高安全等级安全操作系统的隐蔽通道分析*

卿斯汉[1,2+]

[1](中国科学院 信息安全技术工程研究中心,北京 100080)

[2](中国科学院 软件研究所,北京 100080)

Covert Channel Analysis in Secure Operating Systems with High Security Levels

QING Si-Han[1,2+]

[1](Engineering Research Center for Information Security Technology, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62635150, E-mail: qsihan@ercist.iscas.ac.cn, http://www.ercist.iscas.ac.cn

Received 2004-04-29; Accepted 2004-07-16

Qing SH. Covet channel analysis in secure operating systems with high security levels. *Journal of Software*, 2004,15(12):1837~1849.

http://www.jos.org.cn/1000-9825/15/1837.htm

Abstract: The thirty years, development of covert channel analysis research is summarized. The essence of covert channel and its analysis is depicted according to our theoretical research and engineering experience in this area. The state of the art in the application of covert channel analysis approaches to the real systems is illustrated. Some major threads and emerging trends of the research in this area are presented.

Key words: secure operating system; covert channel analysis; information flow; storage channel; timing channel

摘 要: 总结隐蔽通道分析的 30 年研究进展,根据理论研究与工程实践,说明隐蔽通道及其分析的本质与内涵,指出隐蔽通道分析方法在实际系统中的重要应用,并展望这一领域的若干热点研究方向.

关键词: 安全操作系统;隐蔽通道分析;信息流;存储通道;定时通道

中图法分类号: TP301 文献标识码: A

隐蔽通道的概念最初是由 Lampson[1]于 1973 年提出.他在论文“关于限制问题的注释”中这样定义隐蔽通道:“如果一个通道既不是设计用于通信,也不是用于传递信息,则称该通道为隐蔽通道”.迄今为止,关于隐蔽通道的研究,已经历了 30 年的发展历程,并取得了一系列重要的研究成果[2,3].

隐蔽通道的存在,对安全操作系统是一个重要的威胁.因此,对高安全等级的安全操作系统,各种标准都要求进行隐蔽通道分析.

1985 年,美国国防部发布了橘皮书 TCSEC[4],这是第一个“计算机安全产品评估标准”.橘皮书明确规定,在对 B2 级以上的高等级安全操作系统进行评估时,必须分析隐蔽通道,并且随着安全级别的提高,对隐蔽通道分

* Supported by the National Natural Science Foundation of China under Grant No.60083007 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.G1999035802 (国家重点基础研究发展规划(973))

作者简介: 卿斯汉(1939－),男,湖南隆回人,研究员,博士生导师,主要研究领域为信息系统安全理论和技术.

析的要求越来越严格.

我国国家标准 GB17859-1999“计算机信息系统安全保护等级划分准则”、GB/T 18336-2001“信息技术 安全技术 信息技术安全性评估准则”以及其他相关的国际标准,都有类似的规定.

本文总结隐蔽通道分析的 30 年研究进展,说明隐蔽通道及其分析的本质与内涵,指出隐蔽通道标识方法、带宽分析方法和处理方法在实际安全操作系统中的重要应用,并展望这一领域的若干热点研究方向.因篇幅所限,本文着重讨论隐蔽存储通道.

1 隐蔽通道的基本概念

1.1 隐蔽通道的定义

正确理解隐蔽通道的定义,掌握隐蔽通道的本质与内涵,是隐蔽通道分析的第一步.随着研究的不断深入,人们对隐蔽通道的认识也逐渐加深.

1977 年,Schaefer[5]将隐蔽通道定义为:“如果一个通道从存储单元向描述资源状态的变量传输信息,则称该通道为隐蔽通道”.

1978 年,Huskamp[6]为隐蔽通道下了一个新定义:“如果一个通道是通过资源分配策略和资源管理实现产生的,则称该通道为隐蔽通道”.

1983 年,Kemmerer[7]则将隐蔽通道定义为:“如果一个通道使用非数据客体项从一个主体向另一个主体传输信息,则称该通道为隐蔽通道”.

1990 年,Tsai, Gligor 和 Chandersekaran[8]提出了一种新的观点.他们认为,隐蔽通道与强制访问控制策略有密切联系,亦即,“给定一个强制安全策略模型 M 和它在一个操作系统中的解释 $I(M)$,$I(M)$中两个主体 $I(S_i)$和 $I(S_j)$之间的任何潜在通信都是隐蔽的,当且仅当模型 M 中的相应主体 S_i 和 S_j 之间的任何通信在 M 中都是非法的”.

Lampson,Schaefer,Huskamp 和 Kemmerer 的定义,都是从某一个侧面描述隐蔽通道.Tsai 等人的定义则较为全面地刻画了隐蔽通道的内涵,这是因为:

(1) 指出隐蔽通道分析仅与强制安全策略模型有关,而与自主安全策略模型无关;

(2) 指出隐蔽通道分析不仅与安全性模型相关,而且与完整性模型相关;

(3) 指出隐蔽通道分析与 TCB 规范有关.

1.2 隐蔽通道的分类

我们可以从不同的角度对隐蔽通道进行分类,下面分别加以说明.

1.2.1 隐蔽存储通道与隐蔽定时通道

在工程实践中,为了证明真实隐蔽通道的存在,必须构造隐蔽通道实现的场景.1975 年,Lipner[9]提出,根据场景的不同,可以将隐蔽通道划分为隐蔽存储通道和隐蔽定时通道两大类.

如果隐蔽通道实现的场景是,一个进程直接或间接地写一个存储单元,另一个进程直接或间接地读该存储单元,则称这种隐蔽通道为隐蔽存储通道.

如果隐蔽通道实现的场景是,一个进程通过调节它对系统资源(例如:CPU 时间)的使用,影响另外一个进程观察到的真实响应时间,实现一个进程向另一个进程传递信息,则称这种隐蔽通道为隐蔽定时通道.

注意,在上述定义中,进程即为安全策略模型在操作系统中的解释 $I(M)$的主体.

虽然,我们可以将隐蔽通道分为隐蔽存储通道和隐蔽定时通道两类,但它们在本质上是相同的.二者的相同之处是,它们至少包含一个存储变量,用于传递隐蔽信息.在隐蔽定时通道中,将参考时钟换成同步存储变量,隐蔽定时通道就可以变为隐蔽存储通道.类似地,在隐蔽存储通道中,将同步存储变量换成参考时钟,隐蔽存储通道就变为隐蔽定时通道.

