基于改进 FP growth 的告警关联算法

鲁显光 杜学绘 王文娟
(信息工程大学 郑州 450001)

摘 要 入侵检测系统产生的原始告警存在层次较低、相互孤立、没有关联性等不足,使得安全管理人员难以从中发现未知的、高层次的安全威胁,从而无法了解目标网络的整体安全态势。为了利用低级别告警构建攻击场景,通过分析现有的告警关联知识,针对基于数据挖掘的告警关联算法处理稀疏数据时性能较差的不足,提出了一种新的基于数据挖掘的告警关联算法。首先对现有的告警关联算法进行了分析比较;然后阐述了经典的 Apriori 算法和 FP growth 算法的机制及优缺点,并基于二维表对 FP growth 算法进行了改进;最后使用改进算法挖掘告警之间的关联规则,继而进行告警关联。为了验证所提方法的可行性和性能,使用 Darpa 数据集进行了相关的仿真测试,实验结果表明该方案可以较好地实现告警关联。

关键词 入侵检测,告警关联,关联分析,FP growth 算法
中图法分类号 TP393.08 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.08.010

Alert Correlation Algorithm Based on Improved FP Growth

LU Xian-guang DU Xue-hui WANG Wen-juan
(Information Engineering University, Zhengzhou 450001, China)

Abstract The original alerts generated by intrusion detection system have some shortcomings, such as low level, mutual isolation and irrelevance, which makes security managers be difficult to find unknown and high-level security threats and cannot understand the overall security situation of the target network. In order to make use of low-level alerts to construct attack scenarios, this paper analyzed the existing alert correlation knowledge, and proposed a new alert correlation algorithm based on data mining to solve the problem of poor performance of existing algorithms when dealing with sparse data. In this paper, firstly, the existing alert correlation algorithms were compared, then the principles and merits and demerits of classical Apriori algorithm and FP growth algorithm were elaborated, and the FP growth algorithm was improved based on two-dimensional table. Finally, the improved algorithm was used to mine the association rules between the alerts, and thus the alert correlation was proceeded. In order to verify the feasibility and performance of the proposed method, the Darpa data set is utilized to carry out relevant simulation tests. The experimental results show that the proposed scheme can achieve better alert correlation.

Keywords Intrusion detection, Alert correlation, Correlation analysis, FP growth algorithm

1 引言

随着技术的不断发展,以高级持续性威胁(Advanced Persistent Threat,APT)为代表的多步攻击已经成为目前网络攻击的主要手段。针对传统入侵检测系统只能检测单步攻击的缺点,研究者们提出了告警关联技术,利用入侵检测系统生成的低级别告警构建较为完整的攻击场景,从而帮助安全管理人员更好地理解网络态势。

目前,专家学者已对告警关联技术进行了大量的研究。按照所采用技术手段的不同,现有的告警关联技术可以分为基于相似性的方法、基于场景的方法、基于因果的方法和基于数据挖掘的方法。

基于相似性的方法主要通过计算告警之间的相似性来进行关联。Valdes 等[1]对告警的每一个属性都定义了相应的相似度计算函数,并使用预定义方程计算告警的整体相似性。高会生等[2]分析了告警属性语义信息对告警聚类的作用,定义了具有分层特点的属性相似度函数。朱丽娜等[3]定义了单步攻击相似度和攻击模式相似度,并引入拓扑排序来合并相似的单步攻击以得到攻击模式,进而通过计算攻击模式的相似度来实现预警。基于相似性的方法不需要大量的先验知识,但是结果无法体现攻击之间的因果关系。

基于场景的方法利用专家知识构建攻击的具体场景,明确发起攻击需要满足的前提。Templeton 等[4]提出了一种名为 JIGSAW 的攻击规范语言,用于指定攻击需要满足的条件。只有当所有前提条件都满足时,攻击才能成功。Morin 等[5]提出了一种名为 M4D4 的数据模型。M4D4 基于一阶逻

到稿日期:2018-11-06 返修日期:2019-02-22 本文受国家重点研发计划(2016YFB0501901,2018YFB0803603)资助。
鲁显光(1994—),男,硕士生,主要研究方向为数据挖掘、网络与信息安全;**杜学绘**(1968—),女,博士,教授,主要研究方向为网络与信息安全,E-mail:dxh37139@sina.com(通信作者);**王文娟**(1981—),女,博士生,副教授,主要研究方向为网络与信息安全、数据挖掘。

辑，可以用来查询安全事件及其发生环境的信息，为预测下一步攻击提供必要知识。基于场景的方法可以较好地实现已知告警的关联，但是无法处理未知的告警。

基于因果的方法主要通过构建因果知识网络，利用攻击的前提和后果来进行关联和推测。Jajodia 等[6]通过有向图描述不同攻击步骤之间的依赖关系，结合目标网络进行告警关联。Yu 等[7]将资源状态、攻击行为及攻击证据作为彩色 Petri 网络的不同类型节点，并将其互相关联以完成攻击预测。王硕等[8]借助因果知识网络，综合利用图关系进行告警映射，实时检测已发生的攻击行为。张靖等[9]根据攻击类型及其前提和后果条件建立攻击规划图模型，并构建攻击规划树，进一步提出了基于攻击规划图的报警关联方法。

