基于无干扰模型的操作系统结构化研究

孙 瑜[1,3] 胡 俊[1] 陈亚莎[2] 张 兴[1]
(北京工业大学计算机学院 北京 100022)[1] (海军工程大学信息与电气工程学院 武汉 430033)[2]
(信息网络安全公安部重点实验室 上海 201204)[3]

摘 要 操作系统结构化是目前安全领域的一大难题。以无干扰模型为基础,提出了一种基于分层隔离的进程环境安全模型,给出了进程环境安全的定义和条件。然后对系统结构化要求进行了形式化的描述,并证明通过提出的结构化方法可以获得安全的进程环境。最后结合经典无干扰理论,将本模型由进程环境扩展为适用于整个系统安全的模型。

关键词 操作系统,结构化,无干扰,进程环境
中图法分类号 TP309 **文献标识码** A

Operating System Structuralization Research Based on Non-interference Model

SUN Yu[1,3] HU Jun[1] CHEN Ya-sha[2] ZHANG Xing[1]
(Computer Science Department, Beijing University of Technology, Beijing 100022, China)[1]
(College of Information and Electrical Engineering, Naval University of Engineering, Wuhan 430033, China)[2]
(Key Lab of Information Network Security, Ministry of Public Security, Shanghai 201204, China)[3]

Abstract Structural protection of operating system is currently a difficult problem in security field. This paper studied a process execution environment model based on hierarchy isolation, and gave definition and conditions of process environment security. Then we formalized structural requirements of process environment, and proved that a secure process environment can be available by structured method proposed. Finally combined with classic non-interference theory, the security model was extended from process environment to the whole system.

Keywords Operating system, Structural protection, Non-interference, Process environment

1 引言

操作系统对下层而言是计算机资源的管理者,对上层而言是用户虚拟运行环境的提供者,对整个计算机系统的构建和运行具有承上启下的作用,是计算机软件运行的基础和核心。可以说操作系统安全是整个计算机系统安全的基础,没有操作系统的安全,就谈不上建立于其上的数据安全和应用软件的安全。所以,安全操作系统一直是信息安全研究的重要方向。

高安全级操作系统不是安全功能的简单叠加,必须要有严密科学的结构加以保证,要从操作系统体系结构的层次对安全保障机制进行研究。TCSEC B2[1]级从系统体系结构层面明确提出要实现结构化的保障机制,即强调关键保护元素和非关键保护元素的分离,加强进程的隔离。国内 GB17859 也要求四级系统必须实现结构化的保障。可以说,结构化是高安全级操作系统(B2 级以上)的一个本质特征,对其进行深入的研究具有极为重要的意义。

本文借鉴无干扰模型,提出了一种分层隔离的进程环境安全模型,给出了进程环境安全的定义和条件。然后对进程环境结构化要求进行了形式化的描述,并证明通过本文提出的结构化方法可以获得安全的进程环境。最后将本模型由进程环境扩展为适用于整个系统安全的模型。

2 结构化及无干扰理论概述研究

结构化的概念首先是从以往编程过程中无限制地使用转移语句而提出的,转移语句可以使程序的控制流程强制性地转移向程序的任一处,这将会导致流程无序可循,结构杂乱无章,这样的程序很难获得稳定性和可靠性。因此系统结构化就是要消除随意的转移语句。在 20 世纪 60 年代末,Dijkstra 等人[2]提出,所有程序都可以建立在一组固定的逻辑构成元素上。当前的操作系统无论是 Windows 还是 Linux,都没有实现体系结构层次的结构化,模块或组件内部的结构化可以通过编程的手段来解决,但是体系结构层次的结构化才是操作系统的最大问题,模块或组件之间的调用不受限制,调用可以任意序列,这样给系统安全性带来了很大的问题,如果操作系统没有实现结构化,其上的所有安全机制也就失去了保障

到稿日期:2010-05-30 返修日期:2010-07-30 本文受 973 国家重点基础研究计划(2007CB311100),国家 863 基金项目(2009AA01Z437)信息网络安全公安部重点实验室开放基金资助课题资助。