隐蔽存储通道和隐蔽定时通道的重要不同之处表现在下述两个方面:(1) 信息编码的方式不同;(2) 隐蔽定时通道是无记忆通道,隐蔽存储通道则是有记忆通道.

1.2.2 噪音通道与无噪通道

任何通信信道都可以分为噪音通道和无噪通道两大类,隐蔽通道也不例外.不过,由于隐蔽通道的固有特征,亦即,信息传递以比特为单位,隐蔽噪音通道和隐蔽无噪通道的定义也具有自己的特点.

在隐蔽通道中,如果对发送进程传送的任意比特,接收进程都能以概率 1 正确地解码,则称该通道为无噪通道.相反地,在噪音通道中,对发送进程传送的任意比特,接收进程能够正确解码的概率小于 1.

由上述定义可知,对于无噪通道,不论系统中同时存在多少进程,不论其他进程如何运行,接收进程都能以概率 1 正确解码发送进程传送的任意比特信息.

1.2.3 聚集通道与非聚集通道

为了计算隐蔽通道最大带宽的需要,我们除了研究单独标识的隐蔽通道,即非聚集隐蔽通道之外,还需要研究聚集隐蔽通道.

在隐蔽通道中,发送进程与接收进程使用的同步变量或信息,可以用于对多个数据变量进行操作,则称这种通道为聚集隐蔽通道.由此可见,上述定义强调的是“多个”变量.因此,聚集通道与只用一个数据变量的多个通道是不同的.

至于多个数据变量,既可以单独使用,也可以作为一组使用,以便分摊同步信息的代价.根据发送进程与接收进程设置、读、重置这多个变量方式的不同,聚集通道可以进一步分为串行聚集通道、并行聚集通道和串并混合聚集通道.

例如,如果发送进程与接收进程串行地设置、读、重置所有的数据变量,则该通道构成串行聚集通道.相反地,如果多个发送进程与接收进程并行地设置所有的数据变量,则该通道构成并行聚集通道.又如,如果发送进程与接收进程并行地设置数据变量,但串行地读数据变量,则该通道构成串并混合聚集通道.

但是,串并混合聚集通道无助于达到最大带宽,因此,没有理论研究与工程实践的意义.

1.2.4 基于安全缺陷的隐蔽通道分类方法

产生隐蔽通道的根本原因在于,系统中存在安全缺陷.虽然,Tsai 关于隐蔽通道的定义没有区分隐蔽通道和 TCB 规范的缺陷,但该定义强调模型实现,启示我们可以基于隐蔽通道与系统安全缺陷的相关性,将隐蔽通道划分为如下 3 类:

(1) 基本型隐蔽通道.TCB 规范缺陷产生基本型隐蔽通道的充要条件是,在任何操作系统中,对强制安全策略模型进行任何解释,该 TCB 规范缺陷始终存在.

(2) 特定 TCB 型隐蔽通道.TCB 规范缺陷产生特定 TCB 型隐蔽通道的充要条件是,一个给定的操作系统中,在强制安全策略模型的一个特定解释下,该 TCB 规范缺陷存在.

(3) 不合理型隐蔽通道.TCB 规范缺陷产生不合理型隐蔽通道的充要条件是,一个给定的操作系统中,仅在强制安全策略模型一个特定的但不合理的解释下,该 TCB 规范缺陷才存在.

显然,对于任何高安全等级的安全操作系统,我们都应该通过修改 TCB 规范或模型实现,消除不合理型隐蔽通道.

1.2.5 无害通道与有害通道

类似于将计算机病毒分为良性病毒和恶性病毒两类,我们也可以将隐蔽通道划分为无害隐蔽通道与有害隐蔽通道两大类.

无害通道通常具有以下特征之一:(1) 发送进程与接收进程相同;(2) 系统安全策略允许发送进程与接收进程直接通信;(3) 实际应用该通道十分困难.

相反地,有害通道的特征是:(1) 发送进程与接收进程不同;(2) 系统安全策略不允许发送进程与接收进程通信;(3) 存在一种利用系统安全缺陷构造隐蔽通道的有效机制,使发送进程可以在有限时间内,向接收进程传送一定数量的有用信息.

鉴于我们研究的重点是隐蔽通道的危害性,今后,本文提及的隐蔽通道均指有害通道.

1.2.6 小 结

我们从不同的观点出发,根据不同的需求,可以对隐蔽通道进行不同类型的分类.结合上述分类方法,我们

还可以将隐蔽通道进一步细分,例如,无噪隐蔽存储通道、有害隐蔽定时通道、有害隐蔽存储噪音通道等.

此外,我们也可以根据隐蔽通道的工作方式对隐蔽通道进一步分类.例如,存储通道可以分为:资源耗尽型通道、事件计数型通道、策略冲突型通道等.定时通道可以分为:CPU 调度型通道、I/O 调度型通道、共享硬件资源型通道、存储资源管理型通道等.

1.3 隐蔽通道产生的条件

Kemmerer 给出了隐蔽存储通道存在的必要条件:

(1) 发送进程与接收进程都具有访问一个共享资源的同一属性的权限;

(2) 发送进程可以修改一个共享资源的属性;

(3) 接收进程可以检测该共享资源属性的改变;

(4) 存在某种机制,能够启动发送进程与接收进程之间的通信,并正确调节通信事件的顺序.

Kemmerer 也给出了隐蔽定时通道存在的必要条件:

(1) 发送进程与接收进程都具有访问一个共享资源的同一属性的权限;

(2) 发送进程与接收进程都具有访问一个参考时钟,例如实时时钟的权限;

(3) 发送进程能够调节接收进程检测共享资源属性变化所需的响应时间;

(4) 存在某种机制,能够启动发送进程与接收进程之间的通信,并正确调节通信事件的顺序.

在所有的系统中,处理器都是共享的.因此,所有的进程都共享属性——CPU 响应时间.亦即,接收进程通过监视系统时钟就可以检测 CPU 响应时间的变化.

