计算机科学2000Vol. 27№. 5

隐通道识别技术研究*)

The Study of Covert Channels Identification Techniques

朱 虹 冯玉才

(华中理工大学计算机学院 武汉 430074)

Abstract We must analyse covert channels in the system(operating system or database management system)developed at or above B2 level. A secure computer system uses both discretionary and mandatory access controls to restrict the flow of information through legitimate communication channels. Covert channels are untrusted subjects at a higher security level who transfer information either using manipulation(i. e alteration)of some storage resources and the receivers at a lower security level monitor the alterations or using modulation their running time to affect the receivers' response time. This paper shows several methods in covert channel identification and compares their advantages and disadvantages.

Keywords Covert channel, Trojan horse, Multilevel security model

一、引言

计算机的广泛应用,在促进了社会的进步,为人类创造了巨大的财富的同时,也带来了计算机系统的安全性问题。目前,这个问题已越来越成为关系国计民生的重要问题,也引起了人们的广泛关注。根据 TCSEC[1],开发具有 B2级及以上的操作系统或数据库管理系统必须进行隐通道分析。隐通道分析的关键在于隐通道的识别,只有在隐通道标识出来后才能估算隐通道的带宽,并采取一些策略消除隐通道、限制隐通道的带宽或对隐通道审计。在一安全的计算机系统中,采用了自主存取控制和强制存取控制来限制信息只通过合法的通讯通道(如文件、共享内存和进程信号等)流动。但不幸的是有些不可信的主体,使用系统的某一资源位置的存取或调整自己的运行时间来影响其他主体或其他主体的响应时间而发送信息给其他的主体,这就是存储隐通道和定时隐通道。本文综述了对操作系统的隐通道分析的一些形式化方法,并比较了这些方法的优缺点。

二、基本概念

1. 基本定义

定义1 隐通道是系统的一个用户用违反系统安全策略的方式传送信息给另一用户的机制。在一个系统(操作系统或数据库管理系统)中,给定一个安全策略模型 M 及其解释 I(M),I(M)中的任何两个主体 I(S_h)和 I(S_l)间的潜在通信是隐蔽的当且仅当 M 的相应主体 S_h 和 S_l 间的通信在 M 中是非法的[2]。

定义2 特洛伊木马是一种人为地隐藏在正常软件中的秘密程序,它使所寄存的软件在完成指定功能时执行非授权的任务。通过特洛伊木马的作用,使敏感信息泄露出去。许多利用软件进行计算机犯罪的活动都是通过特洛伊木马程序进行的。

2. 隐通道的特性

隐通道包括存储隐通道与定时隐通道。如果一个隐通道是一个主体直接或间接地写一存储位置而被另一主体直接或间接地读这个存储位置时,这个隐通道是存储隐通道。如果一个隐通道是一个主体通过调整它自己的系统资源(如 CPU 时间)的使用时间,而影响了另一个主体的实际的响应时间,从而发送信息给另一主体时,这个隐通道是定时隐通道。

系统中存在既是存储隐通道又是定时隐通道的通道,此时我们说这个通道同时具备存储隐通道与定时隐通道的特性。隐通道往往通过系统原本不用于数据传送的系统资源来发送信息,并且这种通讯方式往往不被系统的存取控制机制所检测和控制。存储隐通道采用符合其编码方式的对某存储位置的处理来发送信息,即发送者修改系统的某个共享资源属性而接收者能监测到这种修改。定时隐通道则通过其经历的时间来传送信息,发送者通过调整接收者的响应时间来检

*)本文获"九五"国防预研军事项目"数据库安全技术"资助。

测某个共享实体的变化。尽管高安全级的用户有可能利用隐通道给低安全级的用户传送信息，但隐通道的主要的潜在威胁是它有可能被特洛伊木马所利用，而在安全系统之上运行不可信的应用程序又不失安全性是可信计算机系统的主要信条。开发 B2-A1级的安全系统必须对隐通道进行分析，并采取一定的策略消除或限制隐通道的使用。

3. 一个隐通道的实例

数据库系统的共享资源如表名或视图名及其访问权限就是一个潜在的存储隐通道。另外，系统实现机制不同也会引入存储隐通道或定时隐通道。

例1 在一个采用了多级安全模型（如格模型或 BLP 模型）的数据库管理系统中，采用自主存取控制和强制存取控制策略防止了数据库中的数据信息从高安全级流向低安全级，但是一个具有高安全级的不可信的主体（用户）仍然可以通过其他方式使得信息从高安全级流向低安全级。在这样的系统中有一高安全级的用户 H 和另一低安全级的用户 L，他们的非分层范围相同，只是分层密级不同。系统操作遵循下列规则：

