维普资讯 http://www.cqvip.com

隐蔽通道发现技术综述*)

夏 耐 林志强 茅 兵 谢 立
(南京大学软件新技术国家重点实验室 南京 210093)

摘 要 本文首先解释了隐蔽通道的概念,介绍了隐蔽通道的分类。然后花主要精力,结合实例概述了当代国际上已经使用过的隐蔽通道的方法。并且对目前已有的这些方法从不同的方面给出了对比性的评价。在文章的讨论部分,作者基于目前的形势以及传统发现方法的限制给出了针对隐蔽通道发现方法建设性的见解和展望。

关键词 隐蔽通道,安全系统,机密性,安全保护,信息流安全

A Survey on Covert Channel Identification

XIA Nai LIN Zhi-Qiang MAO Bing XIE Li
(National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

Abstract This paper begins with the introduction of the notion or the definition of covert channel. After explaining the classification of the covert channels, it spent most of its words on the covert channel identification methods currently available in the CCA (covert channel analysis) community. Then it compares the methods with each other in different aspects. In the discussion session, the paper analyzes the limitations of these methods and gives out some constructive suggestions.

Keywords Covert channel, Security system, Confidentiality, Security protection, Information flow security

1 前言

当一个安全系统即将建立完成的时候,对这个系统进行隐蔽通道分析是必不可少的。因为系统中存在的隐蔽通道能够使得攻击者绕过所有的安全保护机制来妨害系统的机密性(confidentiality)。隐蔽通道的概念首先由 Lampson[1] 提出,从那以后,有不同的定义试图从不同的角度来描述隐蔽通道的本质和特性。按照 TCSEC[2] 中一个比较通泛的定义来说,隐蔽通道就是"能让一个进程以违反系统安全策略的方式传递消息的信息通道"。具体来说,在一个非自主访问控制的安全系统,如多级安全系统(multi level security)中,假设这个系统的安全模型定义主体(subject)S_h 能够合法地传递信息给主体 S_l 当且仅当前者的级别大于等于后者($h \geqslant l$),那么任何的能让传递信息给的信息通道都称作隐蔽通道。在所有的安全系统中都有许多设计是提供给使用者的信息传送方式,比如 I/O 设备、文件、IPC 等等,它们都是被看作是"显式"通道,是因为那些机制里面用于存放信息的单元都被看作数据客体(object),而一个隐蔽通道用以存放信息的单元一般都不被看作数据客体,比如文件锁、设备忙标志、时间的流逝等等。这也是为什么被称作隐蔽通道的原因。隐蔽通道的存在使得系统中藏有特洛伊木马的两个进程能够肆意地进行信息交流。20 多年来,人们为了解决这个安全问题,对隐蔽通道的发现、评价和处理方法都做了广泛和深入的研究。本文描述的是隐蔽通道的发现技术,并试图对目前已经有的发现方法做一定的评价和比较,同时在此基础上提出作者自己关于这个问题的看法和展望。

本文以下分成 4 节,首先介绍隐蔽通道的分类以及在系统中实际存在的情形,其次列举目前已有的针对不同隐蔽通道的发现方法以及这些方法的比较与评价,再次站在隐蔽通道发现的技术的前沿,讨论一些可能的发展方向以及作者目前已经开始的工作。最后给出一个简单的结论。

2 知识背景

前面介绍了隐蔽通道的定义,从本质上来说,隐蔽通道的存在反映了一个现实的复杂安全系统的实现与该系统设计时候所遵循的安全模型所定义的信息流安全[3](information flow security)的不可避免的差异。由于现实系统的复杂性以及现实系统设计时候更多的性能方面以及兼容性方面的考虑因素,使得严格遵守某个信息流安全的机制是不现实的。这使得目前所有的安全系统的实现的安全机制都只是某个安全模型在一定程度和粒度上的近似,进而隐蔽通道存在于当前所有的安全系统中。事实已经不止一次的告诉人们,没有一个绝对安全的系统,和其他所有的安全问题一样,人们目前所做的工作也都是为了缓解或者限制这样的安全隐患,而不是企图将其从现实的安全系统中彻底地消除。目前,某些国外的安全权威机构已经定义了一些关于怎样处理安全系统中隐蔽通道的标准,如美国军方针对多级别安全系统(MLS)的标准 TCSEC 中对 B2 以上级别的系统都规定了详细的隐蔽通道分析的要求。