1.4 潜在隐蔽通道与真实隐蔽通道

将各种隐蔽通道标识方法静态地应用于顶层规范或源代码时,产生出许多候选的隐蔽通道,即潜在隐蔽通道.但是,这些通道并非都是真实隐蔽通道.因为在系统动态运行时,只有满足一定条件,信息流才会发生.某些静态分析时存在的流条件,在系统动态运行时永远不可能发生.所以,有些非法流不可能构成真实隐蔽通道.此外,如果观察/修改隐蔽通道变量的操作与实际应用场景要求的操作不同,潜在隐蔽通道也不可能成为真实隐蔽通道.例如,有些 TCB 原语只有在特定的参数选择与特定的 TCB 状态下,才能利用隐蔽通道传送信息.因此,证明一个潜在隐蔽通道是真实隐蔽通道的方法是,构造出隐蔽通道实时应用的场景.

1.5 隐蔽通道分析的内容与层次

隐蔽通道的分析包含 3 个方面的内容:(1) 隐蔽通道标识;(2) 隐蔽通道带宽的计算与工程测量;(3) 对被标识的隐蔽通道进行适当的处理.

原则上,隐蔽通道分析可以在安全操作系统任何一个层次上进行.分析的抽象层次越高,越容易在早期发现系统开发时引入的安全漏洞.通常,根据实际需要与所采用的分析方法,隐蔽通道分析在以下 3 个层次进行:(1) 描述性顶层规范(DTLS)级;(2) 形式化顶层规范(FTLS)级;(3) 源代码级.

2 隐蔽通道的标识方法

彻底搜索隐蔽通道,即隐蔽通道标识的工作是隐蔽通道分析中最为困难的一环.其困难性体现在理论和工程实践两个方面:(1) 理论上仍然不够成熟,缺乏严谨且行之有效的方法;(2) 实际工作量庞大,手工分析容易出错,缺乏行之有效的自动工具.

2.1 共享资源矩阵法

2.1.1 共享资源矩阵法

共享资源矩阵法,以下简称 SRM 方法,Kemmerer 于 1983 年提出,是迄今为止最为成功的一种隐蔽通道标识方法.

该方法的分析步骤是:

1. 分析所有的 TCB 原语操作,确定通过 TCB 接口用户可见/可修改的共享资源属性;

2. 构造共享资源矩阵,该矩阵的各行对应于用户可见的TCB原语,各列对应于用户可见/可修改的共享资源属性.如果一个原语可以读一个变量,则将该矩阵项〈TCB原语,变量〉标记为*R*.类似地,如果一个原语可以修改一个变量,则将该矩阵项〈TCB原语,变量〉标记为*M*.最后,将既不能读又不能写的变量合并,分析时将它们视为一个变量.

3. 对共享资源矩阵完成传递闭包操作,具体步骤如下:在矩阵中搜索包含标记*R*的每一项,如果该项所在的行中出现*M*标记,则检查包含该*M*项的所在列.如果在该列的任意一个行中出现*R*标记,且该行与原始*R*项所在列的对应行中没有*R*标记,则在该矩阵项中增加间接读标记*r*.重复以上操作,直到矩阵中无法再增加*r*项时为止.注意,这里区分*r*与*R*仅表明,*r*为间接读,*R*为直接读.在今后的分析中,将*r*等同地视为*R*.

4. 分析每个矩阵行,找出同时包含*R*和*M*的行,并删去其他矩阵行.当一个进程可以读一个变量且另一个进程可以写该变量时,如果写进程的安全级支配读进程的安全级,就可能产生潜在的隐蔽通道.通过对矩阵项的分析,可以得到以下4种不同类型的通道:

(1) 该通道为合法通道,将它标记为"*L*";

(2) 从该通道无法获得有用的信息,将它标记为"*N*";

(3) 发送进程与接收进程是同一个进程,将它标记为"*S*";

(4) 该通道为潜在的隐蔽通道,将它标记为"*P*".

5. 分析矩阵所有的项,构造潜在隐蔽通道的实际应用场景.可以构造出实际应用场景的潜在隐蔽通道,即为真实隐蔽通道.

SRM方法具有以下优点:

1. 该方法简单、直观,实际应用广泛,有多个成功应用的例子[7].

2. 在DTLS,FTLS和源代码级均可应用.

3. 适用于隐蔽存储通道和隐蔽定时通道.

SRM方法的主要缺点是:

1. SRM方法中,在源代码级构造共享资源矩阵工作量最大,但迄今没有自动工具可以应用.手工构造效率低,且容易出错.

2. 该方法不能证明单个的TCB原语或原语对是安全隔离的,因此增量分析新的TCB原语十分不便.

3. 该方法过于保守,即它所标识的潜在隐蔽通道常常不是真实隐蔽通道.

SRM方法自问世以来,出现了各种衍生方法.例如,Porras和Kemmerer[10]提出的隐蔽流树(CFT)方法,在某种意义上可以看作是对SRM方法的补充和发展.构造隐蔽流树所需的信息与构造基本共享资源矩阵所需的信息基本一致,每个操作分别用一个引用表、一个修改表和一个返回表表示.有关详细内容,请参考文献[10].

下面两节,我们介绍另外两种SRM方法的衍生方法.

2.1.2 改进的共享资源矩阵法

从某种意义上讲,共享资源矩阵法是一种"悲观"方法,或者说是一种"过保护"的方法.因此,1995年,McHugh[11]对SRM方法进行了改进.从下述代码片段所构造的共享资源矩阵可以说明这个问题.

```
Global var a,b,c,d
Procedure SC1 (Var u: uvar) =
begin
   a := if b then c else d;
   u := c;
end;
```

资源属性	系统调用 SC1
a	*M*
b	*R*
c	*R*
d	*R*
u	*M*

上述共享资源矩阵说明,既可以通过变量*b*,*c*,*d*中的信息修改变量*a*;也可以通过变量*b*,*c*,*d*中的信息修改变

量 *u*.但是,对源代码进行分析之后证明并非如此.事实上,信息流将受到更多的限制.当系统没有安全缺陷时,这种保守方法是无害的.但当系统存在安全缺陷时,这种方法所标识的大量潜在隐蔽通道将不会是真实隐蔽通道.