基于数据挖掘的方法主要通过机器学习算法发现告警中隐藏的攻击行为模式，来形成关联规则，在此基础上进行攻击场景构建。Norbol[10]进行了基于 PDDL 描述的多步入侵警报关联方法研究，实现了基于 PDDL 的多步攻击描述，为将智能规划方法应用到入侵检测领域奠定了基础。宋珊珊[11]对 WINEPI 算法进行了改进，使之可以更好地服务于告警关联。梅海彬等[12]将告警数据转化为告警序列，根据警报之间的相似度，构建攻击活动序列集，并基于动态规划的思想，通过求序列集的最长公共子序列来发现警报数据中的多步攻击模式。刘敬[13]利用神经网络模型对事件进行融合分析，按照不同攻击场景进行分类，然后基于分类结果提取规则项并组成训练集，规则项以树形结构组成初始关联规则，利用遗传编程方法自动生成针对不同攻击场景的关联规则。李洪成等[14]首先利用 Laplace 机制构建支持差分隐私保护的噪声告警序列前缀树，然后通过遍历噪声前缀树生成泛化告警序列数据集，最后使用 PrefixSpan 算法挖掘告警的频繁模式以实现告警关联。

为了更直观地展示各类方法的特性，我们在表 1 中对不同的方法进行了总结。

表 1　不同方法的比较

Table 1　Comparison of different methods

方法	关联能力	关联精度	关联效率
基于相似性的方法	低(无法体现因果关系)	低(依赖于相似度函数)	高(计算量小)
基于场景的方法	中(依赖于知识库的更新)	高(依赖于专家规则)	低(搜索空间大)
基于因果的方法	较高(可以发现未知关联)	中(依赖于关联的重合度)	低(搜索空间大)
基于数据挖掘的方法	高(可以发现新规则)	较低(新规则准确性难以保证)	低(计算量大)

综合考虑各类方法的优缺点，基于数据挖掘的告警关联方法具有较强的关联能力，这是其他方法难以具备的，因此，本文使用基于数据挖掘的方法，利用改进的 FP growth 算法，实现告警之间关联规则的挖掘。

2　频繁模式挖掘

2.1　Apriori 算法

Apriori 算法[15]是由 Agrawal 等提出的一种经典的关联规则挖掘算法。该算法首先扫描数据库，计算每个项目的支持度，然后使用前一次迭代中发现的频繁 $(k-1)$ 项集迭代地生成新的候选 k 项集。为了计算候选项的支持度，算法需要对数据库进行额外的扫描。在统计候选集的支持度之后，算法消除所有支持度小于最小支持阈值的候选项集，从而生成频繁 k 项集。当不生成新的频繁项集时，算法终止。

在海量数据的情况下，Apriori 算法主要存在两个问题：1)扫描数据库多次，导致时间成本较高；2)操作过程中产生大量的候选集，导致空间成本较高。

2.2　FP growth 算法

FP growth 算法[16]主要包括两个步骤：1)构建频繁模式树(Frequent Pattern Tree，FP tree)；2)从 FP tree 中挖掘频繁项集。

FP tree 的构建需要扫描两遍数据库。第一次扫描对事务中出现的各个项进行计数，将各项按照频数降序排序，删除出现频数少于最小支持度的项，形成列表 L；第二次扫描将数据集中每个事务的项按照列表 L 排序，构造 FP tree 以及树的项头表。

频繁项的挖掘顺序为从下到上，即从项头表中最下面一项开始，依次往上取项。对于每一项的挖掘大体可以分为 3 步：1)在 FP tree 中递归查找对应的条件模式基(Condition Pattern Base)；2)利用条件模式基，构建条件 FP tree；3)重复迭代步骤 1)和步骤 2)，直到树只包含一个元素项为止。