孙 瑜 男,博士生,主要研究方向为安全操作系统、可信计算,E-mail:syking1@163.com;胡 俊(1972—),男,博士,讲师,主要研究方向为全操作系统、可信计算;陈亚莎 女,博士生,主要研究方向为安全系统、可信计算;;张 兴 男,教授,主要研究方向为信息安全。

从而成为了空中楼阁。

1982年J. A. Gogeun和J. Meseguer最早提出无干扰思想[3]，首先建立了基于用户、状态、命令、输出等元素的系统模型。J. T. Haigh, W. D. Young在文献[4]中首先用无干扰的语义解释了访问控制以及MLS和MDS，并研究了其在实际系统SAT中的实现。1992年，Rushby提出了采用状态机的无干扰模型[5]，首先引入了域的概念，系统地描述了无干扰对访问控制和通道控制的解释，并给出了系统关于传递和非传递无干扰策略安全的定义。

J. A. Gogeun和J. Meseguer的无干扰模型中针对的是用户之间的干扰，但是在实际的系统中很难区分用户的边界，且系统中实际的主体为进程。Rushby无干扰模型引用了域的概念，但是并没有作为一个实体对域进行必要的描述。对于操作系统来说，每一个程序都运行于一个虚拟环境，也即进程环境。文献[6,7]都提出了以广义的进程概念作为域来建立安全模型，以解决进程之间的干扰问题。但是在实际的系统中，无法保证整个进程环境都处于相同的安全状态下。例如：通常通用操作系统(Windows或Linux)内核空间由所有进程共享，内部缺乏必要的隔离机制，违背了Rushby无干扰的基本定义分区观察(view-partitioned)。问题在于进程环境的内核态权限过大，可以观察甚至修改任意信息(包括机器状态、系统状态、其他进程状态等)，缺乏结构化控制，以至于无法有效划分无干扰域。

3 基于分层隔离的进程环境安全模型

结构化本质上是为了解决系统内部的非法指令转移，而无干扰理论为指令执行产生的域间干扰问题建立了一个模型，并给出了模型中系统的安全条件，如果把非法指令转移看作一种干扰，就可以将无干扰理论引入到系统结构化问题中来。系统的运行实际是进程的运行，系统的结构化首先就是进程环境的结构化。本节沿用Rushby对无干扰模型的基本描述方法[5]，给出了一个基于分层隔离的进程环境安全模型。

定义1 操作系统中应消除全局变量，对跨模块的访问首先应为对过程的调用，因此进程环境中对数据流的控制也可以通过控制流的控制来完成，可只考虑控制流干扰。同时作为一个动作处理，将返回作为调用的后续动作，因此只考虑转移和顺序$\{shift, seq\}$这两种指令类型。

定义2 系统$M=\{S, I, EV, IT, SD, F, s_0\}$，其中，

S：系统状态集合，包含系统初态$s_0 \in S$，元素用小写字母$s, t, \cdots$表示；

I：系统指令集合，用$i, j, \cdots$表示指令集中的元素，Γ为指令子集，类型为$shift$，$\Gamma \subseteq I$；

EV：系统状态在值域的投影，可以映射为当前状态下观察到的环境输出；

IT：系统指令类型，$IT=\{shift, seq\}$；分别表示指令转移和顺序执行；

SD：安全域集合，$SD=PE\times L$；

PE：进程执行环境集合，表示为$(p_1, p_2, \cdots, p_n)$；

L为进程环境内部分层集合，表示为$(l_0, l_1, l_2, \cdots, l_n)$，通常将TCB作为$l_0$。

系统M定义了如下函数：

F1 $step: S\times I \to S$表示系统执行一个指令后的状态迁移；

F2 $envval: S\times I \to EV$表示系统执行一个指令后观察到的输出；

F3 $run: S\times I^* \to S$表示执行完一个指令序列后系统的状态，该函数具有如下性质：

$run(s, \varepsilon)=s$，$run(s, a\circ\alpha)=run(step(s, a), \alpha)$

F4 $valid: S\times I \to \{\text{true}, \text{false}\}$表示状态$s$下是否可以执行指令$I$；

F5 $type: I \to IT$表示指令的类型；

F6 $srcdom: I \to SD$，表示执行指令的发起域；

F7 $level: SD \to N$，表示安全域所在的层次，N为整数，对应于$(l_0, l_1, l_2, l_3, l_4)$为$(0,1,2,3,4)$；