为此,McHugh 建议在应用 SRM 方法执行传递闭包操作之前,对共享资源矩阵进行以下 3 方面的细化:(1) 区分用户与系统之间的信息流和状态属性之间的信息流.为此,我们将所有从用户到系统的输入归并为共享资源矩阵中的一行,并标记为 *u_in*.该行的所有项均为 *R*.类似地,我们将所有从系统返回给用户的输出归并为共享资源矩阵中的一行,并标记为 *u_out*.仅当用户从系统调用的返回值中,可以获得状态属性的相关信息时,该行的项才为 *M*.(2) 区分信息流的流入属性.为此,我们将共享资源矩阵的列进一步细分,使每一列只包含一个 *M* 项.(3) 区分信息流产生的条件.为此,我们将共享资源矩阵的列进一步细分,使每一列都对应于信息流的某一个产生条件.

这样,上述基本共享资源矩阵就变成如下形状.显然,这时,共享资源矩阵反映了源代码所指明的信息流的真实流动情况.

资源属性	系统调用 SC1(无条件)	系统调用 SC1(=*b*)	系统调用 SC1(≠*b*)
a		*M*	*M*
b		*R*	*R*
c	*R*	*R*	
d			*R*
u_in	*R*	*R*	*R*
u_out	*M*		

2.1.3 模块化共享资源矩阵法

1996 年,Kemmerer 和 Taylor[12]提出了一种模块化共享资源矩阵法,并成功地应用于 Trusted DG/UX 安全操作系统(B2 级)的隐蔽通道分析.DG/UX 是一种类 Unix 系统,它的内核包括大约 170 个子系统,400 余个系统调用,分别包含在 23 个子系统之中.这些系统调用构成内核的外部接口.DG/UX 的内核是高度结构化的,它的每一个系统状态变量都只受一个子系统控制,亦即,只有控制子系统的函数才能访问或修改这些状态变量.在一个子系统的函数中,只有显式地声明为输出函数,才能被其他子系统调用.所以,当一个子系统需要访问由其他子系统控制的系统状态变量时,它必须调用该控制子系统的某个输出函数.因此可以说,DG/UX 的内核子系统具有高内聚性和低耦合性的特征.亦即,每个子系统都是一个单独的逻辑实体,且各个子系统都相互独立.

基于 DG/UX 安全操作系统的上述特征,即可以将子系统视为抽象客体,因此在搜索该系统的隐蔽通道时,Kemmerer 和 Taylor 没有直接将 SRM 方法应用于 DG/UX 的整个内核,而是采用了一种模块化的 SRM 方法.在该方法中,需要定义所谓“同等子系统”.如果一个系统调用,可以直接调用多个子系统的输出函数,则将所有这些子系统称为“同等子系统”.例如,对于系统调用 open,同等子系统包括通道管理子系统 cm 和路径名子系统 pn.系统调用 open 本身,则包含在“系统调用接口”子系统 sci 之中.由于 DG/UX 内核子系统的高内聚性,一个同等子系统的大多数输出函数,或者被某个系统调用直接调用,或者被系统调用直接调用的某个输出函数间接调用.例如,通道管理子系统 cm 共有 48 个输出函数,其中 24 个被系统调用接口子系统 sci 直接调用.

模块化 SRM 方法的分析步骤是,首先对所有的同等子系统进行 SRM 分析,然后应用上述分析所获得的信息进行整个内核的 SRM 分析.

该方法的主要优点是:

1) 可以将一个大系统的隐蔽通道分析分解为多个子系统的隐蔽通道分析,对每个子系统的分析,可以由该子系统的设计或测试分析人员独立完成.

2) 在系统的设计、分析和测试过程中,随着系统的演化,可以方便地进行增量隐蔽通道分析.

该方法的缺点是:

1) 仅适用于高度结构化的内核,即要求操作系统内核的子系统具有高内聚性和低耦合性的特征.

2) 仍然缺乏构造与分析共享资源矩阵的自动工具.当时,仅开发了生成函数依赖关系的自动工具和生成传递闭包的自动工具.

2.2 语法信息流方法

Denning 的信息流格模型[13]是语法信息流方法中最著名的一个,也是最初的系统分析隐蔽通道的方法.

语法信息流方法的分析步骤是,(1) 将信息流语义附加在每个语句之后.例如,当 b 不为常数时,赋值语句 $a:=b$ 产生由 b 到 a 的信息流,用 $a\leftarrow b$ 表示,并称之为“明流”.类似地,条件语句产生暗流.例如,if $x=a$ then $y:=b$ else $z:=c$ 产生的暗流是 $y\leftarrow x$ 和 $z\leftarrow x$.这时,同时存在明流 $y\leftarrow b$ 和 $z\leftarrow c$.(2) 定义安全信息流策略,例如“如果信息从变量 b 流向变量 a,则 a 的安全级必须支配变量 b 的安全级”.(3) 将流策略应用于形式化顶层规范或源代码,生成信息流公式.例如,$a:=b$ 的流公式为 $SL(a)\geq SL(b)$,其中 $SL(x)$表示变量 x 的安全级.(4) 证明流公式的正确性.如果无法证明某个流公式的正确性,则需要进一步对语句进行语义分析,并判断该信息流:(a) 是非法流还是伪非法流;(b) 是否能够产生真实隐蔽通道,而不只是潜在隐蔽通道.

该方法的主要优点是:

(1) 可应用于形式化顶层规范和源代码,并易于进行自动分析.

(2) 可以增量分析单个函数或 TCB 原语.

(3) 不会漏掉可能产生隐蔽通道的非法信息流.

该方法的缺点是:

(1) 不能应用于描述性顶层规范.

(2) 该方法对每个变量或客体都显式或隐式地赋予特定的安全级,因此会产生大量的伪非法流,通过手工语义分析消除伪流需要增加很大的工作量.

(3) 不能准确确定安置隐蔽通道处理代码的 TCB 位置.

2.3 语义信息流方法

1990 年,Tsai, Gligor 和 Chandersekaran[8]对语法信息流方法作了重大改进,增加了语义分析.Tsai 等人提出了一种标识隐蔽存储通道的新方法,基于:(1) 分析编程语言的语义、代码和内核中使用的数据结构,发现变量的可见性/可修改性;(2) 解决内核变量的别名问题,确定内核变量的间接可修改性;(3) 对源代码进行信息流分析,确定内核变量的间接可见性.他们的语义信息流方法在 Secure Xenix 系统的应用中获得成功.