对于隐蔽通道分析的第一步,就是本文所描述的隐蔽通道的发现。在我们介绍隐蔽通道的发现方法之前,我们首先较深入了解一下隐蔽通道的一些特性。

2.1 隐蔽通道的分类

在现实的系统实践中,可以按照不同的特性对隐蔽通道

*)本文研究得到国家高技术研究发展计划 863 课题(2001AA144010)资助。夏 耐 博士研究生,研究系统安全与软件安全;茅 兵 教授,研究人机交互、分布计算和并行处理系统安全;谢 立 博导,教授,研究分布式计算和系统安全。

维普资讯 http://www.cqvip.com

进行分类：按照利用的存放信息的方式，可以分为存储隐蔽通道（storage covert channel）和时间隐蔽通道（timing covert channel）；按照信息通道的特性，可以分为有噪声隐蔽通道（noisy covert channel）和无噪声隐蔽通道（noiseless covert channel）；按照共享变量的利用方式，可以分为聚集的隐蔽通道（aggregated covert channel）和非聚集隐蔽通道（non-aggregated covert channel）。其中和隐蔽通道发现密切相关的是第一种分类方式，也就是存储/时间隐蔽通道。

存储隐蔽通道与时间隐蔽通道的区分最初开始于文[4,5]的一些著作，而到后来的文[6]通过分析磁盘臂隐蔽通道，指出在本质上，这两种隐蔽的通道是没有区别的，或者说这两个种类的隐蔽通道之前的界限是模糊的。但是人们习惯上仍然对于这两种通道进行区别，一般地，定义一个存储隐蔽通道是这样的一种隐蔽通道，当对它的利用"包括一个进程对一个存储变量的直接的或者间接的修改，以及另一个进程对这些存储变量的直接的或者间接的读取"[2]。而对时间隐蔽通道则是指的它的利用"包括一个进程调制它自己关于一个系统资源的使用规律（比如CPU使用时间），这样的操作使得第二个进程对于某个操作响应的时间发生变化"。

Kemmerer在文[7]中定义了两种隐蔽通道存在的条件：

存储隐蔽通道：

- 发送和接收进程必须对某个共享资源的同一个属性有访问权
- 一定有某种方式使得发送进程能够迫使这个属性去改变
- 一定有某种方式使得接收进程能够探测到这个属性的改变
- 一定存在某种机制使得发送/接收双方能够同步发送事件，这个可以是另外一个带宽较小的通道

时间隐蔽通道：

- 发送和接收进程必须对某个共享资源的同一个属性有访问权
- 发送和接收进程必须有一个统一的时间参考，比如一个实时钟
- 发送进程必须能够调制接收者的响应时间来表示一个属性的改变
- 一定存在某种机制使得发送/接收双方能够同步发送事件

可以看出隐蔽通道存在的一个共同的前提条件是资源的共享，这个是一个实际有用的系统中所必须具有的一个特性，所以也就成为隐蔽通道存在的一个基础。而对于两种隐蔽通道的区别之处也就是对时钟的参考，很多时候，由于计算机中的时间概念可以被看作是一个具有相对位置关系的事件序列[6]，进而时间的获取也可以来源于对存储变量的改变的观察，所以如同上面指出的，它们之间的界限是模糊的。换句话说，一个通道时是存储隐蔽通道还是时间隐蔽通道取决于人们对时间概念的看待。事实上，如同文[6]中论证的，系统中任意一个隐蔽通道都可以分解成两个不同的可以互相区别的事件序列的组合。但是一般系统当中，有些机制是被固定看作时间的（如时钟发生器，某个外在的有固定频率的I/O事件），而另一些则更倾向于看作存储变量（如文件锁，磁盘臂的位置等等），这样也就有了很多一般意义上的存储隐蔽通道和时间隐蔽通道。

