

基于FP-Growth的入侵检测研究

孙志强

(长沙理工大学 计算机与通信工程学院, 湖南 长沙 410076)

摘要:数据挖掘可以利用各种分析工具从海量数据中发现模型和数据间的关系并做出预测。为了解决入侵检测在不降低精度的同时提高检测速度的问题,提高算法的效率,将FP-Growth算法应用于入侵检测系统中,提出对FP-Growth算法改进FP-tree的头表结构并引入关键属性来挖掘原始审计数据中的频繁模式,实验结果表明改进后的算法比传统的关联算法在入侵检测中的应用效果更好。可以看出,将FP-Growth算法应用于入侵检测中是可行的。

关键词:入侵检测;关联规则;FP-Growth算法;数据挖掘

中图分类号:TP393.08

文献标识码:A

文章编号:1673-629X(2006)12-0233-04

A Study of Intrusion Detection Based on Algorithm of FP-Growth

SUN Zhi-qiang

(College of Computer and Communication Engineering, Changsha University of
Science and Technology, Changsha 410076, China)

Abstract: Data mining can find the relation between pattern and data from the large number of data and the forecast will be made. In order to adapt to the real-time nature of the intrusion of testing requirements, and enhance the efficiency of algorithms, presented to the FP-Growth algorithms that improves FP-tree table structures and the introduction of key attributes to intrusion detection system. The experimental results showed improved algorithm has better results than traditional association algorithm in the application of intrusion detection.

According to the result, FP-Growth algorithm is useful to intrusion detection.

Key words: intrusion detection; association rules; FP-Growth algorithm; data mining

0 引言

随着计算机和通信技术的发展,网络已成为全球信息基础设施的主要组成部分,但随之而来的是不断暴露的网络安全问题。对目前绝大多数只采用防火墙进行安全防护的内部网络来说,仍有极大的安全风险存在。防火墙只是一种静态网络安全产品,只能进行一些被动的防护。因此,对网络安全的整体解决方案来说,需要一些积极的、主动的防御手段,其中最重要的就是入侵检测技术。

入侵检测系统可以定义为识别针对计算机或网络资源的恶意企图和行为并对此做出反应的系统。目前构造一个IDS是一项十分宏大的工程。很多情况下专业人员需要对已知的攻击方法和已知的系统漏洞进行分析、提取特征,然后用到误用入侵检测中去,或者是采用很多统计分析方法来做异常检测,但是可扩展性和适应性都不强。很多的IDS只能处理某些特定类的数据源,而且更新十分缓慢和昂贵。鉴于以上原因,将数据挖掘引入入侵检测系统中,它的自动发现数据特征的功能在入侵检测中取得了很好的效果。

数据挖掘理论的成熟为入侵检测提供了许多可行的算法。其中关联规则等算法在入侵检测系统中的应用尤其有用。关联规则是寻找在同一个事件中出现的不同项目的相关性。最有名、应用最广的关联规则是Apriori算法。Apriori算法是1993年Agrawal Imielinski和Swami首次提出的。在许多情况下,Apriori的候选产生-检查方法大幅度压缩了候选项集的大小,并导致很好的性能。然而,它有两种开销可能并非微不足道的。第一是它可能需要产生大量候选项集;第二是它可能需要重复地扫描数据库,通过模式匹配检查一个很大的候选集合。而对于入侵检测数据集来说,数据规模一般都很大,如果使用Apriori算法,效率有可能很低,而大量的研究都是对Apriori算法的改进,并没有从本质上解决导致效率较低的问题。HAN JIAWEI等人提出了一种不产生候选集的新型算法FP-Growth算法^[1]。对FP-tree(FP-tree)方法的性能研究表明,对于挖掘长的和短的频繁模式,它都是有效的和可伸缩的,并且比Apriori大约快一个数量级。因此,文中重点研究将FP-Growth应用于入侵检测系统中,并对算法提出改进,以适应入侵检测的研究。