语义信息流方法的分析步骤是,(1) 选择用于隐蔽通道分析的内核原语;(2) 确定内核变量的可见性/可修改性;(i) 通过语义分析,确定内核变量的直接可见性/可修改性;(ii) 对每个原语生成一个“函数调用依赖关系”集合 FCD;(iii)通过信息流分析,确定内核变量的间接可见性;(iv) 在每个原语中解决变量别名问题;(v) 标识在原语间共享的用户进程可见/可修改的变量,消除局部变量;(3) 分析共享变量,并标识隐蔽存储通道.

该方法的主要优点是:

(1) 适用于源代码级的形式化分析,可以发现所有的潜在隐蔽存储通道,并确定强制安全规则是否正确地实现.

(2) 可以发现大量伪非法流.

(3) 可以找出内核共享变量被观察/修改的位置,有助于确定安置审计代码和时间延迟变量的位置.

该方法的缺点是:

(1) 从原语出发构造函数依赖关系集合容易产生状态爆炸,在完成 FCD 的过程中没有退出机制,做了很多无效劳动.

(2) 没有自动工具很难进行手工分析,手工分析不仅工作量大,而且对分析人员的素质要求很高.

(3) 缺乏行之有效的自动工具,且对不同的编程语言需要开发不同的词法分析器和流生成器.

为解决上述问题,He 与 Gligor[14]研制了一种自动工具,可以检查通过 TCB 接口可见的所有信息流,并能区分合法流与非法流.但是,这种工具应用了系统强制安全模型在源代码中的解释,因此不能区分伪非法流与真实非法流.

2.4 无干扰方法

在 Feiertag 等人[15,16]工作的基础上,Goguen 和 Meseguer[17]引入了无干扰方法,它的本质是一个用户不应当知道他不支配的其他任何用户的任何动作.他们将可信计算基 TCB 视为一个状态机,并定义了两个用户进程之间的无干扰概念.假设状态机有一个初始状态,称一个用户进程与另一个用户进程是无干扰的,如果从初始状态开始,删除第 1 个进程所有的输入(等价于从来没有这些输入),第 2 个进程的输出没有任何变化.可以证明,进程之间无干扰具有以下性质[18]:如果一个进程的输入不能影响另一个进程的输出,则不可能从第 1 个进程向第 2 个进程传输信息.

尽管无干扰方法的概念很直观也很简单,但这是一种形式化的分析方法.下面是无干扰的形式化定义.给定一个状态机 TCB,令 X 和 Y 为两个用户进程,w 为一个输入序列,它的结尾是 Y 的输入.令 w/X 表示从 w 中删除所有 X 的输入后剩下的子序列.假设在初始状态输入 w 后,Y 得到的输出为 $Y(w)$.称进程 X 与进程 Y 无干扰,如果对于所有可能的以 Y 的输入为结尾的输入序列 w,都有 $Y(w)=Y(w/X)$.

在应用无干扰方法时,为了避免分析无穷多个输入序列,应当将 TCB 的状态分成不同的等价类.称两个状态是 Y-等价的,如果:(1) 对同一个 Y 输入具有相同的 Y 输出;(2) 对于任何输入,相应的下一个状态也是 Y-等价的.显然,这是一个递归定义.

在实际检验无干扰性时,必须应用下面 Goguen 和 Meseguer[19]给出的展开定理:进程 X 与进程 Y 无干扰,当且仅当对于 X 的任何输入,都使当前状态迁移到一个 Y-等价状态.因此,展开定理使我们能够分析单独的 TCB 函数和原语.只要给出说明 TCB 状态和状态迁移的形式化规范,我们就可以应用展开定理进行无干扰分析.

该方法的主要优点是:

(1) 可以同时应用于形式化顶层规范和源代码;

(2) 可以避免标识伪非法流;

(3) 可以增量分析单个的 TCB 函数和原语.

该方法的缺点是:

(1) 没有支持的自动工具,单纯依靠手工分析不仅工作量大,而且增加人为因素,容易出错;

(2) 该方法是一种“乐观”方法,它要证明的是不存在干扰.因此,该方法适于分析可信进程隔离的 TCB 规范,不适于分析包含大量共享变量的内核;

(3) 在实际应用中,尚无成功的例子.因此,文献[11]将该方法比喻为“皇帝的新衣”,建议目前慎用这种方法.

2.5 小 结

综上所述,目前尚无理论上健壮、实用上行之有效的隐蔽通道标识方法.因此,彻底搜索隐蔽通道仍然是一项困难的任务.困难的程度,依赖于具体的系统和所采用的分析方法.一般地说,系统规模越大,系统越复杂,分析的难度就越高.有的分析方法,例如无干扰方法,虽然是一种理论上严谨的形式化方法,但实际应用却效果不佳.文献[20]报道,同时对 SAT 系统应用 SRM 方法与无干扰方法,SRM 方法除发现无干扰方法找到的隐蔽通道之外,还发现了一个无干扰方法没有找到的隐蔽通道.

我们在安胜高等级安全操作系统的设计过程中,采用了一种新型的“回朔搜索”方法标识隐蔽存储通道,取得了良好效果.对于该方法,我们将另文介绍.

3 隐蔽通道带宽的计算与测量

带宽是隐蔽通道传送数据的速度,单位是比特/秒.带宽的计算或工程测量非常重要,因为隐蔽通道的处理策略依赖于隐蔽通道带宽的确定.

