基于攻击意图的报警信息关联研究*)

柳亚明 许 峰 吕志军 黄 皓

(南京大学软件新技术国家重点实验室 南京 210093) (南京大学计算机科学与技术系 南京 210093)

摘 要 当前的入侵检测系统往往只提供给安全管理员大量低级的报警信息,分析这些报警信息极大地加重了安全管理员的负担,并且使得一系列相互关联的重要报警信息常常被淹没在大量不重要的报警信息中,因此迫切需要对于低级的报警信息进行进一步的关联,建立较为高层的攻击场景。本文提出了一个基于攻击意图的报警关联的模型,使用报警信息所对应的攻击行为的目的即攻击意图构建攻击场景。该模型先把入侵检测系统产生的报警信息转化为相应的攻击意图,再根据预先建立的攻击场景对这些攻击意图进行关联,从而实现了报警信息关联。

关键词 入侵检测,报警关联,攻击意图,攻击场景

The Research of Alert Correlation Based on Attack Intetion

LIU Ya-Ming XU Feng LU Zhi-Jun HUANG Hao

(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract Current intrusion detection systems only offer security administrators a great lot of independent, low-level alert, though there may be logical connections between them. It is a heavy burden for security administrators to analyze so many alerts, and a series of important alerts which are related to each other are often inundated with a large number of unimportant ones. Hence it is necessary to find a appropriate method to construct high-level attack scenarios from low-level attack alerts. This paper presents a model of alert correlation based on attack intention. The proposed approach constructs attack scenarios using attack intentions, i.e. purposes of attack actions corresponding to alerts. When alerts appear, the model turns them into corresponding attack intentions and correlate those attack intentions according to attack scenarios which have been established before.

Keywords Intrusion detection, Alert correlation, Attack intention, Attack scenario

1 引言

随着 Internet 的广泛应用,网络安全问题显得尤为突出。入侵检测系统(IDS)由于其内在的优点已经成为网络安全解决方案中的组成部分,其重要性已经得到广泛的认可。自从 Anderson 在文[1]中提出入侵检测的概念以来,出现了一批较为成熟的检测技术,如模式匹配、协议分析等等,在检测单个攻击行为方面有着很好的效果。然而对于系统威胁较大的攻击往往是由一系列相互联系的攻击行为完成的。现有的入侵检测系统其输出大都是大量低层的报警信息,不能表示报警所对应的单个攻击行为之间可能存在的联系,使得重要的攻击过程所引起的一系列相互关联的报警信息被混杂在大量的报警数据中,安全管理员需要花费大量的时间用于分析报警信息。而在 IDS 的实际应用中,安全管理员有时会因为不堪分析工作的重负放弃分析,使得 IDS 效用大大降低甚至形同虚设。这就迫切需要对于低级报警信息进行关联,建立高层的攻击场景(attack scenario),即从大量报警信息中分析出攻击者实施攻击的行为过程,使得安全管理员能够更好地了解网络中的攻击行为,采取更加有效的应对策略。

通过对于网络攻击行为的研究,我们获得了以下的认识:攻击者在进行攻击时,往往是通过一系列的攻击行为才能达到最终的目的,这一系列的隶属于同一攻击过程的攻击行为的组合称之为攻击场景。攻击场景中的每一个攻击行为都对应着一定的目的,在本文中称之为攻击意图,这样也可以把攻击场景看成由一系列攻击意图组成。由于一个攻击意图可以对应于多个攻击行为,因此攻击意图要比具体的攻击行为更关键、抽象,使用攻击意图构建攻击场景能够更加抽象地表示攻击过程。

在此认识基础上,本文提出了一个基于攻击意图的报警关联模型。在这个模型里,把 IDS 能够产生的报警信息种类称为报警信息类型,报警信息类型是 IDS 运行时产生具体的报警信息的抽象,它不包括报警时间、源目的地址等具体参数。与此相类似,使用攻击意图类型表示一组具体攻击意图的抽象。我们把所有的报警信息类型和对应的攻击意图类型表示为类,使用类之间的继承关系表示报警信息类型与攻击意图类型的对应关系,每个报警信息类型都是由继承攻击意图类型而来的,同时攻击意图类型之间也可能存在继承关系,以便将一个粗略的攻击意图类型能够进一步细分为更加具体的攻击意图类型,以适应攻击场景对于不同粒度的攻击意图的要求。具体的报警信息和攻击意图则表示为所属抽象类型