2.2 隐蔽通道实例

为了能够更好地阐明隐蔽通道的概念和它们的分类，以下列举一些经典实例。由于具体的隐蔽通道的存在直接依赖于TCB，以下的实例并不保证在所有的系统中都实际存在。同时为了简单起见，给定前提是：多级别安全系统，系统中只有两个进程发送者S和接收者R，S的安全级别高于R，系统采用简单的时间片轮转调度策略。

2.2.1 存储隐蔽通道实例

（1）共享资源耗尽通道

假设系统中一个有限的资源，如系统的虚拟内存M，被S和R共享使用，TCB原语alloc(size)用于用户向系统请求分配大小为size的内存，如果当前空闲虚拟内存量empty(M)≥size那么分配成功并返回内存指针，否则返回为null。一种情形，在S运行的时候，它通过频繁申请大小为size′的小块内存直到empty(M)<size′，然后S等待推出运行。另一种情形，S释放所有它占用的内存块，然后推出执行。等到R进入运行，R进行一次alloc(size′)的操作，如果这个操作的返回值为null，那么，R就知道，上一次S的运行是采用的情形一，否则就是采用的情形二。如果事先S和R定义第一种情形代表"0"，第二种情形代表"1"，那么每次S和R的运行就完成了一位的信息传递。

（2）客体存在通道

一个由S产生或者消除的客体，它的存在与否能通过某种方式被R所感知。典型的例子如文件存在通道。在MAC的访问控制机制中，假设一个目录与R同级别，那么如果该目录为空就能够被R所删除，否则删除失败。S通过在该目录中创建文件使得目录非空，或者删除文件使得目录为空。这样，S的不同操作就能被R感知，进而每次S和R的执行可以用来传送一位的信息。

（3）时间隐蔽通道实例

CPU调度通道是一个非常典型的时间隐蔽通道。在一个时间片轮转调度的分时安全系统中，S要么在它执行的时间片内将自己挂起（主动让出CPU），要么使用整个它所赋予的时间片。R只需要参考一个系统时钟来判断自己两次执行之间等待的时间的长短就可以知道上次S执行的情况。S和R可以约定好这两种情况分别代表0和1。

3 隐蔽通道发现方法

3.1 已有方法

20世纪80年代中期，由于TCSEC对B2级别以上安全系统隐蔽通道分析的要求，隐蔽通道分析技术的研究也随之升温。这个时期出现了很多从不同方面对隐蔽通道分析的方法。

3.1.1 非相干性分析

非相关性模型（non-interference）是安全领域的一个经典模型，它首先由Goguen和Meseguer在文[8]中提出。非相关性模型从本质上形式化了这样一个想法：在一个安全系统中，一个用户不能意识到任何不由它所支配的用户（对于多级安全系统，就是安全级别比它高的用户）的任何操作。它将一个安全系统的TCB抽象成一个状态机（state machine），假设A和B是这样一个抽象TCB的两个用户，如果w是对这个状态机从初始状态的输入，并且这个输入的最后一个操作是来自B的，那么定义B(w)为B由最后一个输入获得的输出。同时w/A定义为将w中来自A的输入的删除（裁减）以后得到的子串。这样，定义A非相干于B，当且仅当对于所有可能的以B的输入结尾的输入串w，B(w)= B(w/A)。如果一个多级系统中不存在隐蔽通道，那么任何一个用户都应该非相