收稿日期:2006-03-02

作者简介:孙志强(1979-),男,山东人,硕士研究生,研究方向为数据挖掘;导师:姚跃华,副教授,研究方向为数据挖掘。

1 FP-Growth 算法原理

FP-Growth算法采用了分而治之的策略:在经过了

第一次的扫描之后,把数据库中的频集压缩进一颗频繁模式树(FP-tree),同时依然保留其中的关联信息。随后再将 FP-tree 分化成一些条件库,每个库和一个长度为 1 的频集相关。然后再对这些条件库分别进行挖掘。当原始数据量很大的时候,也可以结合划分的方法,使得一个 FP-tree 可以放入主存中。

FP-tree 的定义如下:

● FP-tree 由 3 个部分组成:一个标记为“null”的根节点(root);子节点:作为根节点之子节点的项前缀子树(item-prefix-subtree)的集合;还有一个频繁项头表(frequent= item header table)。

● 每个项前缀子树的节点有 3 个域: item-name, count 和 node-link。item-name 记录了该节点所代表的项的名字。count 记录了所在路径代表的事务(transaction)中达到此节点的事务个数。node-link 指向下一个具有同样的 item-name 域的节点,要是该项下面没有子节点, item-name 域就为 null。

● 频繁项头表(frequent item header table)的每个表项(entry)由两个域组成: item-name; head of node-link。head of node-link 指向 FP-tree 中具有相同 item-name 的第一个节点。

1)按以下步骤构造 FP-树:

a. 扫描事务数据库 D 一次。收集频繁项的集合 F 和它们的支持度。对 F 按支持度降序排列,结果为频繁项表 L 。

b. 创建 FP-树的根节点,以“null”标记它。对于 D 中每个事务 Trans, 执行:

选择 Trans 中频繁项,并按 L 中次序排序。设排序后的频繁项表为 $[p | P]$, 其中 p 是第一个元素,而 P 是剩余元素的表。调用 insert-tree($[p | P], T$)。该过程执行情况如下。如果 T 有子女 N 的 $N.item_name = p.item_name$, 则 N 的计数增加 1; 否则创建一个新节点 N , 将其计数设置为 1, 链接到它的父节点 T , 并且通过节点链结构将其链接到具有相同 item-name 的节点。如果 P 非空, 递归地调用 insert-tree(P, N)。

2)FP-树的挖掘通过调用 FP-Growth(FP-tree, null) 实现。该过程实现如下:

Procedure FP-Growth(Tree, α)

(1)if Tree 含单个路径 p then

(2)for 路径 p 中节点的每个组合(记作 β)

(3)产生模式 $\beta \cup \alpha$, 其支持度 support = β 中节点的最小支持度;

(4)else for each a_i 在 Tree 的头部

(5)产生一个模式 $\beta = a_i \cup \alpha$ 其支持度 support = a_i . support;

(6)构造 β 的条件模式基, 然后构造 β 的条件 FP-树 Tree $_{\beta}$;

(7)if Tree $_{\beta} \neq \emptyset$ then

(8)调用 FP-Growth(Tree $_{\beta}$, β);

2 FP-Growth 算法在入侵检测中的应用

传统的数据挖掘技术都是为通用数据库设计的, 而要从真正海量审计数据中提取出人们所感兴趣的数据信息——准确的入侵检测的判定规则, 需要强调的一点就是特定的应用环境, 算法必须建立在特定应用的基础之上, 并且需要具有足够的先验知识^[2-6]。

设有如下事务数据库(见表 1), 假设最小支持度为 3。

表 1 事物数据库

TID	Items bought	(ordered)frequent items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