*)基金项目:国家"863"高技术研究发展计划项目"分布式网络监控与预警系统"(编号 2003AA142010)。**柳亚明** 硕士生,研究方向:入侵检测。**许 峰** 博士生,研究方向:计算机网络安全。**吕志军** 博士,研究方向:计算机网络安全。**黄 皓** 教授,博士生导师,研究方向:计算机网络安全、分布式计算。

的对象。攻击场景则由一组表示攻击意图的对象变量以及对象之间的时序关系和属性约束关系来表示。攻击场景库根据现有的攻击领域的专家知识预先建立。当IDS产生一个报警信息后，系统生成此对象的所有的直接或者间接的父类对象，即该报警信息所对应的攻击行为的所有可能的攻击意图，然后用这些父类对象去匹配攻击场景库中的攻击场景中对应的对象变量，在此基础上实现了报警的关联。

本文第2节是关于相关的工作的分析以及与本文工作的比较，第3节是报警信息与攻击意图的表示，第4节主要介绍攻击场景的描述，第5节是报警关联的实现，最后是全文的总结。

2 相关工作分析与比较

在入侵检测领域，报警信息关联有两种不同的定义：(1)综合多个报警信息或者其它数据源的信息以达到降低误报率的作用，提高IDS单个报警的置信度。(2)将底层的报警组织成为高层的攻击场景，以降低安全管理员负担，更有效地应对攻击事件。

本文中的报警信息关联采用第二种定义。在这个方面前人已经做了不少的工作。文[8,13,15]利用报警信息的属性之间的相似性进行关联。文[3]使用数据挖掘的方法从训练数据中发现攻击场景。文[6]定义了一个基于状态/事务的语言STATL表示预先制定的攻击场景。文[4]则先合并多个IDS报告，表示同一攻击行为的报警信息，然后根据事先制订的关于某一攻击产生后下一步会发生何种攻击的策略，进行关联。文[2,10,14]中使用报警信息所对应的攻击的前提条件和后果进行关联。文[11]将报警信息与所要保护的网络的拓扑结构联系起来进行关联，以关联那些表示对系统威胁比较大的攻击的报警信息。文[16]利用隐有色Petri网(Hidden Colored Petri Net)来表示攻击场景进行报警信息的关联，攻击场景的建立利用有关报警信息和所要保护的网络的知识以及训练数据完成。文[9]设计出一个分散的入侵事件关联模型，提出攻击描述语言ASL(Attack Specification Language)来描述分散在多台主机上的攻击事件，事件关联也由分散的多台主机协同完成。文[5]使用模糊系统(Fuzzy System)来描述攻击场景，实现报警信息关联。文[7]结合攻击场景中各个动作的时间和空间顺序通过描述用户的动态行为描述进行关联。文[12]采用主观逻辑(Subjective Logic)进行报警信息的关联。

上述相关的研究工作可以分为两种：

(1)根据实时产生的报警信息之间的联系动态进行报警信息的关联，例如文[2,8,10,13～15]等。

(2)使用实时产生的报警信息去匹配预先生成的攻击场景库中的攻击场景以实现报警信息的关联，例如文[3～7,9,11,12,16]等。

第一种方法的优点在于可以发现一些新的攻击场景，但是缺点也相当明显。其中使用报警信息之间的相似性进行关联，不需要相应的专家知识，但是只能发现一些较为简单的攻击过程。而使用报警信息对所对应的攻击的前提条件和后果进行关联，关联攻击的能力比前者强，但是不能关联某些因果关系不明显的攻击，在进行关联之前，必须清楚地了解各个报警所对应的攻击行为的前因后果的知识，而且关联时计算复杂性比较高。

第二种方法的优点在于能够准确匹配一些较为典型或者出现频率较高的攻击场景，但是其缺点在于只能匹配已知的或在训练数据中出现过的攻击场景，且需要大量的专家知识或者大量有代表性的训练数据才能建立攻击场景库，难以覆盖所有的攻击场景，更不能发现新的攻击场景。