FP growth 算法不需要生成候选集，但是在挖掘频繁模式时需要构建大量的条件模式树，因此在处理稀疏数据集时的表现并不理想。

3　基于 FP growth 的改进算法及在告警关联中的应用

3.1　事务获取

假设对原始告警数据库进行重复去除以及聚合之后，剩余 n 个告警，这些告警聚合成 k 个集群(cluster)。用 a_i 表示告警，对于 $\forall a_i, a_i \in cluseter_j$，其中 $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k$。在告警关联阶段，为了方便表示，使用 I_i 代替 $cluseter_i$。假设告警序列为：$I_3 I_1 I_1 I_2 I_4 I_2 I_3 I_1 I_5 I_1 I_7 I_3 I_3 I_4 I_1 I_5 I_2 I_6 I_2 I_4 I_2 I_4 I_5 I_2 I_1 I_1 I_3 \cdots$。本文使用滑动窗口将告警序列划分为事务集，方法如下：设窗口初始大小为 1，最大长度为 4。窗口左边界初始位置为第一个项目，右边界从第二个项目开始逐一向后扫描，当窗口大小未达到最大长度时，如果新进入窗口的项目与窗口内原有的项目重复，则将当前窗口内的项目存储于事务数据库中，并从当前位置开始建立一个新的窗口；如果窗口已经达到最大长度，而所包含的项目没有重复，则将该窗口内的项目存储于事务数据库，新窗口从旧窗口结束的位置开始建立。这样一直扫描下去，直到得到一个完整的事务数据库。本文使用上述方法对告警序列进行划分，得到的事务集合如表 2 所列。

表 2　事务集合

Table 2　Transaction Sets

事务	项	事务	项
T_1	I_3, I_1	T_6	I_2, I_6
T_2	I_1, I_2, I_4	T_7	I_2, I_4
T_3	I_2, I_3, I_1, I_5	T_8	I_2, I_4, I_5
T_4	I_1, I_7, I_3	T_9	I_2, I_1
T_5	I_3, I_4, I_1, I_5	T_{10}	I_1, I_3

3.2 改进的算法

FP growth 算法在挖掘频繁项集时利用 FP tree 存储关于频繁项集的压缩信息。对于比较密集的，即事务之间共享前缀较多的数据来说，算法的性能非常好，但是如果数据集比较稀疏，则对于一个项要构建多个分支的条件模式树，算法就较为低效。因此，本文考虑对于 FP tree 中较为稀疏的项使用数组查询的方式查找频繁项集，而对于比较密集的项采用从上到下(Top-Down)遍历的方法查找频繁项集。

3.2.1 构建频繁模式树

遍历事务数据库，得到各项的频率计数，并创建二维表，记录每个事务中各个项两两组合出现的支持度计数。例如，事务 T_2 包含项 I_1，I_2 和 I_4，这些项两两组合之后，可以形成 (I_1,I_2)，(I_1,I_4)，(I_2,I_4)，因此二维表中相应的位置均加 1。表 3 列出了本例中各个项的频率计数，表 4 列出了最终的二维表。

表 3 各个项的频率计数

Table 3 Frequency counting of items

项	计数
I_1	7
I_2	6
I_3	5
I_4	4
I_5	3
I_6	1
I_7	1

表 4 二维表

Table 4 Two-dimensional table

项	I_1	I_2	I_3	I_4	I_5	I_6	I_7
I_1	0	1	2	1	2	0	1
I_2	2	0	1	3	2	1	0
I_3	3	0	0	1	2	0	0
I_4	1	0	0	0	2	0	0
I_5	0	0	0	0	0	0	0
I_6	0	0	0	0	0	0	0
I_7	0	0	1	0	0	0	0

假设最小支持度为 2，则 I_6 和 I_7 是不频繁的，因此在原始数据库中的相关事务中删除这两个项，并按照各个项的频率计数对所有事物进行排序。得到的新的事务数据库如表 5 所列。

表 5 新的事务集合

Table 5 New transaction sets

事务	项	事务	项
T_1	I_1,I_3	T_6	I_2
T_2	I_1,I_2,I_4	T_7	I_2,I_4
T_3	I_1,I_2,I_3,I_5	T_8	I_2,I_4,I_5
T_4	I_1,I_3	T_9	I_1,I_2
T_5	I_1,I_3,I_4,I_5	T_{10}	I_1,I_3

遍历新的事务数据库，形成如图 1 所示的 FP tree，节点的数据结构为{Item；Count；ID；Parent；Link Node；Array；Leaf}。其中，Item 表示该节点所属项的名称，Count 表示该节点重复的次数；ID 表示节点在 Tree 中的序号；Parent 表示其父节点；Link Node 表示属于同一个项的链路中的下一个节点；Array 和 Leaf 的取值为 0 和 1，Array 为 1 表示采用数组查询的方式，Leaf 为 1 表示该节点为叶子节点，这两个参数用于查询频繁模式，各个节点的这两个属性的初始值均为 0。各个项在 FP tree 中的节点链路的首节点和链路长度如表 6 所列。