（1）创建一基表时，基表的安全级被设置为空（即无安全级也即具有最低安全级）。

（2）系统实行自主存取控制，任何主体都可以将所建表的使用权限授予其他用户，并且这个授权的提交过程是系统自动完成的。

（3）系统的任一用户可以查询数据字典，了解自己对系统哪些资源具有操作权限，资源的创建者是谁等。

（4）系统实行强制存取控制，使得主体对客体的所有存取遵循下列多级安全模型中的原则，这两条原则主要用于防止信息从高安全级流向低安全级。

a. 向下读：仅当主体安全级中的分层密级大于或等于客体安全级的分层密级，且主体安全级中的非分层范围包含客体安全级中的所有非分层范围时，该主体才能读该客体。

b. 向上写：仅当主体安全级中的分层密级小于或等于客体安全级中的分层密级，且主体安全级中的所有范围包含于客体安全级的范围时，该主体才能写该客体。

由规则（1）用户 H 创建一基表 T，按照规则（2），他将 T 的使用权限授予用户 L（编码1）或不授予用户 L（编码0），由规则（3）L 可以通过查询数据字典了解他具有（编码1）或不具有（编码0）对表 T 操作的权限。使用表名 T 及其存取权限可以实现从 H 传送一位信息到 L，从而引起了信息从高安全级流向低安全级，这种信息传送机制在规则（4）的解释中是非法的。尽管系统采用了强制存取控制策略（规则（4）），但这种非法的信息传送机制绕过了强制存取机制的控制，这就是一个存储隐通道，如图1所示。上述信息传送方式完全可以由特洛伊木马程序来实现。

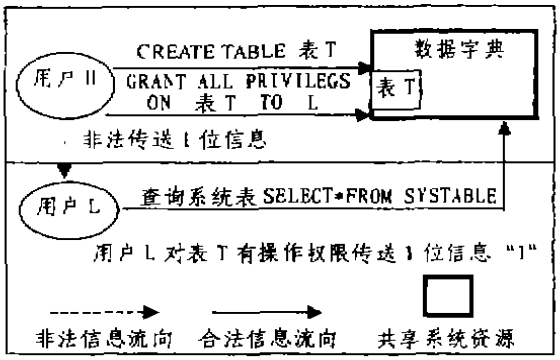

图1

三、隐通道的标识方法

对于数据库管理系统的形式化隐通道标识，文献中说明非常少，而对操作系统的隐通道分析研究有一些形式化的方法，广泛采用了基于操作系统内核形式化顶层说明的分析，包括信息流分析法、共享资源矩阵法、无干扰分析方法、隐蔽流树法等。在形式化或描述性顶层说明中标识潜在隐通道有助于查找在最终实现时导致隐通道的设计漏洞。设计漏洞的早期发现是改正设计的有用前提，此时只需要花较低的代价就可以改进这些漏洞。但是完全依赖于顶层说明查找隐通道既不适当也不需要。首先，这样做不能检测所有的实现代码中的隐通道，因为目前还没有形式化的方法来证明顶层与实现代码的一致性，这就无法保证在实现代码中不存在隐通道；而且形式化或描述性顶层说明不包含数据结构和代码的细节来检测代码中的由实现语言的语义（控制语句，如：分支语句、循环语句等，指针赋值、结构中的变量别名等）引起的间接的信息流动的描述；其次，这种分析导致了错误的信息流动，也就是存在出现在形式化/描述性说明中的隐通道可能在实现代码中不出现的情况。本节重点介绍了隐通道的表示、隐通道标识的信息来源、目前的标识方法和它们的优缺点。

1. 隐通道的表示

操作系统的隐通道可以用 TCB[1] 内部变量及 TCB 原语的两个集合表示，其中一个集合用绕过系统强制安全策略的方式变更变量值（PA_b），另一个集合以绕过系统强制策略的方式用于查看变量值（PV_i）。多个原语对于查看/变更变量是必须的，因为查看/变更变量之后，发送者和/或接收者得建立发送/接收下一