综合上述方法的优缺点，我们决定采用专家知识预先建立攻击场景库的方式对报警信息进行关联。采用这种方法的最主要难点在于攻击场景库的建立，在攻击行为的种类数量较大的情况下，将这些攻击行为或者其对应的报警信息组织到攻击场景库中需要大量的人力物力，攻击场景的数量也会极大；同时由于IDS需要升级以应付层出不穷的新攻击，新的报警信息由此产生，这常常会导致大量新的攻击场景的产生，因此升级极为繁琐。

本文使用攻击意图而不是使用报警信息或其对应的攻击行为构造攻击场景库，这就比较有效地避免了这些缺点：

(1)攻击意图的数量要比报警信息本身或其对应的攻击行为少得多，建立攻击场景库的所需做的工作相对较少。

(2)当产生新的报警信息时，我们只需要把它归结到对应的攻击意图中去，攻击场景库中可以保持不变，只有新的攻击过程被发现时，才需要更新攻击场景库。

3 报警信息与攻击意图的表示

使用攻击意图对报警信息进行关联，首先需要使用合适的方法描述报警信息、攻击意图以及两者的联系。攻击意图具有层次性，一个粗略的攻击意图可以被细分为多个较细粒度的攻击意图，不同的攻击场景可能需要不同层次的攻击意图。同时一个报警信息所对应的攻击行为有时也可能具有一个以上的攻击意图，例如某一报警信息对应一个端口扫描攻击行为，该攻击行为既可以判断被扫描的端口是否开放，又可以根据返回的包的特征判断主机的操作系统，因此它可以有两个攻击意图。需要一个有效的方法描述报警信息与攻击场景以及攻击场景之间的复杂的关系。

本文借助于面向对象的概念，把报警信息类型、攻击意图类型都表示为类。每一个报警信息类型都是由继承相应的攻击意图类型而来的，而一个较粗略的攻击意图类型通过被继承也实现了进一步的细分。报警信息和攻击意图则表示为相应类的对象。

令一个类BaseAlert类为系统中所有类直接或间接的父类，这样所有的类就构成了一个类层次结构。图1是这个类层次结构的一部分的示意图。其中报警信息类型位于最低层，采用方框表示，攻击意图类型使用椭圆表示。

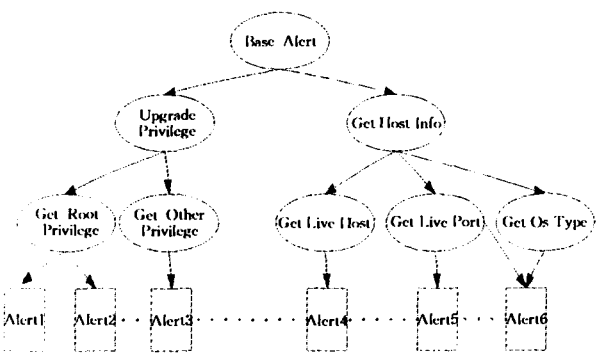

图1

采用如下的规则定义该类层次结构：

(1)类的定义语法使用BNF表示为：

〈类定义〉::=Class〈类名〉['{'〈属性定义〉'}']['('〈属性缺省值定义〉')'][:〈父类列表〉]
〈属性定义〉::=〈属性〉{,〈属性〉}
〈属性缺省值定义〉::=〈属性〉'='〈常量〉{,〈属性〉'='〈常量〉}
〈父类列表〉::=〈类名〉,{〈类名〉}

其中父类列表列出了该类继承的所有父类，属性定义定义了所有除了从父类继承的属性之外特有的属性。该定义省略了属性的数据结构实现细节。

例如 Class A{a1,a2}(a1=5):B,C 表示类 A 继承了类 B 类 C，除了继承了 B 和 C 的属性外，类 A 还有自己特有的属性 a1、a2，a1 的缺省值为 5。

(2)定义了一个类 BaseAlert，所有的其他类都是由它直接或者间接派生出来的，用来定义所有类都包含的关键属性，BaseAlert 的定义为：Class BaseAlert {AletrtID, SenorID, SourceInfo, DestInfo, DestPort, DetectTime, Description}。其中 AlertID 表示报警的编号，SenorID 表示发出报警的探测器的编号，SourceInfo 的源地址信息包括源 IP 地址和端口，DestInfo 表示目标地址信息包括目的 IP 和端口，DetectTime 表示发出报警的时间，Description 表示对于报警所对应攻击的描述。