维普资讯 http://www.cqvip.com

干于级别比它低的用户。

在实际的系统的分析中，这种方法作用于系统的抽象形式化顶层说明(FTLS)，并且由于分析从初始状态开始的整个输入串非常繁琐，Goguen[9]通过一个简化了的定理来体现非相干性，这就是所谓的"展开定理"(unwinding theorem)。这个定理使得实际检查一个系统的非相干性成为可能。这个定理指出，系统的状态可以按照某个用户 B 分成等价类，一个 B 等价类内的所有成员满足：①对于任何 B 的输入，B 所获得的输出是相同的。②对于任何输入的下一个状态也是 B 等价的。多级别系统中的安全级别可以被应用于作为等价状态的的标签。

非相干性分析被应用于 Secure Ada Target (SAT)[10]的隐蔽通道分析当中。SAT 的目标是 A1 级别的安全系统，提供了用 Gypsy 语言描述的 DTLS。分析人员试图证明系统符合"展开定理"，证明失败地方需要重新进行信息流的分析来确定是否存在隐蔽通道。

3.1.2 共享资源矩阵(Shared Resource Matrix, SRM)

共享资源矩阵的方法首先由 Kemmerer 在文[7]中提出，尽管有一定的局限性，它仍然是当前最实用的隐蔽通道发现方法。该方法一般包含以下的步骤：

①根据具体的系统描述(形式化的、非形式化的、源码)分析所有的 TCB 原语，构造一个基本的共享资源矩阵。矩阵的绘制方法是：用户可见的 TCB 原语标识矩阵的行，用户可见或者可改变的 TCB 的变量标识矩阵的列，以 R 或者 M 作为矩阵的元素分别来代表一个 TCB 变量是否能被一个 TCB 原语所读取或者改变。

②构造这个基本共享资源矩阵的传递闭包。构造的方法按照这样的原则：假设一个 TCB 原语 P1 对 TCB 变量 V 的有读取操作，P2 对 V 有改变的操作。那么，P1 对所有 P2 用以改变 V 所读取的变量具有间接读的操作。对于共享资源矩阵来说就是，对于任何一个 R 值的成员检查是否在同一行有 M 成员，如果有的话，将 M 所在列的所有 R 成员添加到原先"R"所在列的对应行上面去。重复这样一个过程，直到不再有新的 R 元素被添加。

③分析上一步所得到的矩阵。如果某个进程能够利用 TCB 原语读一个变量，而另一个的进程能够通过 TCB 原语写同一个变量的话，那么就存在一个的信息通道。对于这个矩阵的分析结果中可能存在的隐蔽通道在一些情况下是不能成为隐蔽通道的：(1)发送方和接收方是同一个进程；(2)这个通道不能用来传送有用的信息；(3)这个通道本身按照安全策略来说是合法的。剩下的通道被称为是潜在隐蔽通道。

④对于上一步所发现的潜在隐蔽通道，分析这个通道所涉及到的矩阵的元素，进而试图构造这个隐蔽通道实际使用的情形。在有些情况下，一个通道的实际利用情形并不存在，所以它也就不能被实际利用，进而成为一个真正的隐蔽通道。