F8 $proc: SD \to PE$，表示安全域所在的进程环境；

F9 $arbit: I\times SD \to \{\text{true}, \text{false}\}$仲裁指令是否可以域切换。

$\forall i\in I, \forall u\in SD$有：

$$arbit(i,u)=\begin{cases}\text{true}, & \text{iff}((type(i)=shift)\wedge level(srcdom(i))\geqslant level(u))\vee(type(i)=seq)\vee(srcdom(i)=u)\\ \text{false}, & \text{otherwise}\end{cases}$$

F10 $call: I \to P(SD)\ \forall i\in I, type(i)\neq seq$定义为指令可以切换的安全域集。

定义3 定义安全域间安全策略$\sim$，$u\sim v\ u, v\in SD$，表示u可以干扰v，即存在从u到v的信息流。反之，$\neg(u\sim v)$表示u无法干扰v。

定义4 函数$origin: I^*\times SD \to P(SD)$表示能执行指令$I$的安全域集，

$origin(\varepsilon, u)=\{u\}$

$origin(a\circ\alpha, u)=$

$$\begin{cases}origin(\alpha,u)\cup\{srcdom(a)\}, & if\ \exists v: v\in origin(\alpha,u)\wedge srcdom(a)\sim v\\ origin(\alpha,u), & \text{otherwise}\end{cases}$$

函数$clear: I^*\times SD \to I*$，$\alpha\in I*$，$v\in SD$，

$clear(\varepsilon, v)=\varepsilon$

$clear(a\circ\alpha, v)=$

$$\begin{cases}a.clear(\alpha,v), & \text{if } srcdom(a)\in origin(a\circ\alpha, v)\\ clear(\alpha,v), & \text{otherwise}\end{cases}$$

式中，$clear(\alpha, u)$表示从α中删除了所有不能干扰u的域的操作的一个子序列(包括直接干扰和间接干扰，即传递和非传递两种情况)。该安全定义表示，若在系统的执行序列中去除所有不能干扰u域的指令后，对u执行的操作不产生任何影响，则系统对该无干扰策略是安全的。

定义5 系统M是可分区观察的，对于$\forall u\in SD$，存在关于S的等价关系$\overset{u}{\approx}$，即域之间是隔离的，任意域只能观察当前域环境的状态。例如：$s \overset{(p,l)}{\approx} t, s, t\in S, p\in P, l\in L$；表示在$s$，$t$状态下，域$(p, l)$观察到系统的环境输出是一致的。

定义6 进程执行环境对于策略$\sim$是安全的，当满足条件：

$$envval(run(s_0,\alpha),a)=envval(run(s_0, clear(\alpha, srcdom(a))),a)$$

即进程环境的安全定义为：进程环境分层隔离，域间不存在非

法的控制流。

为便于系统验证，定义 7 和定义 8 给出了只涉及单步状态的进程环境安全展开条件。

定义 7 设 $s,t\in S,i\in I$，称进程环境满足观察隔离性：当系统可分区观察且

$$s \overset{srcdom(i)}{\approx} t \rightarrow envval(s,i)=envval(t,i)$$

进程环境状态等价表示安全域所能观察到的状态是一致的，那么该域执行一条指令之后观察到的输出一致。

定义 8 设 $s,t\in S,i\in I$，称进程环境满足单步隔离性：当系统可分区观察且

1. $s \overset{u}{\approx} t \wedge s \overset{srcdom(i)}{\approx} t \wedge valid(s,i) \wedge valid(t,i) \rightarrow step(s,i) \overset{u}{\approx} step(t,i)$

2. $s \overset{u}{\approx} t \wedge s \overset{srcdom(i)}{\approx} t \wedge valid(s,i) \wedge \neg valid(t,i) \rightarrow \neg(srcdom(i) \sim u)$