(3)由类 BaseAlert 派生出表示某类攻击意图类型的子类，这些子类可以继续派生出表示细分的攻击意图类型的子类，也可以直接派生出子类用来表示报警信息类型。

(4)该类层次结构中所有的类都定义为虚基类，以保证在多重继承时，对于直接或间接继承了多次的某个父类的属性，子类中只保留一个副本。

(5)允许属性的重载，如果一个子类对于它的父类的一个属性进行了重载，那么父类的属性在该类的对象中被屏蔽。

采用这样的方法将报警信息与攻击意图联系起来，优点在于：1. 很好地表示了报警信息与攻击意图的映射关系以及攻击意图之间的层次关系。2. 新的报警信息和攻击意图很容易扩充到系统中去。

4 攻击场景的描述

攻击场景由多个攻击意图组成，这些攻击意图采用相应类型的对象变量来表示。在一个特定的攻击场景中，有的攻击意图必须先于另一个攻击意图执行，我们称这两个攻击意图间存在着时序先后关系，而另一些攻击意图之间可以以任意的时间顺序执行，我们称这些攻击意图之间存在时序并发关系。攻击意图之间除了有时序关系之外，攻击意图所对应的属性值之间还存在着一定的约束关系，例如某一攻击场景中两个攻击意图的源地址需要相同等，我们称这些攻击意图之间存在属性约束关系。所以在描述攻击场景时，除了需要描述其中包含的攻击意图之外，还需要描述攻击意图之间的时序关系和属性约束关系。

这里使用 BNF 形式化定义了一个攻击场景描述语言：

〈攻击场景〉::=Scenario〈ScenarioID〉'{'
Variable Definition:
〈变量类型声明〉
Schedule Restrict:
〈变量时序关系〉
Attribute Restrict:
〈变量属性约束条件〉
'}'
〈变量类型声明〉::={〈攻击意图类型〉{〈变量名〉,}〈变量名〉;}
〈变量时序关系〉::=变量名|〈变量时序关系〉'+'〈变量时序关系〉|〈变量时序关系〉'|'〈变量时序关系〉;
〈变量属性约束条件〉::={〈变量属性〉〈属性运算符〉〈变量属性〉;}
〈变量属性〉::=〈变量名〉.〈类属性〉
〈属性运算符〉::=>|<|=|>=|<=|<>|∈

在这个定义中变量表示构成攻击场景的各个攻击意图，类属性则指攻击意图类型的属性。变量时序关系中，使用 A+B 表示 A 和 B 的时序先后关系，使用 C|D 表示 C 和 D 的时序并发关系，运算采取向左结合的方法，其中时序先后关系的运算优先级大于时序并发关系的运算优先级。在属性运算符的定义中，由于变量属性可能是表示一组值的集合，因此除了定义了单个值的比较运算符之外，我们定义运算符 in 表示集合之间的包含关系。

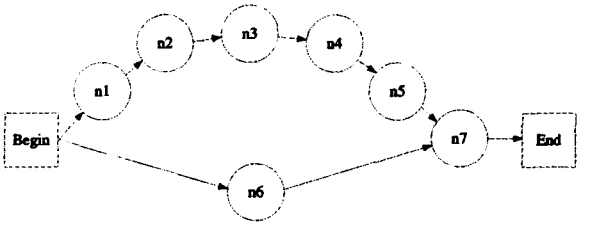

图 2

图 2 使用有向图描述了一个利用代理主机对目标主机进行 DoS 攻击的具体攻击场景，该图中仅仅表示了攻击意图之间的时序关系。图中圆型结点表示攻击意。n1～n5 表示在网上搜索一台活动主机，并入侵成功，利用其作为代理主机，以便在攻击时能够隐藏攻击者自身的地址，其中 n1 表示搜索活动主机，n2 表示搜索活动端口，n3 表示获得某端口守护程序的版本信息，n4 表示获得较高权限，n5 表示安装 DoS 工具；n6 表示搜索目标主机活动端口；n7 表示利用代理主机对目标主机进行 DoS 攻击。Begin 和 End 两个方框表示攻击场景的起始点和终结点，没有实际的意义，因此并不需要在攻击场景的定义中表示出来。则该攻击场景定义如下：

```
Scenario Example{
    Variable Definition:
        GetLiveHost n1;
        GetLivePort n2,n6;
        GetDaemonType n3;
        UpgradePrivilege n4;
        InstallDosTool n5;
        DenialofService n7;
    Schedule Restrict:
        (n1+n2+n3+n4+n5|n6)+n7;
    Attribute Restrict:
        n1. SourcInfo. IP=n2. SourceInfo. IP
        =n3. SourceInfo. IP=n4. SourceInfo. IP
        =n5. SourceInfo. IP=n6. SourcInfo. IP;
        n7. SourcInfo. IP=n2. DestInfo. IP
        =n3. DestInfo. IP =n4. DestInfo. IP
        = n5. DestInfo. IP∈n1. DestInfo. IP;
        n6. DestInfo. IP=n7. DestInfo. IP;
        n3. DestInfo. port∈n2. DestIno. port;
        n7. DestInfo. port∈n6. DestInfo. port
            }
```

