ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.3, March 2007, pp.730−738
DOI: 10.1360/jos180730

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一个支持可信主体特权最小化的多级安全模型*

武延军[1,2+], 梁洪亮[1], 赵 琛[1]

[1](中国科学院 软件研究所 基础软件国家工程研究中心,北京 100080)

[2](中国科学院 研究生院,北京 100049)

A Multi-Level Security Model with Least Privilege Support for Trusted Subject

WU Yan-Jun[1,2+], LIANG Hong-Liang[1], ZHAO Chen[1]

[1](National Engineer and Research Center for Fundamental Software, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: Phn: +86-10-62561197 ext 1010, E-mail: wuyanjun@gmail.com, http://www.ios.ac.cn

Wu YJ, Liang HL, Zhao C. A multi-level security model with least privilege support for trusted subject. *Journal of Software*, 2007,18(3):730−738. http://www.jos.org.cn/1000-9825/18/730.htm

Abstract: The trusted subject supports of the existing multi-level security models are reviewed and a new model called DLS (discrete label sequence) is proposed. It decomposes the lifecycle of a trusted subject into a sequence of untrusted states (US). Each untrusted state is associated with a certain current security label, and only the predefined trusted request events (TRE) can trigger the transition from one US to the other. Thus, the current security level of a trusted subject is dynamically changed according to its application's logic. Definitions of secure states and rules to preserve security are also presented. Compared with the trusted subject implemented by security level range, this model gives a better support of least privilege and achieves the support within the MLS policy framework.
Key words: multi-level security; trusted subject; least privilege; operating system security

摘 要: 对已有多级安全模型的可信主体支持进行回顾和分析,提出了 DLS(离散标记序列)多级安全模型.该模型将可信主体的生命周期分解为一系列非可信状态,对每一个状态赋予一个敏感标记.可信主体的当前敏感标记等于当前非可信状态的敏感标记,非可信状态的切换由预定义的可信请求事件触发.从而可信主体的当前敏感标记可以根据其应用逻辑而动态调整.同时给出了模型保持系统安全性的安全状态和规则.与 Bell 模型等可信主体敏感标记范围模型相比,该模型在多级安全的策略范围内实现了可信主体的特权最小化.
关键词: 多级安全;可信主体;最小特权;操作系统安全
中图法分类号: TP309 文献标识码: A

经典 Bell & Lapadula(BLP)模型[1]来源于具有严格保密需求的军事领域,被公认为保密性安全系统的基本安全模型,也是许多信息系统安全评测标准的制定依据和理论基础(如美国国防部的 TCSEC、中国国家标准

* Supported by the National Natural Science Foundation of China under Grant No.60373054 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA141080 (国家高技术研究发展计划(863))

Received 2005-11-28; Accepted 2006-03-17

GB17859-1999),几乎在迄今为止的所有安全操作系统中得到了实施.在基于 BLP 模型的多级安全(multi-level security,简称 MLS)系统中,主体不能读取敏感标记高于其敏感标记的客体,不能写入敏感标记低于其敏感标记的客体,从而防止机密信息的泄漏.但是,如果严格实施*-属性,几乎不可能构造出一个可用的计算机系统,由此引入了可信主体(trusted subject)的概念.Bell 在文献[2]中将可信主体定义为不受*-属性限制的主体,并给出鉴定可信主体的两个标准:

- 主体对客体的访问违反*-属性,但是这种访问在系统中是必需的;
- 可以证明该主体不会引起*-属性所禁止的信息流动.

