是:它能够自动地从大量的网络数据中构建简洁、精确的正常的或入侵行为模式,解决了传统入侵检测系统手工分析以及对攻击模式进行硬编码的沉重负担。由于该方案具有通用性和自动处理的功能,因而可在许多不同的网络环境中构建相应的 IDS[3]。

利用数据挖掘中的关联规则挖掘,结合入侵检测实际运行环境对网络和主机审计记录挖掘,挖掘出用户的行为模式,建立用户活动简档。一般情况下,用户行为记录具有特定的模式,每条记录有相对固定的属性集,且不同属性在实际中的重要性也有区别,本文提出了一种基于特定模式树的挖掘算法,避免了无趣规则的产生,只需扫描一次数据库,通过对特定模式树的挖掘得到用户的行为模式。

2 关联规则在用户行为模式挖掘中的应用

关联规则挖掘用于分析同一记录中不同属性之间的联系。在一个主机上,每个用户都具有相对固定的行为习惯,长时间保持不变。例如,用户 Lee 每天都是晚上登陆主机,且调用的进程都是用户级别的进程,因此 Lee 的行为记录中时间属性为"night"同时调用进程级别属性值为"user level"的记录占 Lee 所有记录的大部分。通过对大量记录的挖掘,能知道 Lee 的使用习惯,为以后判断 Lee 行为的风险度提供了依据,若某次 Lee 突然调用内核级别的进程,则可以提

① 基金项目:国家自然科学基金资助项目(60673165);湖南省自然科学基金资助项目(05JJ30119)

出警告，可能是入侵行为。

运用关联规则挖掘建立用户正常行为模式库，作为入侵检测中用户行为的参照，可以正确区分正常行为和入侵行为。

关联规则算法主要有 Apriori 系列算法和 FP－树算法，Apriori 系统算法包括 Appriori，DHP，抽样处理算法，分块处理算法等[4,5]。它们都是通用的关联规则挖掘算法，能适用于大多数情况。正因为它们的广泛适用性，使它们在特定情况下不能有很好的效果，执行效率和准确率都有欠缺。例如，Apriori 算法需要多次扫描数据库，这成了算法性能的瓶颈，在建立候选频繁集的过程中，它采用的是全连接的方式，这是低效且盲目的，会产生很多无趣候选项。FP－树算法构建树的是按记录中 1－itemset 项频繁度计数递减的方式进行的，这样也会产生很多无意义规则。本文提出的特定模式树算法针对上述问题进行了优化，它构建树是按记录中属性重要性递减的顺序进行的，不会产生无趣规则，且只需要扫描数据库一次即可。

3　基于特定模式树的关联规则挖掘算法

该算法首先通过预处理把用户行为记录抽象成形式化符号表，然后根据用户行为描述特征参数的重要度排序构成特定的顺序行为模式树，最后设置最小支持度挖掘模式树得到用户行为频繁模式集。

3.1　数据预处理

用户行为记录属于一种特定的类型。这种类型的记录具有固定的成份，即每条记录包含的属性集是一样的，不同记录中相同属性的属性值可能不同。同时，在特定的应用环境下，一条记录中，不同属性之间的重要性(weightiness)是有区别的。例如，一条记录可以表示成<$Attri_1$，$Attri_2$，$Attri_3$，… $Attri_n$>的形式，$Attri_i$ 代表属性 i 的值。属性之间的重要性关系约定为：

$Attri_1 >= Attri_2 >= Attri_3 >= \cdots >= Attri_n$

在入侵检测系统中，用户行为记录满足上述形式，本文选择 Linux 下用户 shell 命令调用记录为实验数据。命令记录可由以下属性组成：Username(用户名)，Timestamp(时间戳)，Command(命令)，Param(命令参数)。根据需要，记录中 Username 属性是关键属性，它的重要性是最大的，其它属性的重要性都比它小，是非关键属性。

将收集的用户数据用上面的格式处理后加入到数据库中，表 1 截取了其中一部分数据。

表 1　用户行为记录表示

ID	Username	Timestamp	Command	Param
1	Li	AM	Emacs	.cpp
2	Wang	FM	yum	update
3	Xu	FM	Gcc	－o
4	LI	AM	Emacs	.cpp
…	…	…	…	…