为了更好地展示整个分析的过程，下面举一个象征性的例子。

例 1 用 SRM 发现隐蔽通道

假设一个系统中有以下的 TCB 原语：

sys_unlink: 删除一个文件名 缩写 S_UL

sys open read: 读方式打开文件 缩写 S_OR

sys open write: 写方式打开文件 缩写 S_OW

sys_creat: 创建文件 缩写 S_CR

以及以下的 TCB 变量：

file. writers 文件的写者 缩写 f. w

file. readers 文件的读者 缩写 f. r

free_blocks 空闲磁盘块 缩写 f_b

容易构造基本的 SRM 如表 1，然后构造它的传递闭包如表 2。

表 1 基本 TCB 共享资源

变量	TCB 函数接口			
	S_OR	S_OW	S_CR	S_UL
F_b			RW	W
f. r	W			R
f. w		W	W	R

表 2 共享资源闭包

变量	TCB 函数接口			
	S_OR	S_OW	S_CR	S_UL
F_b			RW	WR
f. r	W		R	R
f. w		W	RW	R

容易发现里面存在的两个潜在的隐蔽通道：(1)资源耗尽通道，由 TCB 接口函数 S_CR 和 S_UL 以及 TCB 变量 f_b 组成，一个高级别的进程可以通过 S_CR 和 S_UL 使得磁盘空闲快为 0 或者非 0，而这可以一个低级别的进程通过 S_CR 探测到，进而这个通道可以成为一个实际被利用的隐蔽通道。(2)文件读者数量通道，这个通道可以由 S_OW，S_UL/S_CR 所组成。发送者通过 S_OR 来读方式打开一个低级别的文件，会导致低级别的接受者 S_UL 的操作的失败。同样这也是一个可以被实际利用的隐蔽通道。(3)文件写者通道，这个通道可以由 S_OW，S_UL/S_CR 所组成。但是这个通道并不是一个真实的隐蔽通道。理由是 MAC 规定高级别的进程能够进行 S_OW 的文件必定是同一个级别，而对于高级别的文件，一个低级别的进程是无法进行 S_UL 操作的。所以，不存在实际被利用的情况。

3.1.3 隐蔽流树(Covert flow trees, CFT)

隐蔽流树首先由 Porras 和 Kemmerer 在文[11]中提出，它的中心思想和 SRM 类似，也是考虑 TCB 原语与 TCB 变量之间的改写和(间接)读取关系。所不同的是，它采用了形象直观树形的表示方法，并且支持图形化的分析工具。

隐蔽流树分析的第一步是根据 TCB 的说明或者源码对每个 TCB 原语构造 3 个列表：引用列表、改写列表和返回列表，分别对应了这个原语所读取的变量、所能改写的变量以及返回结果给用户的时候引用的变量。这一步类似于 SRM 方法中构造基本的资源共享矩阵。

在具体构造一个 CFT 的时候，选择一个 TCB 的变量作为这个 CFT 分析的焦点，以之作为根节点。然后，根节点分做两个分支，左边的分支代表发送者对这个变量的改写动作(modification)可能使用的 TCB 原语，右边分支代表接收者对这个变量的改变的认知(recognition)所使用的 TCB 原语。紧接着的工作是继续扩展右边的认知分支。扩展的原则是：如果这个变量出现在某个原语的返回列表中，那么将那个原语添加成为一个"直接认知"(direct recognition)的分支，如果它不出现在任何一个原语的返回列表中，那么"失败"的页节点作为"直接认知"。如果这个变量出现在一个原语的引用列表中，并且这个原语的改写列表非空，那么这个原语被作为一个

维普资讯 http://www.cqvip.com

“间接认知”(interred-recognition)的分支被添加。然后，添加这个原语中改写列表中的每一个变量作为一个新的认知右边分支，这个原语作为左分支。对于每个新的这样的认知分支，不断递归地进行以上的操作，直到所有的叶节点都是“直接认知”或者“失败”。