上述定义和鉴别标准给出了判断一个主体是否为可信主体的依据,却没有对可信主体的可信程度进行规范化描述.在最初的 BLP 模型中,可信主体的权能是不受限制的,它的所有行为都认为是可信的,这显然与最小特权原则相悖.所以在实际系统中,围绕可信主体的权限控制机制,出现了许多 BLP 模型的变体.其中影响较大的是 Bell 在 BLP 模型网络解释中提出的主体敏感标记范围模型.在该模型中,主体的当前安全级是一个二元组(α-min, α-max),从而使得可信主体对强制访问控制策略的超越被限制在一个范围内.此后,Mayer 针对操作系统进程间通信的特性,对其进行了修订[3],修订后的模型被应用于 TMach 以及 SELinux 的内核中.它们都基于 BLP 模型的静止性(tranquility)原理,即主客体的敏感标记函数是与系统状态无关的函数,在系统运行的任何时刻,主客体的敏感标记不会发生变化.这种静态敏感标记方案的最大优点是实施简单,正确性易于通过形式化进行证明.然而,从直观上讲,敏感标记范围本身的粒度太粗,在进程生命周期内保持不变,意味着可信主体的特权将保持不变,这并不符合最小特权原则.为方便起见,后文中把这两个模型统称为 Bell 模型.

为了进一步限制可信主体的特权,在开发 GEMSOS 时,Schell 等人提出了多级可信主体的概念.Lee 等人[4]也提出了相似的部分可信主体的概念.二者都认为可信主体在特定行为下才是可信的,从而将可信主体过于宽泛的权限限制在一个安全管理员可以管理和控制的范围内.但是,这两种概念并没有明确给出主体敏感标记的动态调整方案.

文献[5]基于主体历史敏感标记,提出了内外层空间判定方案,建立了具有动态适应性的 ABLP(adaptive BLP)模型,并在红旗安全操作系统(red flag secure operating system,简称 RFSOS)中进行了实施.但是,文献[6]随后指出:ABLP 在模型本身和模型设计上存在原子操作涉及步骤和变量过多、可信主体作用不明确、可能存在隐通道等问题.针对 ABLP 的问题,文献[6]提出了一种改进的动态标签模型:DBLP(dynamic BLP).DBLP 为客体也提供了一个安全级别的范围,并且区分了访问的多级特性和单级特性.文献[6]还对 DBLP 模型的安全不变量进行了证明,在理论上具有一定的完备性.但是, DBLP 模型引入多级客体缺乏充足的理由,易于带来新的隐蔽信道.同时,DBLP 判断条件层次多,尤其在涉及多级客体、IPC 对象的安全决策上甚至需要修改现有主流操作系统的核心机制才能实施,难度较大.就本文需要讲述的重点而言,DBLP 仍然采用了 TMach 敏感标记范围的方式实现可信主体的访问权限控制.

还有一种方式是采用多级安全策略之外的机制实现可信主体,例如,Posix1003.1e 草案[7]所描述的权能(capability)模型直接定义了一些超越 MAC 的标志位:如果一个进程具有这些权能标志位,就能够超越多级安全策略的限制.但是,引入其他模型或机制将给 MLS 系统状态的转换带来不可预料的影响,增加了安全性验证的难度.从 MLS 视角来说,这种不受任何限制的超越也不符合最小特权原则.为此,文献[8]采用可信度的概念,通过为主体附加一个兼顾完整性的安全标记将可信主体纳入安全策略中,既便于实施又易于验证,这种思想很值得借鉴.

基于以上分析,我们认为:既然可信主体对于多级安全操作系统不可避免,如果能在满足系统需要的前提下使其特权最小化,并且在 MLS 策略范围内支持这种最小特权,则将对提高多级安全操作系统的安全性和实用性具有重要的意义.为此,我们提出了一个基于离散标记序列(discrete label sequence,简称 DLS)的多级安全模型,试图在 MLS 安全策略的范围内,以可信主体本身的应用逻辑为依据解决可信主体的最小特权问题.

本文第 1 节介绍 DLS 模型,包括模型变量、敏感标记函数和安全保持规则.第 2 节给出可信主体敏感标记序列与当前敏感标记的确定算法.第 3 节对 DLS 模型与敏感标记范围模型进行比较,说明 DLS 模型对可信主体

最小特权化具有更好的支持.第 4 节总结全文.