位的环境。因此隐通道识别的首要目标就是发现所有的用于变更或查看的 TCB 内部变量及 TCB 原语，即所有的三元组（变量，PA_h，PV_i）。其次，确定在通道内设置时间延迟、噪声及审计代码降低通道带宽并监控隐通道使用的 TCB 的位置。

内核变量的可查看性[5-6]与变量的引用不同，同样，变量的可变更性与变量的修改也不同。例如赋值语句 $V_a=V_b$ 表明 V_a 被修改，V_b 被引用。如果 V_b 的状态和内容不通过内核界面输出给用户进程，V_b 的引用就是用户进程不可见的。而且假设 $V_c=k$（k 为常量）是仅有的赋给 V_c 的语句，此时 V_c 被修改但不是可变更的，因为 V_c 被设置一值（V_c 被赋一常数），不能用于隐蔽信息流。若一个内核变量值的变更可被另一调用内核原语的用户进程检测，则这个变量在内核界面是可见的。即变量的可查看性和变量的可变更性以是否在内核界面可见为原则。变量可以是一个对象的属性，也可以是一种数据结构变量。

2. 隐通道识别的信息来源

隐通道识别的首要信息来源是：①包含 TCB 原语描述、CPU 和 I/O 处理器指令及其对系统对象和寄存器的影响、TCB 参数及其指令域等的系统参考手册；②详细的 B2-A1 级系统的描述性顶层说明及 A1级系统的形式化顶层说明；③TCB 源代码及处理器指令码。使用这种方法的优点是：这些信息可广泛采用。缺点是：首先，无论使用哪种系统参考手册，只能将 TCB 及处理器作为基本的"黑盒子"，系统实现细节不可见，使用系统参考手册不能达到发现所有的或近似于所有的隐通道的目的。其次，基于系统参考手册进行分析太晚而不利于隐通道的处理，一旦系统实现手册已写出，通过修改 TCB 界面规定来消除已发现的隐通道已不再可行；再次，目前还没有方法来显示仅依赖于系统参考手册进行隐通道分析的精确程度。

到目前为止，大多数识别方法使用形式化顶层 TCB 说明作为隐通道识别的主要来源。这些说明比系统参考手册往往包含更详细的、适当的信息；使用顶层说明有助于检测可能在最终实现时导致隐通道的设计漏洞，设计缺点的早期发现有助于将更正设计缺陷的代价降到最小。

3. 识别方法

（1）*符号信息流分析*　将信息流语义同每一条说明（或实现）语言的语句相联系。例如形如"a:=b"的语句引起信息从 b 流向 a（用 b→a 表示，当 b 不为常量时），类似的形如"if v=k then w:=b else w:=c"的语句引起信息从 v 流向 w。并且定义一种流向策略如："若信息从变量 x 流向 y，y 的安全级必须支配 x 的安全级"，当用于说明性语句或代码时，流向策略有助于产生流向公式。如流向公式"a:=b"，就是：a 的安全级≥b 的安全级，在信息流路径上将单个语句的所有信息流公式联合而成的完整的程序和 TCB 原语说明或 TCB 代码的流向公式，必须被证明是正确的。若某一个程序流公式不能被证明，这个特别的流向可能导致隐通道的信息流向，而必须进一步分析。即必须完成语义分析来确定：①这个未被证明的流向是真的还是错误的非法信息流向；②这个未被证明的流是否有导致隐通道的使用场景。

人们已建立了各种工具将符号流分析应用于形式化说明，符号信息流分析法有以下优点：可以以直接的方式自动实现；可用于形式化顶层说明和源代码；可以增量地用于单个函数和 TCB 原语；在特别的说明（或代码）中，它不会错过任何导致隐通道的信息流向。

符号流分析法有这样三个缺点：首先，会产生大量的错误的非法信息流向（必须通过手工语义分析消除这种流向）；不适宜用于非形式化说明；也不适宜为在 TCB 中放置隐通道处理代码提供帮助。

（2）*共享资源矩阵法*（SRM）　用 SRM 法识别隐通道是由 Kemmerer[4]提出并用于若干个项目，该方法用于 TCB 描述或代码时需要以下几个步骤：

·分析 TCB 的所有原语操作，而不管这些操作是形式化的还是非形式化说明的，还是在源代码中的。

·建立一由用户可见的 TCB 原语作为行而可查看/可变更 TCB 变量为列的共享资源矩阵，根据这些属性是读还是变更，用 R 或 M 标记每一个〈TCB 原语，变量〉项（这一步假设通过 TCB 界面已经确定了可查看/可变更变量）。那些既不可单独查看也不可更改的变量合在一起作为一个单独的变量来分析。