5 报警关联的实现

系统把 IDS 产生的报警信息转变成相应的攻击意图去匹配攻击场景库中的攻击场景，并把匹配后的结果经过选择后提供给用户以实现报警信息的关联。

5.1 攻击场景的匹配

攻击场景的匹配过程分为预处理和实时匹配两部分，其中预处理在系统每次启动的时候完成，而实时匹配在每个新的报警信息产生的时候都会被执行。

5.1.1 预处理 实时匹配算法中需要每个报警信息的

所有攻击意图，也就需要找到每个报警信息类型的所有直接或间接的父类。如果在实时匹配的时候再从类定义中动态寻找某个类的父类，会降低实时匹配的速度，而同种报警多次出现会导致重复寻找父类的动作，所以需要预先找出来。另一方面，可能会有一些报警信息类型在当前的攻击场景库中并没有使用到，我们使用预处理将这些报警信息类型排除掉，可以提高匹配的效率。预处理主要处理流程如下：首先，找出攻击场景库中的所有攻击场景出现过的攻击意图类型，把它放到一个集合 S 中；然后对于每个报警类型，在类定义中可以找到表示其所有间接或者直接的所有父类；如果某个父类没有出现在 S 中，则将它从该报警类型的父类集合中删去，如果报警类型所有的父类都没有在 S 中出现，则舍弃这个报警类型，这样剩下的报警类型以及其对应的父类集合就是预处理的结果。

5.1.2 实时匹配 实时匹配主要处理流程如下：当一个新的报警信息送入关联模块，对于该报警信息对象所对应类的所有父类均生成一个对象，这些父类对象即为该报警信息所对应的攻击意图，然后使用每一个攻击意图去匹配攻击场景库中的攻击场景。

实时匹配的关键就在于攻击意图匹配攻击场景的效率。为了更好地描述所使用的攻击意图匹配算法，在这里引入前置集、后置集、并发集、攻击场景的实例集、变量赋值、完全匹配、匹配度等一系列的概念。在一个攻击场景中，某个攻击意图(对象变量)的前置集指所有必须发生在该攻击意图之前的攻击意图的集合，后置集指必须发生在该攻击意图之后的攻击意图的集合，并发集则是指所有与该攻击意图有时序并发关系的攻击意图的集合。由于攻击场景表示一种抽象的攻击过程，因此可以被实例化，一个攻击场景的实例集就是该攻击场景所有实例的集合。当一个实例刚刚建立时，所有的对象变量均没有被赋值，某一攻击意图被匹配到该实例中去时，相应的对象变量即用此攻击意图进行赋值，则变量已经被赋值意味着该变量已经被某一攻击意图匹配到了。某个攻击意图与一个实例上的某个对象变量完全匹配是指该变量尚未被赋值且与此攻击意图类型相同，该变量的前置集为空集或者已经全部被赋值到了，其后置集的所有变量都没有赋值，并且在该实例中的所有已经被赋值的变量的属性与该攻击意图的属性满足攻击场景中的属性约束条件。当某个攻击意图与一个实例中的某个对象变量类型相同又不能完全匹配的时候，用匹配度描述某实例与当前某个攻击意图的匹配程度：