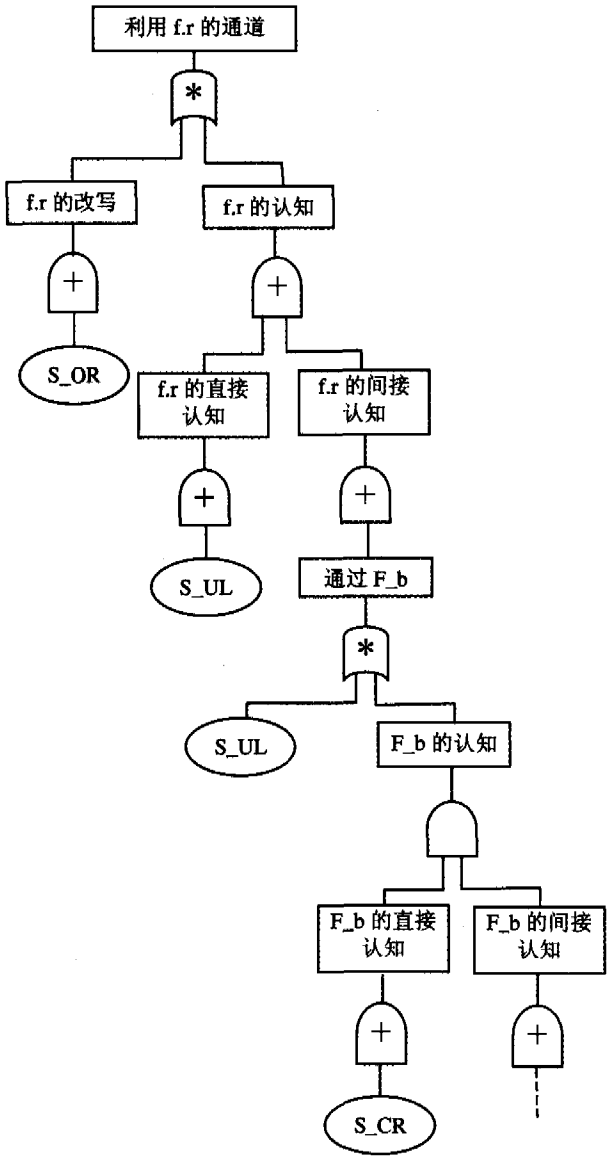

图 1　CFT 结构

通过穿越构造完成的 CFT，可以发现潜在的隐蔽通道。穿越的操作将 CFT 中所有原语的节点分成两个列表，一个包含了改写这个变量的 TCB 原语，另一个包含了直接或者间接认知这个变量的原语。这分别来自两个列表中的任意两个原语的组合构成了潜在的隐蔽通道。下面以上一个例子中的 TCB 原语和变量来说明 CFT 的构造，如图 1。

事实上图 1 并不完整，虚线省略了延伸的部分，因为变量 F_b，f.w 和原语 S_CR，S_UL 构成了一个信息传递的环路，使得这个图是一个无穷图，这个在文[11]中有详细描述，这里就不再细述。

3.1.4　信息流分析(information flow analysis)

基于语言的信息流分析(language based information flow)是一个拥有庞大分支的用于对安全属性建模或者证明的方法体系[3]。它的通用性使得它同样可以应用于隐蔽通道的发现。

这样的方法通过对每条系统的 TCB 描述或者源码语句添加信息流语义，然后通过这些语义分析系统中时候是否存在非法的信息流。举例来说对于 Pascal 语句假设 B、D、E 为常数，a、c 为变量 if a=B then c:=D else c:=E 就导致了一个从 a 到 c 的信息流动[12]。在具体分析一个多级别系统的时候，分析者根据系统的安全策略，将每个变量标签以一个安全级别。假设我们用 level(x)来表示 x 的安全级别。那么在以上的一个信息流中，如果 level(a)>level(c)那么这个信息流就是不合法的。当在系统分析的时候遇到这样非法的信息流的时候，就有可能存在一个隐蔽通道。造成这个非法信息流的原因(是那些 TCB 原语造成的)需要进一步的系统分析才能达到。至于这个隐蔽通道是否能被实际利用需要看这个通道时候能找到对应的实际利用的情况。

3.2　已有方法的比较

由于目前所有存在的方法都不能完美地解决隐蔽通道的问题，所以它们各自都有自己的优势与劣势，相对来说目前使用的比较广泛的是 SRM 方法，而如果想达到比较理想的效果，如同文[13]所说的那样，往往需要多种方法的优势互补来达到。