1 DLS 多级安全模型

1.1 DLS组成元素

(1) 基本变量

我们采用文献[6]提出的变量类型来定义 DLS 模型的变量.

主体集合 S: $\wp SUBJECT$,包括非可信主体 S'和可信主体 S_T.

客体集合 O: $\wp OBJECT$.

敏感标记集合 L: $\wp LEVEL$.

访问方式集合 A: $\wp OPERATION$,包含$\{r,a,w\}$,分别代表只读、只写和读写.

请求事件集合 R: $\wp REQUEST$,决策集合 D: $\wp DECISION$.

敏感标记间的控制关系集合$\{=,\geq,\leq,\neq\}$:$L\leftrightarrow L$,分别表示相等、控制、被控制和不可比较.

(2) 敏感标记序列和敏感标记序列集合

设 $P(L)$表示 L 的幂集,$SL\in P(L)/\varnothing$,为 L 的一个非空子集.

定义 **1.1**. $q=(l_1,l_2,\ldots,l_m,\ldots)$是 SL 上的敏感标记序列,如果对于非空有限序列 q 有

$$\forall i\leq length(q),q[i]\in SL,$$

其中:i 为正整数;$length(q)$表示 q 的长度;$q[i]$表示序列 q 的第 i 个元素.

定义 **1.2**. 敏感标记序列集合 $Q=\{q|q$ 是 L 上的敏感标记序列$\}$.

(3) 敏感标记函数

DLS 模型与经典 BLP 模型相同,非可信主体的当前敏感标记在它的整个生命周期内是固定不变的.而 DLS 模型中可信主体的当前敏感标记是变化的,并且被限制在一个敏感标记序列上发生变化.因此有下列敏感标记函数:

- 非可信主体

最大敏感标记函数 f_s:$S'\mapsto L$;

当前敏感标记 $f_c=f$.

- 可信主体

敏感标记序列函数 q_s:$S_T\mapsto Q$;

最大敏感标记函数 $f_s=upperbound(\{q_s[i]|0<i\leq length(q_s)\})$,是敏感标记序列中所有敏感标记的最小上界;

当前敏感标记函数 f_c:$S_T\mapsto\{q_s[i]|0<i\leq length(q_s)\}$.

其中,q_s 用来记录可信主体生命周期内所经历的敏感标记变化序列,它与 f_c 的具体关系将在后文详细叙述.

- 客体

敏感标记函数 f_o:$O\mapsto L$.

1.2 DLS安全保持规则

一般来说,安全模型包括两个基本部分:安全的定义和操作规则.前者包含了关键性安全需求,对应系统的安全策略;后者描述所定义的“安全”是如何实施的[9].下面按照文献[6]的方式给出 DLS 保持安全状态的形式化描述,然后给出安全实施规则.

定义 **1.3**. 系统状态Σ是一个四元组$(V,R\times D,\omega,v_0)$,其中:

- V 是状态空间;
- $R\times D$ 是当前状态下的请求输入集合和决策输出集合;
- ω是状态转换函数ω:$R\times D\times V\mapsto V$,给定某个状态下的请求输入和决策输出,该函数负责确定下一个状态;
- v_0 是系统的初始状态.

定义 **1.4**. V 是定义 1.3 中系统Σ的状态空间,C 是 V 的子集,CT 是 $V\times V$ 的子集.一个状态 v 被称为是关于 C

的安全状态,如果 $v\in C$;一个请求和判断(r,d)被称为关于 C 和 CT 是安全的,如果$((v,r,d),v^*)\in\omega$,则

- 如果 $v\in C$,那么 $v^*\in C$;
- $(v,v^*)\in CT$

系统Σ关于 C 和 CT 是安全的,如果:

- v_0 关于 C 是安全的;
- $\forall(r,d)\in R\times D$ 关于 C 和 CT 是安全的.