匹配度＝(该变量前置集所有已经赋值且与此攻击意图之间满足属性约束和时序约束的变量个数)/(此攻击意图的前置集中变量的个数)。

算法 1：攻击意图匹配算法

定义 Type(P)表示攻击意图 P 的类型；$S_{instance}(C)$表示攻击场景 C 所对应的实例集；$S_{variable}(C,T)$表示一个攻击场景中类型为 T 的变量集合；函数 $PM(I,P,V)$表示在实例 I 中，攻击意图 P 和变量 V 能够完全匹配时返回值为 true，否则返回 false；$S_{before}(C,V)$、$S_{after}(C,V)$、$S_{current}(C,V)$分别表示攻击场景 C 中变量 V 的前置集、后置集和并发集；$S_{match}(I)$表示实例 I 中已经被赋值了的变量的集合；$S_{empty}(I,V)$表示在实例中 V 的前置集中尚未被赋值的变量的集合。$Num(S)$表示集合 S 的元素个数。

输入：攻击意图 P，攻击场景集合 $S_{scenario}$，每个攻击场景对应的实例集。

输出：每个攻击场景所对应的新的实例集。

(1)将 $S_{scenario}$ 中所有包含 Type(P)类型的变量的攻击场景加入 $S_{candidate}$ 中；

(2)令 C 为 $S_{candidate}$ 中的第一个攻击场景；$S_i=S_{instance}(C)$；

(3)令 V 为 $S_{variable}(C,\text{Type}(P))$中的第一个变量；

(4)如果 S_i 不为空，则 goto(6)；

(5)如果攻击场景 C 中 V 的前置集的变量数目小于或者等于 1，则生成一个新的实例，将 P 赋值给该实例中的变量 V；goto(15)；

(6)如果在 S_i 至少存在一个实例 I 满足 $PM(I,P,V)$为真，那么 goto(7)否则 goto(8)；

(7)对于 S_i 中的每一个实例，如果该实例中 P 与 V 能够完全匹配，那么在该实例中令 $P=V$；goto (15)；

(8)对于 S_i 中的所有实例，计算每一个实例中 P 与 V 的匹配度；

(9)令匹配度最大的实例为 I_{max}；建立一个攻击场景 C 的新的实例 I_{new}；在 I_{new} 中令 $V=P$；

对于 $S_{match}(I_{max})\wedge S_{before}(C,V)$中的每一个变量，如果其在 I_{max}中所对应的值与 P 满足属性约束和时间约束，则将该值赋予 I_{new} 中的同一个变量；令 $S'=S'-I_{max}$；

(10)如果 S'为 Φ，则 goto(12)；

(11)令 S'中匹配度最大的实例为 I'；对于 $S_{match}(I')\wedge S_{empty}(I_{new},V)$中的每一个变量，如果其在 I'中所对应的值与 I_{new} 中所有在此前已经被赋值的变量的值之间满足时间约束和属性约束关系，则将该值赋予 I_{new} 中的同一个变量；

如果 $S_{match}(I_{new})-V$ 等于 $S_{before}(C,V)$，则 goto (13)，否则令 $S'=S'-I'$然后 goto(10)；

(12)如果($Num(S_{before}(C,V))>5\&\&Num(S_{empty}(I_{new},V))/Num(S_{before}(C,V))<0.8 \parallel Num(S_{before}(C,V))<=5\&\&Num(S_{empty}(I_{new},V))>1$)，则从系统中删去 I_{new} 并 goto (15)；

(13)令集合 S_{tmp} 为空集；对于 S_i 中的每一个实例，如果该实例中 V 的并发集中已经被赋值的所有变量的值与 I_{new} 中已经赋值的所有变量的值之间满足时间约束关系和属性约束关系，则将该实例放入 S_{tmp} 中；

(14)计算 S_{tmp} 中的每个实例中 V 的并发集里已经被赋值的变量的数量，找出该数量取最大值的实例，将该实例中的并发集中已经被赋值的变量的值赋给 I_{new} 中相应的变量；

(15)如果 V 已经是 $S_{variable}(C,Type(P))$中最后一个变量，则 *goto*(16)，否则用 V 表示集合中的下一个变量并且 goto(4)；

(16)如果 C 已经是 $S_{candidate}$ 中的最后一个攻击场景，则 goto(17)，否则用 C 表示集合中的下一个攻击场景并令 $S_i=S_{instance}(C)$然后 goto(3)；