首先扫描一遍这个数据库, 计算每个项的计数值并按支持度降序排列, 将结果放入列表 L 中, 此时, $L = ((f:4), (c:4), (a:3), (b:3), (m:3), (p:3))$ 。执行算法的第二步, 创建一个标记为“null”的根节点。开始对数据库的第二遍扫描。对第一个事务的扫描将建立这棵树的第一个分支: $\langle (f:1), (c:1), (a:1), (m:1), (p:1) \rangle$ 。注意, 在这个事务中的频繁项已经被按照 L 中的顺序进行了排序。对于第二个事务来说, 它已经排序好的频繁项列表 $\langle f, c, a, b, m \rangle$ 同已经存在的路径 $\langle f, c, a, m, p \rangle$ 有共同的前缀 $\langle f, c, a \rangle$, 所以把这个前缀中的所有节点的 count 增加 1, 然后新节点 $(b:1)$ 被创建并且被作为节点 $(a:2)$ 的子节点, 随后, 新节点 $(m:1)$ 被创建并作为节点 $(b:1)$ 的子节点。对第三个事务, 因为它的频繁项列表只同以 f 为前缀的子树有一个共同节点 $\langle f \rangle$, 所以把这个节点的 count 增加 1, 并且创建新节点 $(b:1)$, 把它作为 $(f:3)$ 的子节点, 依次类推, 扫描完整个数据库。

为了方便对树的遍历, 一个频繁项头表(frequent item header table)被建立了, 头表表项的 node-link 指向树里面具有相同 item-name 的节点。具有相同 item-name 的节点通过 node-link 被连结在一起。如图 1 所示。

这里有一个问题, 即当每次在 FP-tree 中找到与头表有相同的 item-name 时, 必须回到头表中, 一个一个找出最后一个 node-link 的项后, 则将新找到的项再加入, 这种做法每次有一个新的项目出现时就必须从头表第一个开始找, 直到找出最后一个, 时间复杂度为 $O(n)$, 数据量大时效率不佳。

例如, 在上例中, 如果此时又要新增加一个含有 m 节点的分支, 则新增加的 m 节点要加入头表的 node-link 时, 必须先找到头表的 m 节点, 通过其 node-link 才能找到第一个 m 节点的位置, 再通过第一个 m 节点的 node-link, 找到下一个 m 节点所在的位置, 重复上述步骤一直到找出最后一个 m 节点后, 才将其加入。显然, 这样重复

多次,对算法的性能有很大影响。

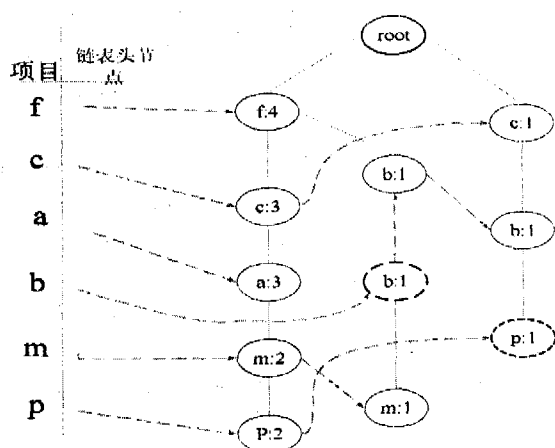


图 1 频繁项头表

要解决这个问题,可以在头表节点中加上一个 tail 域,用来记录每一项的最后一个值,这样当头表加入一个新值时就不需要循序地一个一个找到最后一个 node-link,而可以直接将新增加的项放到 tail 域中即可。

回到上面的例子,在头表中增加的 tail 域,记录着每一项的 node-link 所指的最后一个节点的位置。如果新增一分路,内容为 x, b, m, 则:

(1)在头表中没找到 x。

①将 node-link 指向新增加的 x 的位置;

②再将 x 节点所在位置加入其 tail 域中。

(2)在头表中找到 b 的 tail 域,已经有值存在,即 tail 非空。

①将 tail 域中所指节点的指针,先指向新增加的 b 节点所在位置;

②再将新增加的 b 节点加入 tail 域中。

(3)在头表中找到 m 的 tail 域,已经有值存在,即 tail 非空。

①将 tail 域中所指节点的指针,先指向新增加的 m 节点所在位置;

②再将新增加的 m 节点加入 tail 域中。

头表数据结构的变化见表 2。

表 2 头表数据结构

item-name	count head of node-link	tail
HTable:		
Typedef struct		
{		
int item-name;		
unsigned int count;		
struct node * node-link;		
struct node * tail-link;		
}HTable;		