接下来从以下几个方面来对这几种方法进行评价和比较。

①方法作用的对象。也就是用作隐蔽通道分析的原材料，可供选择的有：TCB 的形式化说明(FTLS)，非形式化说明，和系统实现的原代码。

②方法的虚报率：也就是说在用该方法所发现的所有通道中，在实际的系统其实是一个合法的通道的概率。

③方法的自动化程度：该方法是否有自动化的工具辅助，以及对分析人员的手工分析的依赖程度。针对形式化说明的自动化分析的工具如 HDM，Ina，Gypsy[14]等。

④方法对于隐蔽通道实际利用情况发现的指导作用：也就是如果该方法发现了一个隐蔽通道，分析人员根据方法所提供的信息，构建出它实际被利用的情形的难度。

⑤方法的增量性：该方法是否可以被用于单个的 TCB 函数，进而使得当系统有新的 TCB 原语添加的时候能够在原来分析的基础上进行增量式的分析。

几种方法的对比如表 3。它们各自的有缺点因此也就一览无余。

表 3　已有各方法特点比较

方法 / 特性	作用的对象	虚报率	自动化程度	对实际利用的指导作用	增量性
非相干性分析	FTLS	无	存在自动化分析工具，但不完善	指导意义很小，需要结合其它方法	是
共享资源矩阵	FTLS、非形式化 TCB 说明、源码	较低	对于源码的分析不存在自动化工具	有一定的指导意义	否
隐蔽流树	同上	较低	同上	比较直观地反映通道的利用情况	否
信息流分析	FTLS、源码	较高或者不稳定	存在自动化分析工具，但不完善	指导意义不大	是

维普资讯 http://www.cqvip.com

4 讨论

隐蔽通道分析这么多年，可以看出所有方法之间的一些共同点，从而也不可避免地带来一些与生俱来的限制。如何突破这些限制，成为人们关注的焦点。

首先，它们都是基于一些封闭自主开发的安全操作系统。这些系统的开发的严格性以及软件工程方面的规范性保证了这些方法作用的对象都是很容易获得的，包括 DTLS、FTLS、以及具有相对稳定特性的源码（高度模块化、层次性、模块之间的低耦合性、较小的系统变动等）。而相对目前越来越多出现的基于开源的软件的安全系统如基于 Linux 的增强安全系统（LIDS、SELINUX 等），由于开源的软件在起初设计的时候并没有明确的需求分析以及设计说明，代码的风格也比较自由随意，系统模块之间的耦合程度很高，层次性不明显，并且代码变动频繁，都给传统的隐蔽通道的分析方法带来非常大的障碍，目前，类似的系统对隐蔽通道的处理都处在经验阶段。而这在另外意义上也说明了这些方法仍然面临着通用性的问题，它们目前所适应的范围还是非常窄。

其次，所有的方法所面对的都是一个系统的静态说明。如果要从这些静态说明中分析所有系统运行时候动态的所有状态可能不仅是一项非常庞杂的工作，而且也不利于精确定位隐蔽通道的实际存在。目前，用以发现动态运行时候的隐蔽通道的机制是隐蔽通道审计[15]。也就是在用以上的方法完成整个系统的静态分析之后，对可能涉及到的 TCB 原语进行审计，进而猜测可能的隐蔽通道使用。这种方法的粒度非常粗，而且误报率很高。另一方面，我们应该认识到在系统运行的某一阶段，系统所涉及到的 TCB 原语与变量都是系统的很小的一个子集，能否通过分析这些很小的子集进而确定系统不同运行阶段所涉及到的隐蔽通道是个很新颖的问题。