定义 **1.5**. 一个规则是函数$\rho:R\times D\mapsto D\times V$,如果$\rho(r,v)=(d,v^*)$,则$\rho$关于 C 和 CT 是安全的,如果 $v\in C\Rightarrow v^*\in C$ 且 $(v,v^*)\in CT$.

下面,我们将给出 DLS 符合定义 1.5 的规则.在以下规则中沿用 BLP 模型关于状态的定义,即 $v=(b,f,H,M)$. 其中,$b\in S\times O\times A$,代表系统当前允许的访问集.

规则 **1.1**(简单安全性规则).

如果一个主体能够只读访问或者读写访问一个客体,那么它的最大敏感标记必须控制客体的敏感标记,即

$$\forall s\in S,\forall o\in O,(s,o,r)\in b\vee(s,o,w)\in b\Rightarrow f_s(s)\geq f_o(o).$$

规则 **1.2**(*-安全性规则). 这一规则分为自由(Liberal)*-安全性规则和严格(Strict)*-安全性规则.

自由*-安全性规则:

$$\forall s\in S,\forall o\in O,(s,o,r)\in b\vee(s,o,w)\in b\Rightarrow f_c(s)\geq f_o(o),$$
$$\forall s\in S,\forall o\in O,(s,o,a)\in b\vee(s,o,w)\in b\Rightarrow f_c(s)\leq f_o(o).$$

严格*-安全性规则:

对于非可信主体,有

$$\forall s\in S',\forall o\in O,(s,o,a)\in b\vee(s,o,w)\in b\Rightarrow f_c(s)=f_o(o);$$

对于可信主体则,有

$$\forall s\in S_T,\forall o\in O,\forall x\in A,(s,o,x)\in b\Rightarrow f_c(s)=f_o(o).$$

即一个可信主体对其他任何客体任何形式的访问,都要求该主体的当前敏感标记与客体的敏感标记相同.通过这一规则,可信主体当前可访问客体被严格限制在与当前敏感标记相同的客体上.表面上看,可信主体受到了比非可信主体更多的限制,事实上,可信主体的当前敏感标记是变化的,因此这种限制是必要的.

规则 **1.3**(自主安全性规则).

$$\forall s_i\in S,\forall o_j\in O,(s_i,o_j,x)\in b\Leftrightarrow x\in M_{ij}.$$

规则 **1.4**(客体创建规则).

$$\forall s\in S,\forall o\in O,o\text{ 由 }s\text{ 新创建}\Rightarrow f_c(s)\leq f_o(o).$$

在 BLP 模型中,新创建客体被认为是一个非激活客体被激活的过程,因此,该规则也可以叫做激活规则.该规则表明,新建客体的敏感标记应不小于主体的当前敏感标记.

规则 **1.5**(可信主体敏感标记变化规则).

每一个可信主体都有自己的当前敏感标记变化规则,这些规则由安全管理员根据实际系统计算,并通过可信计算基 TCB 配置和实施.我们将在第 2 节叙述规则 1.5 的细节.

1.3 DLS模型的安全性分析

如果仅考虑非可信主体,以上规则与经典 BLP 模型的规则表述是一致的,系统安全性很容易通过 BLP 模型的证明方式进行证明,因此,系统只需考虑可信主体的规则实施是否安全.换言之,如果我们能证明可信主体实施以上规则能保持系统安全性,那么就可以证明 DLS 模型的安全性.

从模型层面来说,以上只有规则 1.2 的严格*-属性规则和规则 1.5 涉及到可信主体.而对于严格*-属性规则来说,只要通过规则 1.5 保证当前安全敏感标记是正确的,那么在当前敏感标记确定的情况下,严格*-属性规则对可信主体的限制比非可信主体还要严格,其安全性是显然的.这样,唯一需要保证的就是在系统的实现层面上保证可信主体在规则 1.5 下保持安全性.