3. $\neg(srcdom(i) \sim u) \rightarrow s \overset{u}{\approx} step(s,i)$

单步隔离性表示在非传递情况下，满足一定条件执行单步指令操作后依然可以保持状态等价。

根据定义 7 和定义 8 可得出如下定理 1，证明了满足观察隔离性和单步隔离性可以获得安全的进程环境。

定理 1 一个进程执行环境若满足：初始状态 s_0 安全，观察隔离性和单步隔离性；则该进程执行环境对于策略～是安全的。

证明：

由观察隔离性可知：

由 $run(s_0,\alpha) \overset{srcdom(\alpha)}{\approx} run(s_0,clear(\alpha,srcdom(\alpha)))$ 可以直接推导出：

$envval(run(s_0,\alpha),a)=envval(run(s_0,clear(\alpha,srcdom(\alpha))),a)$

因此只需证明：

$$run(s_0,\alpha) \overset{srcdom(\alpha)}{\approx} run(s_0,clear(\alpha,srcdom(\alpha)))$$

即 $run(s_0,\alpha) \overset{u}{\approx} run(s_0,clear(\alpha,u))$；

假设当 $\alpha=\varepsilon$ 时下列命题成立，假设：$s,t\in S$

$$s \overset{origin(\alpha,u)}{\approx} t \wedge i\circ\alpha\in I^* \rightarrow run(s,\alpha) \overset{u}{\approx} run(t,clear(\alpha,u))$$

对 α 长度进行归纳证明：当 $\alpha=\varepsilon$ 时，命题显然成立。

现证明当 $i\circ\alpha$ 时命题也成立。即假设：

$$s \overset{origin(i\circ\alpha,u)}{\approx} t \wedge i\circ\alpha\in I^* \rightarrow run(s,i\circ\alpha) \overset{u}{\approx} run(t,clear(i\circ\alpha,u))$$

证明 $run(s,i\circ\alpha) \overset{u}{\approx} run(t,clear(i\circ\alpha,u))$

对 i 分情况讨论：

情况 1 假设 $srcdom(i)\in origin(i\circ\alpha,u)$

$origin(i\circ\alpha,u)=i\circ origin(\alpha,u)$

需证：$run(step(s,i),\alpha) \overset{u}{\approx} run(step(t,i),clear(\alpha,u))$

设 $\forall v\in origin(\alpha,u)$，易知 $v\in origin(i\circ\alpha,u)$ 且 $s \overset{v}{\approx} t$；

当 $srcdom(i)\sim v$：由定义 $srcdom(i)\in origin(i\circ\alpha)$

由假设可知：$s \overset{srcdom(i)}{\approx} t$；

若 $valid(s,i) \wedge valid(t,i)=$ true，由定义 8 的条件 1 可得出：

$$step(s,i) \overset{v}{\approx} step(t,i)$$

若 $valid(s,i) \wedge valid(t,i)=$ false，由定义 8 的条件 1 可得出 $\neg(srcdom(i)\sim v)$，与假设矛盾。

令 $v=u$，由归纳假设：$run(step(s,i),\alpha) \overset{u}{\approx} run(step(t,i),clear(\alpha,u))$，证毕；

当 $\neg(srcdom(i)\sim v)$，由定义 8 的条件 4 可得：

$$s \overset{v}{\approx} step(s,i), t \overset{v}{\approx} step(t,i)$$

由 $s \overset{v}{\approx} t$；且 $\approx$ 为等价关系，$\forall valid(s,i) \wedge valid(t,i)$ 得：$step(s,i) \overset{v}{\approx} step(t,i)$；同理证毕；

情况 2 假设 $srcdom(i)\notin origin(i\circ\alpha,u)$

$origin(i\circ\alpha,u)=origin(\alpha,u)$

需证：$run(step(s,i),\alpha) \overset{u}{\approx} run(t,clear(\alpha,u))$；

由 $srcdom(i)\notin origin(i\circ\alpha,u) \rightarrow \neg(srcdom(i)\sim u)$；

由定义 8 的条件 3 得：$s \overset{u}{\approx} step(s,i)$

由假设 $s \overset{origin(i\circ\alpha,u)}{\approx} t$ $u\in origin(i\circ\alpha,u)$ 且 $\approx$ 为等价关系，得：$t \overset{u}{\approx} step(s,i)$