·对共享资源矩阵项完成一传递闭包，并加入相应矩阵项。这一步中将标识所有间接的对变量的读。只要一 TCB 原语可读的变量 x 可以被基于读变量 y 的 TCB 函数变更时，这个 TCB 原语间接地读变量 y。（只要 SRM 方法应用于非形式化的系统参考手册所定义 TCB 界面说明一而不是运用于每个原语的 TCB 内部说明一完成这一步仅能识别在 TCB 之外的进程可以怎样使用隐蔽地由 TCB 界面得到的信息。因此只要用 SRM 法将 TCB 作为黑盒子处理，就可以取消传递闭包这一步骤，因为这一步没提供任何附加的在 TCB 说明或代码内的流动信息）。

·分析包含行项是'R'或'M'的每一矩阵列，每当这些列的变量由一个进程来读而由另一变量来写而且前一进程的优先级不支配后一进程的优先级时，这些列的变量可能支持隐蔽通信。对共享资源矩阵的分析导致四种可能的结果：在两个通信进程间存在合法的通道（授权通道），用'L'标记；不能从一个通道获得有

用的信息，用'N'标记它；发送者和接收者进程是相同的，用'S'标记；一个潜在的隐通道存在，用'P'标记它；

·通过分析所有的矩阵项，发现潜在隐通道的使用场景。

这个方法有以下的优点：首先，用于形式化或非形式化 TCB 软硬件说明，也可用于 TCB 源代码；其次，它不要求赋予表示在矩阵中的内部 TCB 变量安全级，消除了主要的错误的非法信息流的来源。但缺少安全级赋予变量有如下负面的影响：首先，个别的 TCB 原语（或原语对）不能孤立地被证明是安全的，这一缺点增加了增量分析新的 TCB 功能的复杂性；其次，SRM 分析可标识潜在的隐通道，而这种隐通道可由信息流分析法自动消除。

尽管 SRM 法可以用于源代码分析，但是给 TCB 源代码自动构造共享资源矩阵（这项工作是最繁琐的步骤）的工具至今不存在。

（3）*无干扰分析*　要求将 TCB 看成抽象机器。从用户进程的观点来看，TCB 提供了某种服务，一个进程的请求代表了抽象机器的输入。TCB 的响应（即数据值、错误消息、正确响应）是 TCB 的输出，而 TCB 内部变量的内容构成了它当前的状态。每一个输入都引起 TCB 状态的变化（如果必要的话）和输出，一个用户进程与另一个用户进程间无干扰是指将从初始状态起所有的来自第一个进程的输入取消好象它们从未输入过一样，而由第二个进程观察到的输出不变。即一个用户进程的输入不能影响另一个进程的输出，则在第一个进程和第二个进程间没有信息传递。

令 X 和 Y 是某个抽象机 TCB 的两个用户进程。若 w 是机器的以来自 Y 的一个输入结尾的输入序列，令 Y(w)是输出 Y 接收最后的输入后的输出（假定当 w 输入时机器处于初始状态）。为了表达无干扰，设 w/X 是所有的 X 输入从 w 中删除后的剩余子序列。如果对所有可能的以 Y-输入结尾的输入序列有 Y(w)＝Y(w/X)，则 X 与 Y 间无干扰。

无干扰将整个输入序列包括许多 X-输入与单个 Y-输出相联系，在隐通道分析中，传统的观点是一旦 X 和 Y 间存在隐通道，每一个单个的 X-输入都对下一个 Y-输出有影响。而无干扰分析提出了另一种观点，Y 的每一个输出都不影响所有的以前的 X 输入。这似乎有必要分析所有过去的 X 输入因为假如每个 X 输入某一延时都报告有一 Y 输出，就象若 X 输入立即引起下一个 Y 输出一样，隐通道即产生了。要分析从机器的初始状态以来的所有历史输入是非常麻烦的，也是不必要的，因为当前状态由所有的必要的信息确定下一个 Y-输出，于是 X 和 Y 间无干扰可以用当前状态而不是所有优先输入的历史状态表示。若 X 与 Y 间无干扰，X 的输入对 Y 的下一输出无影响，而无干扰的要求比这更强，因为它要求 X 的输入对 Y 的任何后续输出无影响。为避免分析无界的输入序列，将 TCB 的状态分成等价类，不必区分是使用现在的还是后续的 Y 输出。两个状态是 Y 等价的，是指：①对相同的 X 输入有相同的 Y 输出；②任意的输入后相应的下一状态也是等价的，即，X 与 Y 间无干扰当且仅当每个 X 的输入占据一个等价于 Y 的状态。用这种方法即可查找系统中的隐通道，