改进后的 FP-Growth 算法能够省掉很多重复的过程,效率比原算法有了较大的提高。但是基本的数据挖掘算法并没有过多考虑专业领域的知识,这样就会产生许多对实际应用无关的规则。为了在入侵检测中更有效地利

用挖掘到的模式,可采用以下限制条件^[7]。

系统审计记录的各属性的重要性是不同的,即有的属性对描述数据起着关键的作用,它们是最基本的信息,而有的属性只是提供一些辅助信息。如网络连接记录中,源主机地址、目的主机地址、源端口、目的端口、时间这些属性对描述一次网络连接提供了重要的信息,而网络连接中的传送的字节数对一次网络连接并不能提供多大的信息。

例如,有一条规则:

源端口字节数 = 150 → 网络连接标记 = SF[0,4,0,2]

很明显主机源端口发送的字节数与网络的一次连接状态这两个属性之间是没有关系的,这就是一条误用的规则,它并不能说明什么问题,如果不去掉它,可能会起到一些误导的作用。既然网络连接中最基本的属性是服务类型(service),那么可以将它作为关键属性,生成的频繁关系中就必须包括这一属性字段。这样可在入侵检测中采用关键属性对所生成的频繁关系进行限制,就是要求关联规则算法所产生的模式中必须包含关键属性。设 R 是一条所产生的模式:

if R 的条目集中有关键属性 then

保留这条模式

else

删除这条模式,它不能提供有用的信息

end if

3 实验结果和分析

实验环境为 Windows XP, AMD3000+, 内存 512M。数据获取方式为在学院网络实验室内进行模拟,其中包括了使用泛洪攻击工具进行拒绝服务攻击获得的异常数据和使用密码猜测攻击工具获得的异常数据。部分数据见表 3。

表 3 部分实验数据

duration	src-host	dst-host	port	state	src-bytes	dst-bytes
1	202.118.85.14	202.118.85.31	2273	SF	3	4
1	202.118.85.31	202.118.85.14	43303	SF	4	2
1	202.118.85.31	172.20.100.119	3314	SF	4	2
1	202.118.85.14	202.118.85.31	2274	SF	2	4
1	172.20.100.119	202.118.85.31	2274	SF	2	4
1	202.118.85.31	172.20.100.119	3315	SF	4	2
1	202.118.85.31	202.118.85.14	43304	SF	4	2
...
6	172.20.100.39	218.199.102.210	8000	REJ	2	0
6	202.118.85.14	218.199.102.210	8000	REJ	2	0
1	202.118.85.14	202.118.85.31	2275	SF	2	4
0	172.20.100.119	202.118.85.31	2273	SF	2	4
1	172.20.100.119	202.118.85.31	2275	SF	2	4
1	202.118.85.14	202.118.85.31	2276	SF	2	4

设最小支持度为 50%, 利用 Apriori 算法和改进的 FP-Growth 算法分别对其进行挖掘, 经归纳整理得到表 4 所示结果。

表 4 实验结果

	Apriori 算法	FP-Growth 算法
生成的规则数	24	19
检测到的入侵事件	35	46
生成规则运行时间(s)	6.03	4.60

从上表可以看出 FP-Growth 算法在生成规则少于 Apriori 算法的情况下检测出了更多的入侵事件,有效地降低了规则的冗余度和系统的漏警率;同时生成规则所需要的时间也比 Apriori 算法要少。在处理实际入侵检测所产生的海量数据库和各种各样的入侵事件时,FP-Growth 算法的速度优势将更加明显地体现出来。从总体上来看,FP-Growth 算法的性能要优于 Apriori 算法,能够更好地为入侵检测系统服务。

4 结束语

文中研究了在入侵检测中应用 FP-Growth 算法。详细讨论了 FP-Growth 的实现方法,并对它进行了改进,实验证明改进的 FP-Growth 算法比传统的关联算法在入侵检测中的应用效果更好。但是我们注意到 FP-Growth 算法也没有检测出所有的入侵事件,因此进一步提高检测的精确度成了今后工作中应重点研究的内容。

(上接第 121 页)