(17)算法结束。

5.2 报警关联的输出

在攻击场景的匹配过程中，系统中产生了大量攻击意图的实例，因此需要从大量实例中选择较为重要的实例输出给用户。

实例的输出策略：对于攻击场景中的每个变量按照其表示的攻击意图在这个攻击场景中的重要性预先定义一个权值，这样对于攻击场景的每个实例计算被赋值的变量的权值之和与所有变量权值之和的比值，输出其比值大于某一阈值的实例。如果攻击场景中的攻击意图的重要性难以量化，可

以简单地令所有变量的权值都为1。

在将实例输出时，把实例中已经被赋值的表示相应的攻击意图的变量重新转换成相应的报警信息，以方便用户查看，而那些没有被赋值的变量则表示IDS可能漏检的攻击行为或者攻击者可能采取的下一步攻击行为。

在没有被输出给用户的实例中有许多是错误匹配后的产物，这些实例的数量随着时间不断地增大。而系统中的实例的数量直接影响实时匹配的效率，因此我们对于每个攻击场景定义了一个生存时间。一个攻击场景的所有实例在系统中存在的时间都不超过该攻击场景所规定的生存时间，这样就使得实例的数量不至于过大，而一些错误匹配产生的实例也能及时地从系统中清除出去，从而提高了匹配的效率。

结论 本文提出一个基于攻击意图的报警关联模型，使用报警信息的攻击意图而非报警信息本身或者所对应的攻击行为来构建攻击场景，能够有效体现同一类攻击场景的本质特征，降低了攻击场景的数量，使得建立攻击场景库更为容易；同时当新的报警信息类型出现时，无需在攻击场景库中增加新的攻击场景。在该模型中借助于面向对象的概念把报警信息类型和攻击意图类型表示类，并把它们放入一个类层次结构中去，把具体的报警信息和攻击意图表示为对象，很好地表现了攻击意图的层次性以及攻击意图与报警信息之间的关系，新的报警信息类型或攻击意图类型产生时，只需要添加相应的类即可。在此基础上定义了一个语法较简单的攻击场景描述语言。在进行报警信息关联的时候采用的实时匹配算法其复杂度较低、匹配准确性高。相对于其他一些利用已知攻击场景库进行报警关联的方案，本文中的模型实现较为简单，系统的适应性较强。

参 考 文 献

1 Anderson J P. Computer security threat monitoring and surveillance:[Tech. rep]. JamesP. Anderson Co., Fort Washington, PA. 1980

2 Cuppens F, Miege A. Alert correlation in a cooperative intrusion detection frame-work. In: Proc. of the 2002 IEEE Symposium on Security and Privacy. of the 17th Annual Computer Security Applications Conf. 2002

3 Dain O, Cunningham R. Fusing a heterogeneous alert stream into scenarios. In: Proc. of the 2001 ACM Workshop on Data Mining for Security Applications, 2001. 1～13

4 Debar H, Wespi A. Aggregation and correlation of intrusion-detection alerts. In Recent Advances in Intrusion Detection. LNCS 2212, 2001. 85～103

5 Dickerson J E, Juslin J, Koukousoula O, Dickerson J A. Fuzzy intrusion detection. IFSA World Congress and 20th North American Fuzzy Information Processing Society (NAFIPS) Intl. Conf. Vancouver, British Columbia, Volume 3, July, 2001. 1506～1510

6 Eckmann S, Vigna G, Kemmerer R. STATL: An Attack Language for State-based Intrusion Detection. Journal of Computer Security, 2002, 10(1-2): 71～104

7 Jiang G, Cybenko G. Temporal and Spatial Distributed Event Correlation for Network Security. In: 2004 American Control Conf. (ACC), Boston, June 30 - July 2. 2004

8 Julisch K. Mining Alarm Clusters to Improve Alarm Handling Efficiency. In: Proc. of the 17th Annual Computer Security Applications Conf., New Orleans, LA, Dec. 2001

9 Kruegel C, Toth T, Kerer C. Decentralized Event Correlation for Intrusion Detection. In: Proc. of the Intl. Conf. on Information Security and Cryptology (ICISC), Lecture Notes in Computer Science, Springer Verlag, Korea, Dec. 2001

10 Ning P, Cui Y, Reeves D S. Constructing attack scenarios through correlation of intrusion alerts. In: Proc. of the 9th ACM Conf. on Computer and Communications Security, Washington, D. C., Nov. 2002. 245～254