归纳假设得证 $run(step(s,i),\alpha) \overset{u}{\approx} run(t,clear(\alpha,u))$；

令 $s=t=0$，得 $run(s_0,\alpha) \overset{u}{\approx} run(s_0,clear(\alpha,u))$；证毕。

本节根据无干扰的基本原理，建立了一个分层隔离的进程环境安全模型，提出了获得安全进程环境的充分条件。但是模型的抽象性决定了很难将安全条件直接用于指导系统结构化的实现，下节我们将提出一种针对进程环境的结构化方法，同时证明进程环境满足安全充分条件。

4 进程环境安全模型的结构化解释

以上我们已经证明了进程执行环境对于策略～安全的展开条件，并且对于操作系统结构化的基本含义也已进行了描述，建立了进程环境安全的模型。下面我们将以形式化的方式来定义结构化对于分层隔离的进程环境所要求满足的特性，然后进一步证明由此可以获得安全的进程环境。

定义 9 描述了进程环境结构化应满足的 3 个基本规则，并给出了形式化的描述。

定义 9 对于进程环境，结构化要求指令执行具有单向性、内部一致性、单步确定性。内部一致性即假设 1 表示没有跨域转移时，单步指令执行后的状态对指令发起域来说保持一致。单向性即假设 2 表示进程环境域间只能进行有序分层调用。单步确定性即假设 3 表示进程环境中域间调用应通过统一接口，且执行过程是确定的。

即满足以下假设：

1. $contents(s,n)=contents(t,n)|_{srcdom(i)} \rightarrow contents(step(s,i),n)=contents(step(t,i),n)|_{srcdom(i)}$

2. $\exists i\in I, srcdom(i)=u, v\in call(i) \rightarrow u\sim v \forall v\in call(i) \rightarrow arbit(i,v)=$ true

3. $contents(s,n)=contents(t,n)|_u \wedge u\in call(i) \wedge (level(srcdom(i))=level(u)+1) \rightarrow contents(step(s,i),n)=contents(step(t,i),n)|_u$

4. $contents(step(s,i),n)\neq contents(s,n)|_u \rightarrow \exists i\in I, u\in call(i)$

其中，n 表示进程执行环境中所有的状态值，包括上下文、堆栈、进程状态、机器状态等。$contents(s,n)|_u$ 表示进程在状态

s 下 u 观察到 n 的值。

根据定义 9，定理 2 证明了结构化假设可以满足进程环境安全的充分条件。

定理 2 一个进程环境若初态 s_0 安全且满足结构化假设，则进程环境对于策略～是安全的。

证明：设 $\forall s,t\in S,i\in I,u\in SD$

由等价关系$\overset{u}{\approx}$的定义可得：

$s\overset{u}{\approx}t\rightarrow contents(s,n)=contents(t,n)|_u$

根据结构化假设 1，易知：

$s\overset{srcdom(i)}{\approx}t\rightarrow contents(step(s,i),n)=contents(step(t,i),n)$

根据观察隔离性定义，得：

$s\overset{srcdom(i)}{\approx}t\rightarrow envval(s,i)=envval(t,i)$

证明：$\neg(srcdom(i)\sim u)\rightarrow s\overset{u}{\approx}step(s,i)$

反证法：假设$\neg s\overset{u}{\approx}step(s,i)$

$contents(step(s,i),n)\neq contents(s,n)|_u$

由结构化假设条件 4 得：

$u\in call(i)$

由结构化假设条件 2 得：

$u\in call(i)\rightarrow srcdom(i)\sim u$

这与假设$\neg(srcdom(i)\sim u)$矛盾。证毕。

证明：$s\overset{u}{\approx}t\wedge s\overset{srcdom(i)}{\approx}t\wedge valid(s,i)\wedge valid(t,i)\rightarrow step(s,i)\overset{u}{\approx}step(t,i)$

情况 1 $\neg(srcdom(i)\sim u)$

设 $valid(s,i)\wedge valid(t,i)=$true，得：

$s\overset{u}{\approx}step(s,i),t\overset{u}{\approx}step(t,i)$

由等价关系定义得：$step(s,i)\overset{u}{\approx}step(t,i)$。证毕。

情况 2 $(srcdom(i)\sim u)$

设 $valid(s,i)\wedge valid(t,i)=$true，若

$valid(s,i)\wedge valid(t,i)=$false

则$\neg(srcdom(i)\sim u)$得证。

需证：

$s\overset{u}{\approx}t\wedge s\overset{srcdom(i)}{\approx}t\rightarrow contents(step(s,i),n)=contents(step(t,i),n)|_u$