了彼此间的关联关系。

下面是系统的整个工作流程,假设用系统管理员的角色登陆 login.jsp,系统将请求提交给 loginservlet,loginservlet 根据用户提交的信息调用业务逻辑层相关的 bean,并进行相应的帐户和密码验证,若通过则由 loginservlet 转发到 main.jsp,否则转发到 login.jsp。验证成功后管理员要添加人员,系统即调应 addperson bean,当管理员添加完信息提交后,系统将通过数据访问层在数据库的 person 表中添加相应的人员信息。

4 结 语

由上文可以看出,整个系统使用了 MVC 设计模式^[5],系统的主要优点是:

(1)在开发的过程中,只要定义好相应的接口规则,开发人员即可以专注于自己模块的开发,从而提高了系统的开发效率。

(2)Web 层与其它层相分离,可以根据当前的情况更新页面内容,增加系统需求的新功能,而其它的代码却无

(上接第 214 页)

子工业出版社,2003。

[2] TMS320C54X DSP CPU and Peripherals[M]. US:Texas Instruments. 1999.

[3] 赵红怡. DSP 技术与应用实例[M]. 北京:电子工业出版社,

参考文献:

- [1] HAN JIAWEI, PEI JIAN, YIN YIWEN, et al. Mining Frequent Patterns without Candidate Generation[C]//In: Proc Conf on the Management of data(SIGMOD'00, Dallas, TX). New York, NY, USA: ACM Press, 2000.
- [2] Agrawal R, Imielinski T, Swami A. Mining Association Rules between Sets of Items in Large Databases[C]//Proc Conf on Management of Data. New York, NY, USA: ACM Press, 1993:207-216.
- [3] 王新宇, 杜孝平, 谢昆青. FP-Growth 算法的实现方法研究[J]. 计算机工程与应用, 2004(9):174-176.
- [4] 吕 锋, 陈华胜. 关联算法的改进及其在审计数据挖掘中的应用[J]. 武汉理工大学学报:信息与管理工程版, 2004, 26(5):5-9.
- [5] 赵艳铎, 宋斌恒. 基于逆向 FP-树的频繁模式挖掘算法[J]. 计算机应用, 2005, 25(6):1385-1387.
- [6] 朱秋萍, 毛平平, 罗 俊. 基于关联规则的入侵检测系统[J]. 计算机工程与应用, 2004(26):160-162.
- [7] Bonchi F, Goethals B. FP-Bonsai: the ART of Growing and Pruning Small FP-trees: Proc 8th Pacific-asia Conference on Knowledge Discovery and Data Mining, PAKDD'04, Sydney, Australia[C]. Heidelberg, Germany: Springer-Verlag, 2004:155-160.

需太大的改变,从而保证了软件的可扩展性。

(3)业务逻辑层与其它层的分离,最大程度地提高了代码的重用性,只要对公共的组件进行优化,系统的整体性能即可得到明显的提高。

通过实践证明,该系统在 J2EE 框架中采用了 MVC 模式开发,极大地提高了系统的健壮性、可维护性和可扩展性。从而为许多基于 B/S 的系统提供了参考。

参考文献:

- [1] Bergsten H. JSP 设计[M]. 第 3 版. 北京:中国电力出版社, 2004.
- [2] 孙卫琴. 精通 Struts: 基于 MVC 的 JavaWeb 设计与开发[M]. 北京:电子工业出版社, 2005.
- [3] Roman E. 精通 EJB[M]. 第 2 版. 北京:电子工业出版社, 2002.
- [4] 易可可, 陈志刚. 基于 MVC 模式的 Web OA 系统设计与研究[J]. 计算机工程与应用, 2005(4):112-115.
- [5] 杨开英, 刘 树. MVC 设计模式在 J2EE 平台上的研究与实现[J]. 微机发展, 2004, 14(11):114-116.

2003.

[4] 汤华庚. TI54xxDSP 与 51 单片机的接口技术[J]. 单片机与嵌入式系统应用, 2004(1):27-30.

[5] 谢劲励. TMS320VC54X 接口技术应用研究[J]. 工业控制计算机, 2004, 17(3):17-19.