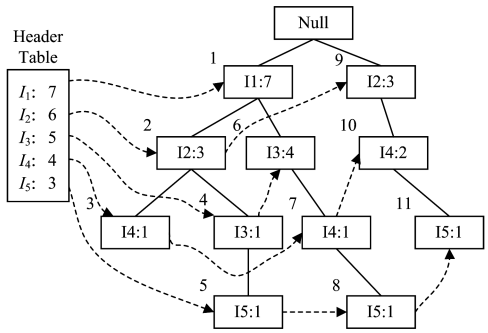

图 1 构建的 FP tree

Fig. 1 Constructed FP tree

表 6 各项链路的首节点和链路长度

Table 6 First node and link length of each link

Item	Link Hand	Link Size
I_1	1	1
I_2	2	2
I_3	4	2
I_4	3	3
I_5	5	3

在 FP tree 中，ID 为 3，5，8，11 的节点中 Leaf 被置为 1，其他节点的 Leaf 仍为 0。

3.2.2 生成频繁项集

生成频繁项集的过程可以分为 3 个阶段：第一阶段利用数组求出关于稀疏项的频繁二项集 set_1；第二阶段对于 FP tree 中的密集项，按从上到下的顺序进行遍历，得到频繁项集 set_2；第三阶段结合 set_2 按照 Apriori 算法的思路对 set_1 进行扩展，得到稀疏项的多项集 set_3。由表 6 可以看出，I_1 有一个节点，I_2 和 I_3 有两个节点，I_4 和 I_5 有 3 个节点。假设稀疏阈值为 3，则 I_4 和 I_5 为稀疏项，I_1，I_2 和 I_3 为密集项。将 tree 中 I_4 和 I_5 处节点的 Array 属性记为 1。

(1)稀疏项处理：对于 I_5，提取与 I_5 相关联的行和列，以给出项目 I_5 和其他项目之间的支持计数。$\{I_1,I_5\}$：2+0=2，$\{I_2,I_5\}$：2+0=2，$\{I_3,I_5\}$：2+0=2，$\{I_4,I_5\}$：2+0=2，由此 I_5 的支持度计数为$\{I_1:2,I_2:2,I_3:2,I_4:2\}$，构成的频繁二项集为$\{I_1,I_5\}$，$\{I_2,I_5\}$，$\{I_3,I_5\}$，$\{I_4,I_5\}$。

对于 I_4，$\{I_1,I_4\}$：1+1=2，$\{I_2,I_4\}$：3+0=3，$\{I_3,I_4\}$：1+0=1，$\{I_5,I_4\}$：0+2=2，由于最小支持度为 2，因此将 I_3 忽略，得到 I_4 的支持度为$\{I_1:2,I_2:3,I_5:2\}$，构成的频繁二项集为$\{I_1,I_4\}$，$\{I_2,I_4\}$，$\{I_4,I_5\}$。

(2)密集项处理：对于密集的 I_1，I_2，I_3 来说，我们采取从上到下遍历的方法，在查询一个新的节点时，先观察该节点的 Array 以及 Leaf 属性，可分为 4 种情况。

1)Array=0 且 Leaf=0：该节点不采用数组查询的方式，且不为叶子节点。记录该节点表示的信息，并继续遍历该分支下的后续节点。

2)Array=0 且 Leaf=1：该节点不采用数组查询的方式，且为叶子节点。记录该节点表示的信息，该分支扫描结束，下

一步扫描其他分支。

3）Array＝1 且 Leaf＝0：该节点采用数组查询的方式，且不为叶子节点。跳过该节点，继续扫遍该分支下的后续节点。

4）Array＝1 且 Leaf＝1：该节点采用数组查询的方式，且为叶子节点。跳过该节点，该分支扫描结束，下一步扫描其他分支。

我们从最上方的节点 1 开始扫描，由于节点 1 的 Link Node 为 Null，证明该项只有这 1 个节点，对该节点扫描得到：$\{I_1:7\}$，$\{I_1:3,I_2:3\}$，$\{I_1:4,I_3:4\}$，$\{I_1:1,I_2:1,I_3:1\}$，其中 $\{I_1:1,I_2:1,I_3:1\}$ 由于小于阈值而被删除。

接下来扫描节点 2，节点 2 的 Link Node 为 9，而节点 9 的 Link Node 为 Null，因此该项要扫描 2 个节点。经扫描得到：$\{I_2:3\}$，$\{I_2:1,I_3:1\}$，$\{I_2:3\}$，整理得到 $\{I_2:6\}$

然后扫描节点 3，节点 3 的 Link Node 为 6，而节点 6 的 Link Node 为 Null，因此该项要扫描 2 个节点。经扫描得到：$\{I_3:1\}$，$\{I_3:5\}$，整理得到 $\{I_3:5\}$。

经过上述两个过程，我们得到的频繁多项集为 $\{I_1,I_2\}$，$\{I_1,I_3\}$。

（3）组合处理：对于上述两个阶段生成的频繁项集：$\{I_1,I_5\}$，$\{I_2,I_5\}$，$\{I_3,I_5\}$，$\{I_4,I_5\}$，$\{I_1,I_4\}$，$\{I_2,I_4\}$，$\{I_1,I_2\}$，$\{I_1,I_3\}$，按照 Apriori 算法的思想，频繁三项集合的候选集为 $\{I_1,I_2,I_4\}$，$\{I_1,I_2,I_5\}$，$\{I_1,I_3,I_5\}$，$\{I_1,I_4,I_5\}$，对于数据库扫描得到 $\{I_1,I_3,I_5\}$ 为频繁项。