在利用关联规则挖掘用户行为模式，就是对数据库中按规定格式存储的记录数据进行挖掘，找到记录属性之间的联系。

3.2　算法描述

基于特定模式树的关联规则挖掘用户行为首先要通过扫描用户记录数据库建立特定模式树，然后对特定模式树进行自上而下的递归挖掘，得到最大频繁集。

(1) 构建特定模式树。表 1 中属性值形式差别很大，不利于简洁表示，因此将表 1 转换成表 2 的符号表示形式。

表 2　用户行为记录符号表示

$Attri_1$	$Attri_2$	$Attri_3$	$Attri_4$	$Attri_5$
T_1	I_1	I_3	I_4	I_6
T_2	I_1	I_3	I_4	I_6
T_3	I_1	I_3	I_4	I_7
T_4	I_1	I_3	I_5	I_6
T_5	I_1	I_3	I_5	I_7
T_6	I_2	I_3	I_4	I_6

其中 $Attri_1$，$Attri_2$，$Attri_3$，$Attri_4$，$Attri_5$ 分别代表属性 ID，Username，Timestamp，Command，Param。这些属性中，Username 是关键属性，每条挖掘得到的规则中都应包括关键属性。

构成模式树的结点定义如下：

```
typedef struct NODE{
  string NodeName; //结点内容名字
  vector <NODE * > Childern; //指点结点孩子的指针集
  int Count; //结点的计数
  };
```

构建特定模式树时，每次从数据库中读入一条记录，就将此记录中的属性按照重要性从大到小的顺序重新排列，再加入到树中。表 2 中的记录构成特定模

式树的过程如下：

模式树的根结点 root 是一个除了孩子结点信息外，其它域为空的节点。用 NULL 标记该节点。扫描数据库，每个事务重新排列属性顺序，表 2 中记录在数据库中的存储就是按照属性重要性递减的顺序排列的，因为重排列后顺序不变。接着按以下步骤将记录加入到特定模式树中。例如，第一个事务“T_1：I_1，I_3，I_4，I_6”包含四个项{I_1，I_3，I_4，I_6}，构成了模式树的第一个分枝，该分枝具有四个节点，I_1 作为根节点的孩子链接到根，I_3 链接到 I_1，I_4 链接到 I_3，I_6 链接到 I_4。每个节点的计数域初始为 1。第二个事务是第一个的重复，沿着每一个事务的路径各个节点计数加 1。第三个事务和第二个有相同的前缀{I_1，I_3，I_4}，沿着前缀路径各个节点计数加 1，在第四个属性处产生了分枝，因此节点 I_4 新增一个孩子节点 I_7，其计数初始化为 1。一般地，当为一个事务考虑分枝时，沿共同前缀上的每个节点的计数增加 1，为跟随在前缀之后的项创建节点并链接。

根据表 2 创建的特定模式树如图 1 所示。

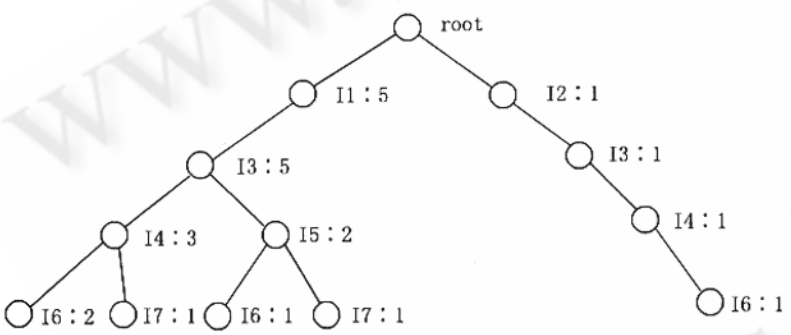

图 1　用户行为模式树

4　挖掘特定模式树

特定模式树的挖掘从根节点开始，对每个节点进行如下操作，访问节点的各个子节点（child node）计数域，若子节点计数小于最小支持度计数，终止在此分枝上的挖掘，若子节点计数不小于最小支持度，将此节点加入先前成功挖掘的部分路径中，再递归挖掘此子节点的子节点（child's child node）。新建另一个分枝，跳过子节点，直接挖掘子节点的子节点，子节点的子节点中相同的节点合并，新的计数值是合并结点计数值的和，在此分枝上递归挖掘。按照以上方法自项向下挖掘，当挖掘到叶子节点时，将传递下来的挖掘路径加入到最终的频繁模式集（frequent_patterns）中。