若 $srcdom(i)=u$，由结构化假设条件 1 易得命题成立。

若 $srcdom(i)\neq u$

情况 1 设 $contents(step(s,i),n)\neq contents(s,n)|_u$

$u\in call(i)$

由假设可知 i 直接改变了执行环境状态值，得：

$(type(i)=shift)\wedge level(srcdom(i))=level(u)+1)=$true

由结构化假设条件 3，命题得证。

情况 2 $contents(step(t,i),n)\neq contents(t,n)|_u$和情况 1 证法相同。

情况 3 $contents(step(s,i),n)=contents(s,n)|_u\wedge contents(step(t,i),n)=contents(t,n)|_u$

由定义 7 得 $s\overset{u}{\approx}t\rightarrow contents(s,n)=contents(t,n)|_u$

易证命题。证毕。

定理 3 在定理 2 和基本无干扰模型的基础上，证明了可以将进程环境安全进一步推广至整个系统的安全。因篇幅有限，证明过程略。

定理 3 若系统 M 满足：系统 M 初态 s_0 是安全状态；不同进程环境之间不存在指令转移；即 $\forall i\in I,\forall v\in SD,\exists v\in call(i)\rightarrow proc(srcdom(i))=proc(v)$；进程执行环境满足结构化假设；进程之间满足无干扰策略～；那么系统 M 对于策略～是安全的。

结束语 自 20 世纪 70 年代起，Denning，Bell，Lapadula 和 Biba[8-10] 等人对安全模型进行了大量的基础研究，特别是美国提出可信计算机评估标准 TCSEC 以来，系统安全模型得到了广泛的研究，并在各种操作系统中实现了多种安全模型。然而，大部分的研究都是集中在访问控制及其他安全功能上，而对如何保障这些机制在操作系统中正确实施以及系统自身的可靠性方面却涉及甚少。本文提出基于分层隔离的进程环境安全模型，并对基于模型的结构化保障方法进行了初步的研究。在以后的工作中，我们将进一步研究进程环境之间的隔离，同时解决在模型具体实现中所遇到的各种问题。对于一些必须存在的非结构化转移，我们要运用可信计算和密码学技术研究一种安全合理的方法（例如管道）来加以解决。通过系统结构化，对于针对内核 rootkit 的攻击要加以分析，并且能在一定程度上解决目前系统所存在的隐通道问题。

参 考 文 献

[1] Narayanan S, Mcllraith S. Simulation, verification and automated composition of Web services[C]// Proc. WWW'02. ACM, 2002: 77-88

[2] DoD 5200. 28-STD[S]. Department of Defense Standard. DoD Trusted Computer System Evaluation Criteria(orange). Meade, MD, USA. National Computer Security Center, Ft. Dec. 1985

[3] Dijkstra E W. Hierarchical Ordering of Sequential Processes[Z]. Operating System Techniques, 1972

[4] Goguen J A, Meseguer J. Security policies and security models [C]// Proc. of the 1982 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, April 1982: 11-20

[5] Haigh J T, Yong W D. Extending the noninterference model of MLS for SAT[A]// Proceedings of the Symposium on Security and Privacy[C]. Oakland, CA, 1986: 232-239

[6] Rushby J. Noninterference, transitivity, and channel-control security policies[R]. CSL-92-02. Menlo Park: Stanford Research Institute, 1992

[7] 赵佳. 基于无干扰理论的可信链模型[J]. 计算机研究与发展, 2008, 45(6): 974-980

[8] 张兴. 基于进程的无干扰可信模型[J]. 通信学报, 2009, 30(3): 6-11

[9] Denning D E. A lattice model of secure information flow[J]. Commu. ACM, 1976, 19(5): 236-243

[10] Bell D E, La Padula L J. Secure Computer System: Unified Exposition and MUTICS Interpretation[R]. MTR-2997, AD-A 023 588. July 1975

[11] Biba K J. Integrity considerations for secure computer systems [R]. MTR 3153. The Mitre Corporation, April 1977