综上，所有的频繁项集为 $\{I_1\}$，$\{I_2\}$，$\{I_3\}$，$\{I_4\}$，$\{I_5\}$，$\{I_1,I_5\}$，$\{I_2,I_5\}$，$\{I_3,I_5\}$，$\{I_4,I_5\}$，$\{I_1,I_4\}$，$\{I_2,I_4\}$，$\{I_1,I_2\}$，$\{I_1,I_3\}$，$\{I_1,I_3,I_5\}$。

3.2.3　关联规则挖掘

在关联规则挖掘阶段，首先使用传统的方法计算规则的可信度，例如对于项集 $\{I_1,I_3,I_5\}$ 来说，支持度 $sup\{I_1\}=7$，$sup\{I_3\}=5$，$sup\{I_5\}=3$，$sup\{I_1,I_3\}=5$，$sup\{I_1,I_5\}=2$，$sup\{I_3,I_5\}=2$，$sup\{I_1,I_3,I_5\}=2$；$conf\{I_1->I_3,I_5\}=0.28$，$conf\{I_3->I_1,I_5\}=0.4$，$conf\{I_5->I_1,I_3\}=0.67$，$conf\{I_1,I_3->I_5\}=0.4$，$conf\{I_1,I_5->I_3\}=1$，$conf\{I_3,I_5->I_1\}=1$，假设可信度阈值为 0.8，那么规则 $I_1,I_5->I_3$ 和 $I_3,I_5->I_1$ 就可以认为是可信的。

在实际应用中还需要考虑告警之间属性的因果关系。如果不存在明显的因果关系，则以可信度计算结果为主；否则按照因果关系的规则进行处理。

3.2.4　改进算法的伪代码

改进算法的伪代码如算法 1－算法 3 所示。

算法 1　用于告警关联的改进 FP growth 算法

输入：告警序列 x(n)，最小支持度阈值 min_sup，最小可信度阈值 min_conf，稀疏项阈值 α

输出：告警间的关联规则

```
Use sliding window to divide x(n) into transaction set Dataset, and the total number of transactions is N
for every Trans in Dataset do
    count each item in Trans and construct a two-dimensional Table
end for
sort items in descending order according to their support
if item.support< min_sup then
        item is Infrequent, and delete it
end if
get the frequent item list L
create FP tree, and the root node is Root
for every Trans in Dataset do
        delete the infrequent items and sort the frequent item in the order of L
    Set the new transaction to be [t|T], where t is the first item, and T is the remaining item
    call Construct tree([t|T], Root)
end for
if node is leaf node then
        node.leaf<—1
end if
if item.link size>=α then
        item is sparse
        if node.item =item then
            node.array <—1
    end if
end if
if item is sparse then
    use Table to find support for other items, and get frequent item set Set1
else
    choose the most frequent item Item_f, call Mining FP(FP tree, min_sup, Item_f), and get frequent item set Set2
end if
use Set_1 and Set_2 to get Set_3 according to the idea of Aprior algorithm
for all frequent items do
    if conf( A—>B)>=min_conf then
        output A—>B
    end if
end for
```

其中，$conf(A->B)$ 表示规则 A 到 B 的可信度。

算法 2　用于构建 FP tree 的 *Construct tree* ($[t|T]$, *Parent*) 算法

输入：事务[t|T]，节点 Parent

输出：FP tree

```
If Parent have child nodes Child and Child.item= t.item then
        Child.count++
else
        create a new node newChild
        newChild.item =t.item
        newChild.count++
        newChild.parent=Parent
    Link newChild to other nodes with the same item
end if
if T is not null then
        t<— the first item in T, and delete it
```

```
12.      call insert_tree( [t|T],Child)
13. end if
```

算法 3 用于挖掘频繁项集的 Mining FP (FP tree,min_sup,$Item_f$)

输入:已经构造好的 FP tree,最小支持度 min_sup,频繁项 $Item_f$

输出:密集项的频繁项集 Set_2

```
1.  Take Item_f as the root node,and search child nodes Child
2.    if Child.array=0 and Child.leaf=0 then
3.          record the information represented by the node and find
            the subsequent nodes.
4.  elseif Child.array=0 and Child.leaf=1 then
5.          record the information represented by the node and find
            other branches
6.  elseif Child.array =1 and Child.leaf=0 then
7.      skip the node and find the subsequent nodes
8.  else
9.      skip the node and find other branches
10. end if
11. if Item_f.link node=null then
12.      combine sup(Item_f,Item_other)
13.      if sup(Item_f,Item_other)>=min_sup then
14.        get subSet from sup(Item_f,Item_other),and Set_2 = Set_2 ∪
           subSet
15.      end if
16.    let the next frequent item in the header be Item_next
17.      if Item_next.array=1 then
18.          call Mining FP (FP tree,min_sup,Item_next)
19.      end if
20.    if all frequent items in the header have been scanned then
21.          return Set_2
22. elseif Item_f.link node=SameitemNode then
23.  call Mining FP (FP tree,min_sup,SameitemNode)
24. end if
```