2 基于状态的可信主体当前敏感标记确定方案

2.1 基本思想

从主体生命周期的角度观察,可信主体可以被看作是以不同的敏感标记运行的非可信进程在时间和逻辑上组合而成.这些不同的敏感标记组成了可信主体的可访问敏感标记状态空间,反映了可信主体的特权状态,进而反映了可信主体的应用逻辑.我们充分利用了这一思想,根据可信主体的行为和对特定敏感标记的客体的实际访问需求,将可信主体的生命周期分解为一系列的非可信状态,每个状态下可信主体都具有一个特定的当前敏感标记,对应一个受限的可访问客体集(根据规则 1.2).不同状态之间的切换由状态转换事件触发.

我们引入非可信状态(untrusted state,简称 US)的概念来分解可信主体对 MLS 策略的超越过程,每一个非可信状态对应一个当前敏感标记和一系列可接受的状态转换事件.同时,引入可信请求事件(trusted request event,简称 TRE)的概念来描述非可信状态切换的触发机制,每一个可信请求事件由事件类型、参数和可达状态编号列表 3 个分量组成.这些信息都从系统的安全策略库(security policy database,简称 SPD)读取.此方案如图 1 所示.

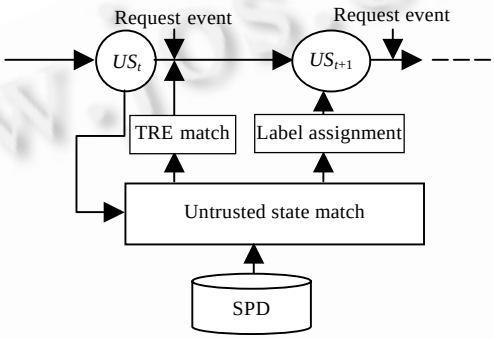

Fig.1 Dynamic tuning scheme for current label of trusted subject

图 1 可信主体当前敏感标记动态调整示意图

设 t 时刻可信主体所处的状态为 US_t,它向系统发出类型为 e 的请求事件.系统首先从可信主体的 SPD 中查找 US_t,如果存在匹配的非可信状态,则进一步判断 e 是否为当前状态下可信请求事件.如果 e 匹配,则根据其参数将可信主体转入下一个非可信状态 US_{t+1},并赋予相应的当前敏感标记.因此,规则 1.5 可以非形式化地表述为:可信主体的当前敏感标记是其当前非可信状态的敏感标记,当前敏感标记的变化由可信请求事件触发.下面从模型层面给出其安全性的形式化表述.

2.2 规则安全的形式化描述

规则 1.5 实际上是将可信主体相关的状态从全局状态变量 $v=(b,f,H,M)$中分解出来,只考虑 $b\in S_T\times O\times A$ 所对应的状态分量.下面在一些给定集的基础上给出了动态敏感标记方案下可信主体子安全状态的定义.首先给出以下模型元素:

- *Prog*:可信主体对应程序集合;
- *Proc*:可信主体对应进程集合;
- *PUState*:所有可信主体对应进程可能的非可信状态集合;
- *TRE*:可信请求事件集合;
- $\boldsymbol{P}(X)$为集合 X 的幂集.

定义 **2.1**. 可信主体相关的子系统状态 v 是一个六元组($PLT,CPL,PUS2TRE,PUS2L,TRE2US,P2P$),其中:

- $PLT\subseteq\boldsymbol{P}(Prog\times PUState\times L\times\boldsymbol{P}(TRE))$,一个程序的敏感标记表,为程序和其非可信状态及此状态下的敏感标记、所允许的可信请求事件 4 者间的关系;
- $CPL\subseteq\boldsymbol{P}(Proc\times PUState\times L)$,一个进程当前非可信状态的敏感标记信息表,为进程与非可信状态下敏感标

记间的多对多关系;

- *PUS2TRE*:*Proc*×*PUState*→***P***(*TRE*),*P2US*(*p*,*us*)函数得到进程 *p* 当前非可信状态 *us* 下允许的可信请求事件;
- *PUS2L*:*Proc*×*PUState*→*L*,*PUS2L*(*p*,*us*)函数将进程 *p* 的当前非可信状态 *us* 映射到其当前敏感标记;
- *TRE2US*:*Proc*×*PUState*×*TRE*→*PUState*,*TRE2US*(*p*,*tre*)函数用来计算进程 *p* 在当前状态下收到可信请求事件 *e* 后的下一个非可信状态;
- *P2P*:*Proc*→*Prog*,*P2P*(*p*)函数将进程 *p* 映射到其对应程序.