算法描述如下：

```
void pattern_gen( Min_Sup, NowPattern, Node * node )
  {
    if ((node->children).size() == 0 )
    {
      将当前模式 NowPattern 加入到 frequent_patterns 中;
      Return;
    }
    for_each Node * p ∈ node->Children
      if( p -> Count >= Min_Sup )
        Pattern_gen( Min_Sup; NowPattern + (p->Nodename), p);
    Node * sub= sub_tree_gen(node);
    pattern_gen(sub);
    return;
  }
      Node * sub_tree_gen(Node * node)
      {
        新建子树根节点 Node * sub;
        将 node 节点的孙子节点作为 sub 的儿子节点加入到 sub->Children 域中，合并产生的重复分枝;
        return sub;
      }
      void free_tree( Node * node ); //释放 node 及其后继节点所占内存空间;
```

在图 1 中，当最小支持度为 50%时，从 I1 开始的连续路径{I_1，I_3，I_4}满足最小支持度，故将它以及它的子集{I_1，I_3}，{I_1，I_4}，{I_3，I_4}加入到频繁模式集中。进一步观察发现 I_6 分布在 I_4，I_5 两个分枝上面。进行移除 I_4，I_5 时，I_6，I_7 由 I_3 的孙子节点变为了 I_3 的儿子结点，I_6 的两个分枝合并，变换过程如图 2。

通过变换能得到另一路径{I_1，I_3，I_6}也满足最小支持度，故它以及它的子集{I_1，I_6}，{I_3，I_6}（{I_1，I_3}之前已经加入了）可以加入到频繁模式集中。

5　小结

通过实验和比较分析新的基于特定模式树的算法较 Apriori 算法有以下优点：

（1）预防了无趣项的产生。Apriori 算法中 C_i 是

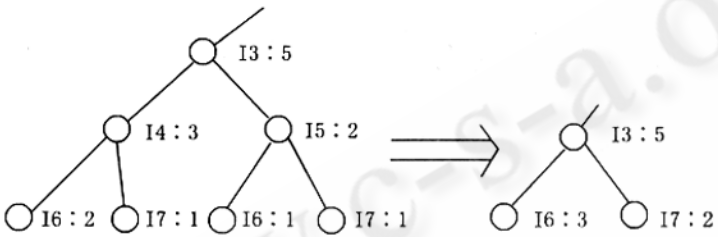

图 2　sub_tree_gen()过程

由 C_{k-1} 连接而成，从而可能产生诸如{Li，AM，AM，su}这样的候选项，而这是无意义的模式，浪费了时间比较计算这种无趣项。而新算法采用了树作为整体结构，树的每一层代表了一个属性，因此树的深度就是用户行为记录的属性个数，这样的结构以及它的挖掘算法保证了同属性的值不会出现在同一候选模式中。

(2) 采用树的结构，只需扫描一次数据库。论文的方法吸收了 FP－tree 算法的优点，采用树的存储结构，大大节约了存储空间，一般情况下只需扫描一次数据库就可以了。

(3) 执行速度较 Apriori 快，具有实用性。

参考文献

1 Wenke Lee, Stolf. S.J, Mok. K.W. A data mining framework for building intrusion detection models. Security and Privacy. Proceedings of the 1999 IEEE Symposium on May 1999:120－132.

2 Adriaans, P. and D. Zantinge, Data Mining, Addison－wesley, New York, 1996.

3 李玲娟、王汝传，一种基于数据挖掘的入侵检测系统模型，南京邮电学院学报，2005，25(5)：40－43.

4 Lazem, S.? Adly, N. Nagi, M. A new index structure for querying association rules. Advanced Information Networking and Applications. AINA 2006 20th International Conference on, April 2006 (2):5pp.

5 Tian－rui Li, Wu－ming Pan. Intrusion detection system based on new association rule mining model. Granular Computing, 2005 IEEE International Conference on July 2005(2): 512－515.