其中,sup($Item_f$,$Item_{other}$)表示 $Item_f$ 与 $Item_{other}$ 的支持度。

3.2.5 算法复杂度分析

设事务的数目为 N,项的数目为 M,其中密集项数目为 M'。原始算法在第一次遍历数据库时,时间复杂度为 $O(N)$;第二次遍历数据库时,完成对 FP tree 的构建,时间复杂度为 $O(N\log N)$;挖掘频繁模式时,对所有项的条件模式树进行递归查找,时间复杂度为 $O(M\log M)$。综上所述,算法的整体复杂度为 $O(N)+O(N\log N)+O(M\log M)=O(N\log N)+O(M\log M)$。

而改进算法在第一次遍历数据库时,构建了二维表,时间复杂度为 $O(N+1)=O(N)$;第二次遍历数据库时,完成对 FP tree 的构建,时间复杂度为 $O(N\log N)$;挖掘频繁模式时,对于稀疏项进行数组查询,时间复杂度为 $O(1)$,对于密集项进行树的遍历,时间复杂度为 $O(M'\log M')$,组合阶段遍历数据库,时间复杂度为 $O(N)$。综上,算法的整体复杂度为 $O(N)+O(N\log N)+O(1)+O(M'\log M')+O(N)=O(N\log N)+O(M'\log M')$。算法的整体复杂度没有较大提升,但是当数据集中的稀疏项较多时,改进算法的执行时间较短。

改进算法虽然构建了二维表,但是在挖掘频繁模式阶段,不需要构建过多的条件模式树,因此在内存消耗上较原始算法也有所优化。

4 实验验证

本文实验主要分为两部分。第一部分采用 DARPA2000 提供的攻击场景测试数据集 LLDOS1.0(Inside)作为告警信息来源,来验证改进算法关联告警数据的可行性。LLDOS1.0 中包含一个完整的 DDOS 攻击过程,可以分为 5 个阶段。

(1)执行 IPsweep。攻击者利用 IPsweep 脚本扫描 4 个网段:117.16.112.0/24,117.2.16.113.0/24,172.16.114.0/24 和 172.16.115.0/24。在扫描过程中,攻击者不断发送 ICMP Echo Request 数据包,并通过监听 ICMPEcho Reply 数据包来确定网络中在线的主机。

(2)端口探测。攻击者运行 Sadmind 探测程序,使用程序的 Ping 选项来查找运行着 Sadmind 服务的在线主机。

(3)获取访问权限。攻击者多次尝试对运行着 Sadmind 服务的主机发起 Sadmind 缓冲区溢出攻击。通过缓冲区溢出在目标主机上建立根用户,然后以根用户的身份通过 Telent 远程登录,继而获得目标主机的根访问权限。

(4)安装和启动 DDoS 软件。攻击者在目标主机上安装 DDOS 程序,并使用 rsh 命令启动程序,为发动攻击做最后的准备。

(5)发动 DDoS 攻击。攻击者 Telnet 到 DDOS 主机并发动对远程服务器的 DDOS 攻击。

攻击过程可以通过类似于图 2、图 3 的形式更加直观地展示出来。如图 2 所示,攻击的前 4 步主要是 IP 为 202.77.162.213 的攻击者与 IP 分别为 172.16.112.10,172.16.112.50,172.16.115.20 的目标主机之间的通信。如图 3 所示,第 5 步为攻击者操控目标主机,利用伪造 IP 对 IP 为 131.84.1.31 的受害者发起的 DOS 攻击。

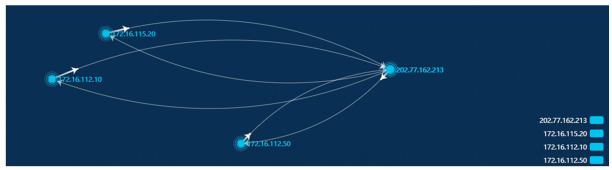

图 2 前 4 步中主要攻击的 IP 关系图

Fig. 2 IP diagrams of major attacks in the first four steps

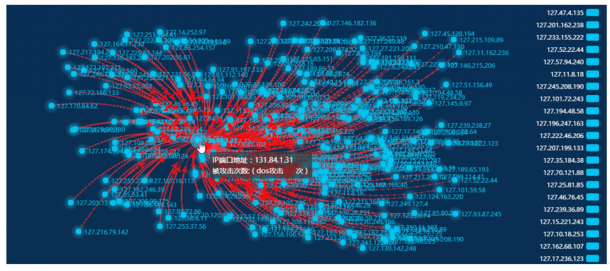

图 3 第 5 步中呈现的 DOS 攻击

Fig. 3 DOS attacks presented in step 5

第二部分将改进算法和原有算法进行对比来测试算法的性能。

实验硬件环境为 i5 2.40 GHz CPU 以及 8GB DDR3 内存,操作系统为 Windows10,实验程序使用 Java 编写,开发环境为 Visual Studio Microsoft 2015,使用的编译软件为 Myeclipse10,数据库管理软件为 NavicatforMySQL11.1。