无干扰分析有以下优点：可用于形式化的 TCB 描述和源代码、避免发现假的非法流、可增量地用于单个的 TCB 函数和原语。但也有三个实际的缺点：可用于形式化的 TCB 顶层描述，还可用于源代码，将无干扰手工应用于大型 TCB 是不实际的，无干扰分析是"乐观的"，它试图证明在 TCB 描述或代码中不出现干扰，它的最佳应用是可信进程分离的 TCB 描述而不是包含大量共享变量（即内核）TCB 构件。

（4）*隐蔽流树法*[7]　这是一种较新的方法，提供了查找构成隐通道的场景的方法。隐蔽流树（CFT）法使用了树的数据结构模型化了信息从一个共享资源向另一个共享资源的流动，实现了系统地搜索通过共享变量属性发送而最终被监听进程所检测的信息传送方式。构造隐蔽流树所需要的信息与基本共享资源矩阵所需要的信息相同。每一个操作都用三个列表表示：引用列表、修改列表和返回列表。返回列表包含更新用户输出时被引用的资源属性。由文[7]提出的 CFT 法不区别流向多个目标的信息流，但只要稍作修改就可以标识多个目标的信息流。

首先确定一个重点分析的资源属性，CFT 树由发送者所作的对资源属性的修改动作和由接收者对该资源属性的修改的认可动作序列组成，CFT 树根的左子树由发送者所作的对资源属性修改的一系列调用序列组成；树根右子树由接收者的所作的认可资源属性修改的一系列操作序列组成。CFT 树的构造是将修改列表中所包含的对象的任一操作加入到认可路径的直接认可分支，将在引用列表中所包含的对象的任意操作加入到导出的认可分支的操作中。对上述加入的资源属性修改列表，加入直接和间接的认可操作，反复进行这个过程，直到所有的路径以直接认可结束或达到预先指定的深度并且剩余的导出认可路径以"false"标记。

周游 CFT 树得到由发送者和接收者传送信息的操作序列。将这些序列简化，删除冗余的操作对或加入建立某一操作的有效操作的前提条件的操作（如：在读

（下转第99页）

为 $\sigma_{t\subseteq F}(R)$，这个表达式比例1中约束代数表达式更为简单。

例4 下列的空间查询能用 NCALG 代数表示：

区域包含：在 R 中，选择包含于空间区域 rt 的空间对象，rt 的约束关系表示为 $p_r\sigma_{t\subseteq F}(R)$。

区域相交：在 R 中，选择和空间区域 rt 相交的空间对象，rt 的约束关系表示为 $p_r\sigma_{t[\sim]p}(R)$。

相邻查询：选择与空间对象 sp 相邻的对象，sp 的约束关系表示为 $p_r\sigma_{c_1}(\sigma_{t_2}(R))_r c_1\equiv\neg(Q_{Int}(t)\rhd\lhd Q_{Ext}(p))_r c_2\equiv(Q_{Bnd}(t)\rhd\lhd Q_{Bnd}(p))$。

空间连接(基于相交)：选择空间对象对(r，s)，$r\in R$，$s\in S$，r 与 s 相交。$\sigma_c(R\rhd\lhd\rho_{[x,x',y,y']}(S))_r c\equiv(Q_1(t)\rhd\lhd Q_2(t))$，$Q_1(t)\equiv\pi_{[x,y]}(t)$，$Q_2(t)\equiv\rho_{[x|x',y|y']}(\pi_{[x',y']}(t))$。