3.1 影响隐蔽通道带宽计算的因素

影响带宽计算的因素很多,其中最重要的有:(1) 噪音与延迟.对于任何操作系统与硬件平台,噪音与延迟都是影响带宽计算的两个最重要的因素.(2) 编码与符号分布.最大可达带宽依赖于符号编码方案的选择.通常假

定,0 和 1 是被传送的符号,且传送 0 与传送 1 所需的时间相同.在这种分布下,离散无记忆通道达到最大带宽,所以上述假定是合理的.(3) TCB 原语的选择.在一个隐蔽通道变量与多个 TCB 原语相关的情形下,应该选择能够达到最大带宽的原语计算带宽.在理想情形下,即只有发送和接收两个进程的条件下计算带宽之后,还必须考虑通道的真实使用场景.否则,计算出的带宽就会偏高或者偏低,影响隐蔽通道的处理策略.(4) 测量与使用场景.需要根据实际应用场景进行以下两类测量:(i) 隐蔽通道涉及的 TCB 原语的性能;(ii) 进程切换时间或上下文切换时间.虽然隐蔽通道的使用场景还包括收发同步,但同步机制是发送者与接收者秘密制定的,所以我们无法预测同步场景.因此,假设同步时间忽略不计.这种假定不影响最大带宽的计算.所有的原语测量和进程切换时间测量必须是可以重复的,使第三方可以验证带宽计算的正确性.(5) 系统配置与初始化.TCB 原语性能与进程切换时间和系统结构参数密切相关,包括:(i) 系统组件的速度,例如 CPU 速度;(ii) 系统配置,例如有无高速缓存;(iii) 组件大小,例如内存大小;(iv) 配置的初始化;(6) 隐蔽通道的聚集.一般地,隐蔽通道的串行聚集和并行聚集都可以增加有效带宽.估计通道聚集对最大带宽影响的最简单的方法是,(a) 对于串行聚集和并行聚集,都设上下文切换时间为 0;(b) 对于并行聚集,求各个隐蔽通道带宽之和.

3.2 隐蔽通道带宽的计算方法

目前,计算带宽的方法主要是 Millen[21]提出的形式化方法和 Tsai 与 Gligor[22]提出的非形式化方法,它们分别适用于不同的场合.

Tsai 与 Gligor 提出了一种 Markov 模型,模拟隐蔽存储通道的使用,并在不同的系统负荷与程序行为的条件下,计算通道的最大带宽.这个模型基于以下基本假设:在隐蔽信息的传输中,0 与 1 的分布相同.对于对称离散无记忆通道,当输入的 0 与 1 概率相等时,通道达到最大容量.对于有限状态通道,当它的状态迁移时间近似相等时,假定通道具有均匀的迁移时间(等于迁移时间的平均值)是一种合理的假设.在这种假设下,可以节省估计最大带宽的工作量.

在只有发送进程和接收进程的情形,计算无噪隐蔽通道最大带宽 $B(0)$的公式十分简单:

$$B(0)=b(T_R+T_S+2T_{CS})-1.$$

其中,b 是编码因子,在实际应用中通常假设为 1.T_R 表示接收进程观察共享变量所需的时间,以及接收进程建立隐蔽通信环境所需的时间.T_S 表示发送进程修改共享变量所需的时间,以及发送进程建立隐蔽通信环境所需的时间.当为一个新进程分配 CPU 时,内核从当前进程向新进程执行一个“上下文切换”操作.T_{CS} 即表示进程切换或上下文切换所需的时间.对于 Secure Xenix 系统,T_{CS}=20ms.

在其他情形下,当通过隐蔽通道传输 0 与传输 1 所需的时间有明显不同时,上述假设就不能成立.为了解决这个问题,提出了其他的编码技术.1964 年,Shannon[23]提出了一种计算具有非均匀迁移时间的有限状态通道的容量的方法.Millen 将这种方法用于使用独立通道变量的有限状态隐蔽通道.在该方法中,为了计算最大可达带宽,假设隐蔽通道是无噪通道,通道工作期间只有接收与发送两个进程,收发同步的时间为 0.然后,将隐蔽通道模拟为有限状态机(图),且这些图都是确定性的.图 1 是一个隐蔽通道的两状态图,它刻画了隐蔽通道传送信息的场景.其中,通过 TCB 原语调用传送 0 和 1 所需的时间不一定相同,因此图中分别用 a,b,c,d 这 4 个时间单元表示不同的传送时间.

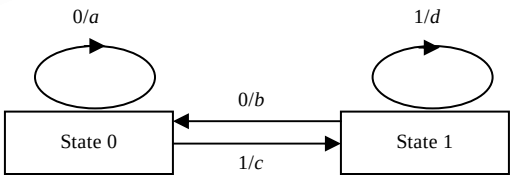

Fig.1 Two-State graph of a covert channel

图 1 隐蔽通道的两状态图

在信息论中,通道的最大传输速率定义为通道容量 C:

$$C=\lim_{t\to\infty}(\log_2 N_i(t))/t \quad (1)$$

$N_i(t)$是从状态 i 开始在时间段 t 内所能传输的符号序列数,且满足下述差分方程组:

$$N_i(t)=\sum_j N_j(t-T_{ij}) \quad (2)$$

其中,T_{ij} 是由状态 i 迁移到 j 所需的时间.

对于图 1 中的两状态情形,方程组(2)变成:

$$N_0(t)=N_0(t-a)+N_1(t-c),$$
$$N_1(t)=N_0(t-b)+N_1(t-d).$$

为了确定通道的容量,我们只需求 $N_i(t)=A_ix^t$ 的渐近上界.将它代入方程组(2),我们得到如下方程组:

$$A_ix^t=\sum_j A_jx^{t-T_{ij}}.$$

上述方程组可以表示成矩阵形式:$(P-I)A=0$,其中,P 是由 x 的负幂组成的矩阵,I 是单位矩阵.$(P-I)$是奇异矩阵,所以它的行列式 $\mathrm{Det}(P-I)=0$.因此,由等式(1)可得:

$$C=\lim_{t\to\infty}(\log_2 A_ix^t)/t=\log_2 x.$$

一般地,x 可以有多个解.我们在计算带宽时,应当选取其中最大的解.

在图 1 的两状态情形,我们有如下的行列式及关于 x 的方程:

$$Det(P-I)\begin{vmatrix} x^{-a}-1 & x^{-c} \\ x^{-b} & x^{-d}-1 \end{vmatrix}=0,$$

$$x-(a+d)-x-a-x-d+1-x-(b+c)=0.$$

1999 年,Shieh[24]对 Millen 的上述工作进行了改进.他指出,仅仅分析独立的通道变量是不够的,因为,很多通道变量具有依赖关系;在两种主要的隐蔽存储通道中,资源耗尽型通道可以模拟为有限状态通道,但事件计数型通道不能模拟为有限状态通道.Shieh 将这类通道称作无限状态通道,并给出计算有限状态通道和无限状态通道带宽的公式以及实现最大可达带宽的编码方法.