4.1　算法的可行性分析

原始的 DARPA 数据集中告警数量较大、误告警较多，影响关联的效率，而且关联出的规则不具有通用性，因此，本文在进行关联之前使用 IGenClust 算法[17] 对原始告警进行聚类，将相似的告警聚集到相同簇。

将聚类结果保存在 MySQL 数据库中，以 172.16.115.20 为例图 4 给出了使用 Navicat 查看数据库中原始数据的各个阶段的存储情况。

源IP	目的IP	源端口	目的端口	类型	时间	簇
202.77.162.213	172.16.115.20			ICMP PING	03/07-12:06:16	4
172.16.115.20	202.77.162.213			ICMP Echo Reply	03/07-12:06:16	4

(a)阶段 1

202.77.162.213	172.16.115.20	54790	111	RPC portmap sadmind request UDP	03/07-12:09:51	5
202.77.162.213	172.16.115.20	54792	32773	RPC sadmind UDP PING	03/07-12:09:51	5

(b)阶段 2

202.77.162.213	172.16.115.20	60251	32773	RPC sadmind query with root credentials	03/07-12:19:42	7
202.77.162.213	172.16.115.20	60251	32773	RPC sadmind UDP NETMGT_PROC_SERVI	03/07-12:19:42	7

(c)阶段 3

172.16.115.20	202.77.162.213	1022	514	RSERVICES rsh root	03/07-12:24:12	1

(d)阶段 4

图 4　告警数据在数据库中的存储情况

Fig. 4　Alert in database

算法使用的数据为聚类后的数据，设定算法的支持度阈值为 50，可信度阈值为 0.8，稀疏阈值为 10，经过运行改进算法得到的关联规则为 4－＞5，5－＞7，7－＞1。图 5 给出了 Myeclipse 中算法结果的截图。

```
====关联规则====
5->7 : 0.8592689376144391
4->5 : 0.8002910511867727
7->1 : 0.9342417419097616
```

图 5　算法的关联结果

Fig. 5　Association results of algorithm

查找数据集中相应数据所属集群，发现大部分符合我们得到的关联规则，即本文算法可以应用于告警关联之中。当入侵检测系统检测到新的告警，并进行类别判定之后，可以使用之前求出的规则推测之后可能发生的攻击并加以防范。

4.2　算法的性能分析

实验的第二部分主要比较不同算法在运行时间以及内存消耗上的差别。

我们对 DARPA99 数据集进行误告警去除[18]，并从中选取 7 组数据作为实验数据。对于每一组数据，使用 3 种算法分别进行关联分析，记录算法的运行时间和内存消耗。这 7 组数据如表 7 所列。

表 7　实验数据

Table 7　Experimental data

数据集名称	数据总数
DATA1	2238
DATA2	3353
DATA3	4093
DATA4	4858
DATA5	5658
DATA6	10483
DATA7	18143

数据总数和运行时间的关系如图 6 所示，数据总数和内存消耗的关系如图 7 所示。由图可知，与 FP growth 算法以及本文的改进算法相比，Apriori 算法由于需要多次扫描数据库，因此消耗的时间较多；由于需要构建候选频繁项集，因此需要消耗大量的内存。而与 FP growth 算法相比，本文算法在时间和内存消耗上都有一定的优势。

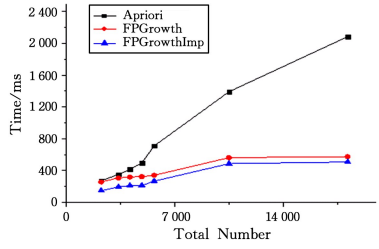

图 6　不同实验数据下算法执行时间的变化

Fig. 6　Different execution time under different experimental data

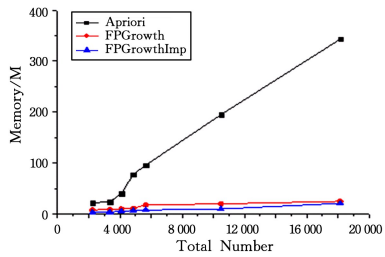

图 7　不同实验数据下算法内存消耗的变化

Fig. 7　Different memory consumption under different experimental data

结束语　针对现有基于 FP growth 的告警关联算法处理稀疏数据效率较低的问题，本文提出了一种改进算法来进行告警关联。算法综合 FP growth 算法和 Apriori 算法的思想进行频繁项挖掘，在较好地处理稀疏数据的同时，对原始算法的内存消耗和运行时间进行了一定的优化。本文将改进算法应用于 Darpa 数据集中，通过实验验证了算法的可行性和性能方面的提升。目前数据集的稀疏项阈值是建立在大量实验的基础上的，将来会对其进行进一步的研究，以找到一个可以普遍使用的阈值判定标准。

参 考 文 献

[1]　VALDES A，SKINNER K. Probabilistic Alert Correlation [C]// International Symposium on Recent Advances in Intrusion De-

tection. Springer-Verlag,2001:54-68.