11 Porras P, Fong M, Valdes A. A mission-impact-based approach to INFOSEC alarm correlation. In: Proc. of the 5th Intl. Symposium on Recent Advances in Intrusion Detection (RAID 2002). 2002. 95～114

12 Svensson H, J Ø sang A. Correlation of Intrusion Alarms with Subjective Logic. In: Proc. of the Sixth Nordic Workshop on Secure IT Systems (NordSec2001), Copenhagen, Denmark, 1-2 Nov. 2001

13 Staniford S, Hoagland J, McAlerney J. Practical automated detection of stealthy portscans. Journal of Computer Security, 2002, 10(1-2): 105～136

14 Templeton S, Levitt K. A requires/provides model for computer attacks. In: Proc. of New Security Paradigms Workshop, ACM Press, 2000. 31～38

15 Valdes A, Skinner K. Probabilistic alert correlation. In: Proc. of the 4thIntl. Symposium on Recent Advances in Intrusion Detection (RAID 2001). 2001. 54～68

16 Yu D, Frincke D. A Novel Framework for Alert Correlation and Understanding (Springer-Verlag), in Springer's LNCS series, vol 3089, In: Intl. Conf. on Applied Cryptography and Network Security (ACNS), 2004. 452～466

(上接第25页)

条件，来研究Petri网单链子网合成的其它性质(如有界性、公平性等)的保持问题。

参 考 文 献

1 Hcare C A R. Communicating sequential process [J]. Communications of the ACM, 1978, 21(8): 666～677

2 Taubner D. Finite representation of CCS and TCSP programs by automata and Petri nets [J]. LNCS, 1990, 369

3 Milner R. A calculus of communicating systems [J]. LNCS, 1980, 92

4 Goltz U. On representing CCS programs by finite Petri nets [J]. In Mathematical Foundations of Computer Science. LNCS, 1988, 324

5 Casavant T L, J G. A communicating finite automata approach to modeling distributed computation and its application to distributed decision～making [J]. IEEE Trans, Computers, 1990, 39(5): 542～558

6 Bednarczyk M A, Bernardinello L, et al. Modular system development with pullbacks [J]. In: Proc. the 24th Intl. Conf. Application and Theory of Petri Nets. Eindhoven, The Netherlands, 2003. 140～160

7 Morin R. Decompositions of asynchronous systems [J]. In: Proc. CONCUR'98, LNCS 1466, Springer, 1998. 549～564

8 Mäkelä M. Model checking safety properties in modular high-level nets [J]. In: Proc. the 24th Intl. Conf. on Application and Theory of Petri Nets. Eindhoven, The Netherlands, 2003. 201～219.

9 van Hee K, Sidorova N, et al. Soundness and separability of workflow nets in the stepwise refinement [J]. In: Proc. the 24th Intl. Conf. on Application and Theory of Petri Nets. Eindhoven, The Netherlands, 2003. 337～356

10 Bernardinello L. Synthesis of net systems [J]. In: Proc. Application and Theory of Petri nets, LNCS 691, Springer-Verlag, 1993. 89～105

11 Bernardinello l, Ferigato C, et al. Towards modular synthesis of EN systems [J]. In: Caillaud B, et al eds. Synthesis and Control of Discrete Event Systems, Kluwer Academic Publishers, 2002. 103～113

12 Souissi Y. On liveness preservation by composition of nets via a set of places [J]. In: Rozenberg G, ed. LNCS 483, New York: Springer-Verlag, 1990. 457～470

13 Jiang Changjun, Wu Zehui. Net Operations [J]. Journal of Computer Science and Technology, 1992, 7(4): 333～344

14 Kovalyov A. An O(|S| X |T|)-algorithm to verify liveness and boundedness in extended free choice nets [J]. In: Proc. the 10th IEEE Intl. Symposium on Intelligent Control, Monterey, California, USA, 1995. 597～601

15 Esparza J, Silva M. A polynomial-time algorithm to decide liveness of bounded free choice nets [J]. Theoretical Computer Science, 1992, 102: 185～205

16 Kovalyov A, Mcleod R D. Strongly connected free-choice systems having non dead home markings are live and bounded [J]. Petri net Newsletter, Springer, 1998, 54: 16～18

17 Zhen Q, Lu W M. On liveness and safeness of asymmetric choice nets [J]. Journal of Software, 2000, 11(5): 590～605