3.3 隐蔽通道带宽的测量

理论上计算或估算出的最大带宽必须符合实际.因此,实际测量隐蔽通道的带宽,即测量每个通道真实的最大带宽,是一项十分重要的任务.但是,进行准确测量是十分困难的.因为,我们很难准确测量一个原语的执行时间;而且很难刻画隐蔽通道真实应用的场景.迄今,很少看到关于隐蔽通道成功测量的报道.

在测量隐蔽通道的带宽时,应当遵循如下原则.首先,在有或无出错返回这两种情形,掌握一个原语观察一个变量所需的时间.其次,在测量时,仅选择修改和观察每一个隐蔽通道变量最快的 TCB 原语对.最后,被选择的修改原语和观察原语必须能够合作传送隐蔽信息.注意,当使用 TCB 原语进行隐蔽通道信息传输时,既可能依赖于系统状态,即环境;也可能依赖于调用参数.因此,通道中速度最快的原语不一定被用于数据传输.通常,如果一个原语使用最短的时间观察一个无出错返回的变量.那么,该原语也使用最短的时间观察一个有出错返回的变量.在其他情形,我们需要应用理论计算或估算的结果帮助选择原语.如果原语已经近似地选定,就可以开始测量隐蔽通道真实的最大带宽.

4 隐蔽通道的处理方法

4.1 消除法

根据前言所述各类标准的要求,对已经被标识的隐蔽通道共有 3 种处理方法,即消除法、带宽限制法和审计法.基本原则是,只要有可能,就设法消除隐蔽通道.但在实际应用中,这种方法应用较少,因为有些隐蔽通道根本无法消除,其次消除隐蔽通道往往需要改变系统的设计与实现,例如,消除隐蔽通道可以利用的共享资源;修改隐蔽通道可以利用的接口约定等.因此,有些隐蔽通道尽管可以消除,但消除代价太大.

4.2 带宽限制法

带宽限制法是隐蔽通道处理中最常用的方法,这种策略是事先设定可以接受的阈值,将隐蔽通道的最大带宽或平均带宽降低到阈值以下.通常的作法是:(1) 故意引入噪音;(2) 故意引入延时;(3) 两种方法同时应用.

一种在隐蔽通道中引入噪音的方法是 Hu[25]提出的,他应用了所谓“模糊时间”的概念.安全内核可以限制用户进程只使用虚拟时间,即时间仅与用户进程的动作有关,与系统的真实时间无关[9].该方法的要点是:(1) 降低真实时间与虚拟时间之间的相关性,亦即,使它们之间的关系是随机的;(2) 降低对系统性能的影响.Hu 在 VAX 系统上的实验结果是系统性能降低 5%~6%,可见这是一种实际可行的方法.

Tsai 和 Gligor[22]提出了一种引入冗余进程的方法.将这些用户进程安置在隐蔽通道的发送进程与接收进程之间,可以引入延时.使这些用户进程随机修改隐蔽通道变量,可以引入噪音.实验证明,这种方法可以使典型的隐蔽通道带宽降低 75%,但在降低隐蔽通道带宽的同时,也会降低系统的性能.

在实际应用中,需要考虑许多复杂的因素.例如,从系统性能的角度考虑,在 TCB 原语中引入延时,主要应用于资源耗尽型通道.因为,资源耗尽型通道利用资源耗尽异常传送隐蔽信息,而在正常情形资源耗尽异常出现的频率很低.因此,将延时安置在资源耗尽异常返回的路径上就可以降低通道带宽,同时不影响系统性能.此外,在 TCB 原语中的合适位置安置延时,也是一项重要和困难的任务,需要在各个方面进行权衡与折衷.

4.3 审计法

审计法是一种威慑方法,它的目的是无二义性地检测隐蔽通道的应用,监控系统中已知隐蔽通道的使用情况.对审计机制的基本要求只有一条:不错报.首先,保证审计机制不被旁路,即不漏报;其次,保证准确审计,即不误报.事实上,这个要求很难达到.审计的固有困难性表现在:(1) 很难区分 TCB 原语的正常应用与非正常应用(产生隐蔽通道);(2) 很难区分隐蔽通道中的发送进程与接收进程.甚至,有些隐蔽通道是无法进行审计的.更详细的论述,请参见文献[26,27].

5 总结与展望

隐蔽通道构成实际的威胁,因此,对高安全等级的操作系统,各国的各类标准都要求进行隐蔽通道分析与处理.但是,隐蔽通道分析,特别是隐蔽通道标识是一个众所周知的困难问题,也是一个不断发展中的课题.在这一领域,理论尚不成熟,实际工作量大.我们在安胜高安全等级安全操作系统的隐蔽通道分析中,通过一种新型的“回朔搜索方法”,分析了数十万行源代码,数百个系统调用与数百个共享变量.不仅如此,连实现隐蔽通道和利用隐蔽通道都是困难的.为此,必须在系统中嵌入一个木马程序,搜集敏感信息,将它编码,与一个具有较低安全级的接收程序合作,然后在一个较长的时间段泄漏这些敏感信息,而且不能触发系统的审计报警机制.

为了消除隐蔽通道,往往需要重新设计操作系统.近年来,这方面的成果除本文已经提及的模糊时间技术外,还有模糊传送技术、数据泵技术等新技术不断出现.在搜索隐蔽通道方面,开发了一些信息流分析工具,可以减少人工干预,以发现一些难以直观上察觉的问题.例如,MITRE 流分析器,Gypsy 流分析器,Ina Flo/MLS,SRI HDM 流分析器等.

总之,隐蔽通道分析仍然是一个具有挑战性的课题,特别是迄今为止,尚无系统地标识隐蔽定时通道的方法.在隐蔽通道的研究中,还有许多公开问题等待我们解决.例如,隐蔽通道标识的无干扰方法,理论上是较为成熟的,但它基于抽象的自动机.在实际应用中,必须依靠“展开定理”.但是在应用展开定理时,关键是找到一个合适的“view”函数,如果不能找到一个最合适的 view 函数,则会产生大量伪流,甚至找不到真实隐蔽通道,从而使普遍认可的无干扰模型成为“皇帝的新衣”.我们能否另辟捷径,使无干扰模型可以真正应用于隐蔽通道标识呢?

致谢 作者感谢与朱继锋博士所作的有益讨论.

References:

[1] Lampson BW. A note on the confinement problem. CACM, 1973,16(10).