空间连接(基于相邻)：选择空间对象对(r，s)，$r\in R$，$s\in S$，r 与 s 相邻。$\sigma_{c1}(\sigma_{c2}(R\rhd\lhd\rho_{[x,x',y,y']}(S)))_r c_1\equiv\neg(Q_{1,1}(t)\rhd\lhd Q_{1,2}(t))$，$Q_{1,1}(t)\equiv Q_{Int}(\pi_{[x,y]}(t))$，$Q_{1,2}(t)\equiv Q_{Int}(g(t))$，$c_2\equiv(Q_{2,1}(t)\rhd\lhd Q_{2,2}(t))$，$Q_{2,1}(t)\equiv Q_{Bnd}(\pi_{[x,y]}(t))$，$Q_{2,2}(t)\equiv Q_{Bnd}(g(t))$，$g(t)\equiv\rho_{[x|x',y|y']}(\pi_{[x',y']}(t))$。

差查询：选择 R 中的空间对象 r，S 中没有一个对象 s，r 和 s 在 x 上的投影是相等的。$\pi_{[x,y]}(\sigma_c((\pi_{-x}(R)\setminus\pi_{[x]}(S))\rhd\lhd R'))$，$R'\equiv\rho_{[x|x',y|y']}(R)$，$c\equiv(\pi_{[x]}(t)=\rho_{[x|x']}(\pi_{[x']}(t)))$。

Q_{Bnd}和 Q_{Ext}是分别查询空间对象的边界和内部的简写。

结论 约束数据库用数学约束来建模可能无限的关系元组集，本文中，为了解决 Kanellakis 约束数据模型和约束代数在建模空间对象时的不足之处，在嵌套关系语义的基础上，我们提出了嵌套约束数据模型和嵌套约束代数 NCALG。通过几个例子，我们证明了这个代数比约束代数 CALG[2]更适合于时空应用。文[6]给出了嵌套语义下的约束数据库实例区间约束数据库，它以易于实现而受到约束数据库研究者的重视。

参考文献

1 Kanellakis P C, et al. Constraint Query Languages. JCSS 51, 1995

2 Goldin D Q, Kanellakis P C. Constraint Query Algebras. Constraints Journal, 1996, 1(1)

3 Kanellakis P C, Goldin D Q. Constraint Programming and Database Query Languages. In: LNCS 789: Proc. of the Int. Symp. on Theoretical Aspects of Computer Software. 1994. 96～12

4 Abiteboul S, Kanellakis P C. Query Languages for Complex Object Databases. SIGACT News, 1990, 21(3): 9～18

5 Grumbach S, et al. A Spatial Constraint Database. In: Proc. Workshop on Database Programming Language 1997

6 Chen Lianggang, et al. Interval Constraint Databases and ISQL Language. The Fifth Intl. Conf for Young Computer Scietists. 1999

(上接第103页)

一文件前先打开该文件等)。这些操作序列被检查并确定哪些表示合法操作，将合法操作删除。对剩余的序列进行分析看它们是否可以建立隐通道。

总之，尽管目前有一些形式化的方法来标识隐通道，无论采用什么方法都需要手工进行分析，所有形式化方法都无法代替经验、技巧和敏捷的头脑来完成隐通道的识别。

参考文献

1 National Computer Security Center. Department of Defense Trusted Computer System Evaluation Criteria. Dec. 1985. DoD 5200. 28-STD

2 National Computer Security Center. A Guide to Understanding Covert Channel Analysis of Trusted Systems. NCSC-TG-030, library No. S-240, 572, Nov. 1993

3 McHugh J. Covert Channle Analysis, a Chapter of the Handbook for the Computer Security Certification of Trusted Systems. NRL. Technical Memorandum 5540: 062A, 1996, Naval Research Laboratory, Washington

4 Kemmerer R A. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels. ACM Trans. on Computer Systems, Aug. 1983

5 Ne J, Gligor V D. Information Flow Analysis for Covert-Channel Identification in Multilevel Secure Operating Systems. In: Proc. of the 3rd IEEE Workshop on Computer Security Foundations. Franconia, New Hampshire, June 1990. 139～148

6 Tsai C R, et al. A Formal Method for the Identification of Covert Storage Channels in Secure Code. 1987 IEEE Symposium on Security and Privacy, Oakland, CA IEEE Computer Society, Computer Society Press, 1987. 74～86

7 Porras P A, Kemmerer R A. Covert Flow Trees: A Technique for Identifying and Analyzing Covert Storage Channels 1991 IEEE Computer Seciety Symposium on Research in Security and Privacy, Oakland, CA, May, 1991. 36～51

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

9 Millen J K. Finite-State Noiseless Covert Channels. In: Proc. of the Computer Security Foundations Workshop Franconia, New Hampshire, June 1989. 81～85