[2] GAO H S,LI Y M. An ASON Alarm Correlation Method Based on Hierarchical Attribute Similarity Clustering[J]. Science Technology and Engineering,2015(6):210-214. (in Chinese)
高会生,李英敏. 一种基于分层属性相似度聚类的 ASON 告警关联分析方法[J]. 科学技术与工程,2015(6):210-214.

[3] ZHU L N,ZHANG Z C. Research on hierarchical alerts correlation based on causality[J]. Application Research of Computers,2016,33(3):848-850(in Chinese)
朱丽娜,张作昌. 基于因果关系的分层报警关联研究[J]. 计算机应用研究,2016,33(3):848-850.

[4] TEMPLETON S J,LEVITT K. A requires/provides model for computer attacks[C]// Proceedings of the 2000 workshop on New security paradigms. ACM,2001:31-38.

[5] MORIN B,MÉ L,DEBAR H,et al. A logic-based model to support alert correlation in intrusion detection[J]. Information Fusion,2009,10(4):285-299.

[6] JAJODIA S,NOEL S,KALAPA P,et al. Cauldron mission-centric cyber situational awareness with defense in depth[C]// MILCOM. 2011:1339-1344.

[7] YU D,FRINCKE D. Improving the quality of alerts and predicting intruder's next goal with Hidden Colored Petri-Net[J]. Computer Networks,2007,51(3):632-654.

[8] WANG S,TANG G M,KOU G,et al. Attack path prediction method based on causal knowledge net[J]. Journal on Communications,2016,37(10):188-198. (in Chinese)
王硕,汤光明,寇广,等. 基于因果知识网络的攻击路径预测方法[J]. 通信学报,2016,37(10):188-198.

[9] ZHANG J,LI X P,WANG H J,et al. Real-time alert correlation approach based on attack planning graph[J]. Journal of Computer Applications,2016,36(6):1538-1543. (in Chinese)
张靖,李小鹏,王衡军,等. 基于攻击规划图的实时报警关联方法[J]. 计算机应用,2016,36(6):1538-1543.

[10] NURBOL. Research on Anomaly Detection Based on Data Mining and Multi-stage Intrusion Alert Correlation[D]. Changchun:Jilin University,2010. (in Chinese)
努尔布力. 基于数据挖掘的异常检测和多步入侵警报关联方法研究[D]. 长春:吉林大学,2010.

[11] SONG S S. Study of Integrated alert correlation based on data mining and attack graphs[D]. Shanghai:Shanghai Jiao Tong University,2009(in Chinese)
宋珊珊. 基于数据挖掘及攻击图的告警综合关联研究[D]. 上海:上海交通大学,2009.

[12] MEI H B,GONG J,ZHANG M H. Research on discovering multi-step attack patterns based on clustering IDS alert sequences[J]. Journal on Communications,2011,32(5):63-69. (in Chinese)
梅海彬,龚俭,张明华. 基于警报序列聚类的多步攻击模式发现研究[J]. 通信学报,2011,32(5):63-69.

[13] LIU J. Research on Key Technologies of Intrusion Detection and Alert Association Based on Machine Learning[D]. Beijing:Beijing University of Posts and Telecommunications,2016. (in Chinese)
刘敬. 基于机器学习的入侵检测和告警关联关键技术研究[D]. 北京:北京邮电大学,2016.

[14] LI H C,WU X P. Network Intrusion Correlation Method with Differential Privacy Protection of Alerts Sequence[J]. Computer Engineering,2018,487(5):134-138. (in Chinese)
李洪成,吴晓平. 支持告警序列差分隐私保护的网络入侵关联方法[J]. 计算机工程,2018,487(5):134-138.

[15] AGRAWAL R,IMIELIŃSKI T,SWAMI A. Mining association rules between sets of items in large databases[C]// Acm Sigmod Record. ACM,1993,22(2):207-216.

[16] HAN J,PEI J,YIN Y. Mining frequent patterns without candidate generation[C]// ACM Sigmod Record. ACM,2000,29(2):1-12.

[17] LU X,DU X,WANG W. An Alert Aggregation Algorithm Based on K-means and Genetic Algorithm[C]// IOP Conference Series: Materials Science and Engineering. IOP Publishing,2018,435(1):012031.

[18] LU X,DU X,WANG W. Network IDS Duplicate Alarm Reduction Using Improved SNM Algorithm[C]// 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC). IEEE,2018:767-774.