[2] Qing SH, Liu WQ, Liu HF. Introduction to Operation System Security. Beijing: Science Press, 2003 (in Chinese).

[3] NCSC. A guide to understanding covert channel analysis of trusted system, NCSC-TG-030, 1993.

[4] U.S. Department of Defense. Trusted Computer System Evaluation Criteria, DoD 5200.28-STD. 1985.

[5] Schaefer M, Gold B, Linde R, Scheid J. Program confinement in KVM/370. In: Proc. of the 1977 Annual ACM Conf. New York: ACM, 1977. 404~410.

[6] Huskamp JC. Covert communication channels in timesharing systems. Technical Report UCB-CS-78-02, 1978.

[7] Kemmerer RA. Shared resource matrix methodology: An approach to identifying storage and timing channels. ACM Trans. on Computer Systems, 1983. 256~277.

[8] Tsai CR, Gligor VD, Chandersekaran CS. A formal method for the identification of covert storage channels in source code. IEEE Trans. on Software Engineering, 1990. 569~580.

[9] Lipner SB. A comment on the confinement problem. Operating Systems Review, 1975. 192~196.

[10] Porras PA, Kemmerer RA. Covert flow trees: A technique for identifying and analyzing covert storage channels. In: Proc. of the 1991 IEEE Computer Society Symp. on Research in Security and Privacy. 1991. 36~51.

[11] McHugh J. Covert channel analysis: A chapter of the handbook for the computer security certification of trusted system. NRL Technical Memorandum 5540:062A, 1995.

[12] Kemmerer RA, Taylor TA. Modular covert channel analysis methodology for trusted DG/UXTM. IEEE Trans. on Software Engineering, Vol. 22, 1996.

[13] Denning DE. A lattice model of secure information flow. Communications of the ACM, 1976. 236~243.

[14] He J, Gligor VD. Information flow analysis for covert-channel identification in multilevel secure operating systems. In: Proc. of the 3rd IEEE Workshop on Computer Security Foundations. 1990. 139~148.

[15] Feiertag R. A technique for proving specifications are multilevel secure. Technical Report CSL-109, 1980.

[16] Feiertag R, Levitt KN, Robinson L. Proving multilevel security of a system design. In: Proc. of the 6th ACM Symp. on Operating Systems Principles. 1977. 57~65.

[17] Goguen JA, Meseguer J. Security policies and security models. In: Proc. of the IEEE Symp. on Security and Privacy. 1982. 11~20.

[18] Millen JK. Foundations of covert-channel detection. Technical Report MTR-10538, The MlTRE Corporation, 1989.

[19] Goguen JA, Meseguer J. Unwinding and inference vontrol. In: Proc. of the IEEE Symp. on Security and Privacy. 1984. 75~86.

[20] Haigh JT, Kemmerer RA, McHugh J, Young WD. An experience using two covert channel analysis techniques on a real system design. IEEE Trans. on Software Engineering, 1987. 157~168.

[21] Millen JK. Finite-State noiseless covert channels. In: Proc. of the Computer Security Foundations Workshop. 1989. 81~85.

[22] Tsai CR, Gligor VD. A bandwidth computation model for covert storage channels and its applications. In: Proc. of the IEEE Symp. on Security and Privacy. 1988. 108~121.

[23] Shannon CE, Weaver W. The Mathematical Theory of Communication. Urbana: The University of Illinois Press, 1964.

[24] Shieh SP. Estimating and measuring covert channels bandwidth in multivlevel secure operating system. Journal of Information Science and Engineering, 1999,15:91~106.

[25] Hu WM. Reducing timing channels with fuzzy time. In: Proc. of the IEEE Symp. on Research in Security and Privacy. 1991. 8~20.

[26] Shieh SP, Gligor VD. Auditing the use of covert channels in secure systems. In: Proc. of the IEEE Symp. on Research in Security and Privacy. 1990.

[27] Qing SH, Liu WQ, Weng HZ, Liu HF. Operation System Security. Beijing: Tsinghua University Press, 2004 (in Chinese).

附中文参考文献:

[2] 卿斯汉,刘文清,刘海峰.操作系统安全导论.北京:科学出版社,2003.

[27] 卿斯汉,刘文清,温红子,刘海峰.操作系统安全.北京:清华大学出版社,2004.

见证中国软件科学从弱到强

——《软件学报》举办创刊十五周年学术报告会

原载《科学时报》2004 年 9 月 17 日第 4 版

本报北京 9 月 16 日讯 （记者 王学健）伴随着中国计算机科学和软件科学发展过程中的风风雨雨，中国科学院软件研究所和中国计算机学会联合主办的权威刊物《软件学报》迎来了创刊 15 周年的喜庆日子，并以举办高水平学术报告会的方式庆贺生日。中国科学院、科技部、教育部、国家自然科学基金委员会、中国科协的有关领导出席了学术报告会，很多专家都对《软件学报》给予高度评价。中国科学院软件研究所所长李明树表示，《软件学报》见证了中国软件科学从弱到强的历史进程。该刊物副主编之一、美籍华人、有世界计算机科学“诺贝尔奖”之称的图灵奖得主、美国普林斯顿大学教授姚期智，在报告会上作了《理论计算学的方向》学术报告。

创刊于 1990 年的高级学术刊物《软件学报》，在短短的 15 年中，以锐不可当的势头跻身于全国优秀学术期刊的行列，赢得了计算机领域研究人员、工程技术人员、软件开发及软件应用人员、大专院校的教师及博士生等的普遍青睐，成为国内、国际计算机界公认的重要学术期刊。《软件学报》被高等学校与科研院所学位与研究生教育评估所评为一级学科期刊。

厚达 160 页的《软件学报》现为月刊，主要刊登国际、国内前沿基础性科研成果的文章及国家重大基金项目，如国家自然科学基金、国家重大基础研究攀登计划基金和国家“973”、“863”高科技项目基金的文章等。该刊物曾在 2001 年入选中国期刊方阵“双百期刊”，2001 年和 2002 年分别荣获百种中国杰出学术期刊。目前，该刊物被国际和国内众多数据库收录，如 EI Compendex 全文摘要数据库 美国;SA 英国;COMPUAB 英国;中国科技论文统计与分析年度研究报告;“中国学术期刊综合评价数据库”的来源期刊;中文核心期刊要目总览中的核心期刊等等。