再次，目前所有的方法都是建立在软件层面上，其中包括一小部分硬件的软件性的描述或者说明。对于绝大部分的硬件，目前缺乏能用于隐蔽通道的规范说明。而作为一个安全系统的整体考虑，它是一个硬件与软件的整体。硬件同样也会带来隐蔽通道的问题。如果将一个系统看成一个硬件与软件融合而成的一个黑匣子，将 TCB 的接口（原语）看作是这个黑匣子的输入与输出。隐蔽通道分析的本质就退化成为对黑匣子的输入与输出之间的特定关系分析。如何分析它们之间的特定关系，将有可能成为今后作者的工作方向之一。

再次，由于互联网的飞速发展，互联网上的安全技术也不断成熟。而隐蔽通道的问题同样也存在于这个网络环境中。一个网络上的隐蔽通道可以穿越防火墙，IDS 等的安全机制来达到双方的彼此通信的目的。所以隐蔽通道分析的含义也将扩展到网络环境中。目前我们能看到的已有的工作，经典的如文[16～18]，都是一定程度上的经验主义。类似上面所说的正式的且对于网络环境通用的隐蔽通道发现方法仍然有待于发现。

结论 综上所述，我们可以看到，和其它所有的安全问题一样，隐蔽通道的问题的解决也是一个逐步成熟不断完善的过程。而随着新的形势的出现，隐蔽通道的利用与发现双方也都面临着新的问题。如何面对新的形势提出自己的看法和见解，并付以实施论证，是我们能否在这个研究领域分一杯羹的关键。

参 考 文 献

1 Lampson B W. A Note on the Confinement Problem. Communications of the ACM, 1973,16(10):613～615

2 National Computer Security Center, Department of Defense. Trusted Computer System Evaluation Criteria. DoD 5200. 28-STD, December 1985

3 Sabelfeld A, Myers A. Language-based informationflow security. IEEE Journal on Selected Areas in Communications, 2003,21(1)

4 Lipner S B. A Comment on the Confinement Problem. Operating Systems Review, 1975,9(5):192～196

5 Schroeder M D, Clark D D, Saltzer J H. The Multics Kernel Design Project. In:Proceedings of the 6th ACM Symposium on Operating Systems Principles

6 Wray J C. An Analysis of Covert Timing Channels. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, California, May 1991. 2～7

7 Kemmerer R A. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels. ACM Transactions on Computer Systems, 1981,1(3):256～277

8 Goguen J A, Meseguer J. Security Policies and Security Models. In:Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, April 1982. 11～20

9 Goguen J A, Meseguer J. Unwinding and Inference Control. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, April 1984. 75～86

10 Boebert W E, Kain R Y, Young W D. Secure Computing: The Secure Ada Target Approach. Scientific Honeyweller, July 1985, 6(2):1～17

11 Porras P A, Kemmerer R A. Covert flow trees: A technique for identifying and analyzing covert storage channels. In: 1991 IEEE Computer Society Symposium on Research is Security and Privacy, Oakland, CA, May 1991. 36～51

12 Denning D E. A Lattice Model of Secure Information Flow. Communications of the ACM, May 1976,19(5):236～243

13 Haigh J T, Kemmerer R A, McHugh J, et al. An Experience Using Two Covert Channel Analysis Techniques on a Real System Design. IEEE Transactions on Software Engineering, February 1987,13(2):157～168

14 McHugh J, Good D L. An Information Flow Tool for Gypsy. In: 1985 IEEE Symposium on Security and Privacy, April 22 - 24, Oakland, CA, 1985

15 Shieh S P, Gligor V D. Auditing the Use of Covert Storage Channels in Secure Systems. In:Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, California, May 1990

16 Girling C G. Covert channels in LAN's. IEEE Transactions on Software Engineering, February 1987,2(SE-13)

17 Ahsan K, Kundur D. Practical Data Hiding in TCP/IP. Workshop Multimedia and Security at ACM Multimedia'02, December 6, 2002, Juan-les-Pins on the French Riviera, 2002

18 Rowland C H. Covert channels in the TCP/IP protocol suite: [Tech Rep]. 5, First Monday, Peer Reviewed Journal on the Internet, July