定义 **2.2**. 一个可信主体相关的子系统状态 *v*=(*plt*,*cpl*,*pus2tre*,*pus2l*,*tre2us*,*p2p*)是安全的,当且仅当:

$\forall(p,us,l)\in cpl\Rightarrow(p2p(p),us,l,pus2tre(p,us))\in plt$, 并且对于 $\forall e\in pus2tre(p,us),us'=tre2us(p,us,e),l'=pus2l(p,us'),tre'=pus2tre(p,us')$,有$(p2p(p),us',l',tre')\in plt$.

由定义 2.2 可知:处于安全状态时,一个可信主体对应进程的当前敏感标记将精确地等于此进程对应程序在 PLT 表记录中的敏感标记,并且通过可信请求事件切换到下一个非可信状态后,新非可信状态信息(包括敏感标记、允许的可信请求事件)也必须在 PLT 表中.如果在规则 1.5 的实施中始终满足定义 2.2,那么,子系统就是安全的.

2.3 规则实施

为了保证规则 1.5 的正确实施,除了根据可信主体对应程序的逻辑进行分析配置以外,还需要考虑下面一些问题.首先,所有可能的非可信状态组成了可信主体的状态序列,每个状态的敏感标记值构成了前面提到的可信主体的敏感标记序列 q_s,从而,可信主体的所有可访问权限也被限制在 q_s 上.但需要注意的是:在每一个非可信状态下,由于 TRE 的类型和参数有可能不同,可信主体实际运行过程的当前敏感标记所组成的 q_s 序列可能不同.为此,我们在可信主体的配置语言中使用了 *canswitchto* 原语,表示当前状态在某个可信请求事件到来时可达的其他状态.如果没有 *canswitchto*,则默认进入下一个编号的状态.同时,在配置语言中广泛引入了各种通配符、内嵌常量和内嵌变量,以方便可信主体的配置.通配符主要有“任意(any)”与“非(!)”两种形式,例如任意文件、任意用户、非 root 用户等.内嵌常量有“最低(LOW)”、“最高(HIGH)”分别表示系统最低和最高敏感标记,“空(NULL)”表示非等级类别中的空集,“全部(ALL)”表示所有非等级类别组成的集合.内嵌函数有 USE_EUID,表示使用当前非可信状态下的有效用户敏感级别作为可信主体的当前敏感级别.可信主体的配置集中存放在一个文件中,其格式参见附录 A.

在可信主体的执行路径上设置检查点,可以很容易地找到引起敏感标记变化的请求事件,然后根据这些请求事件是否符合可信程序的逻辑,确定其是否为可信请求事件,继而获得处于可信请求事件之间的非可信状态.基于 Linux 的 MLS 系统可以利用 LSM(Linux security modules)提供的钩子(hook)函数,基于 FreeBSD 的 MLS 系统可以利用 MAC 框架提供的入口点(entry point)函数.

3 DLS 模型与敏感标记范围模型的分析和比较

这里,我们把所有以敏感标记范围实现可信主体的模型(包括前面提到的 Bell 的 BLP 网络解释模型、TMach 的修订模型、DBLP 模型等)统称为敏感标记范围模型.本节将对 DLS 模型与敏感标记范围模型对可信主体最小特权的支持进行比较.

Bell 最初将主体当前敏感标记函数 f_c 变为[*a*-min,*v*-max]是为了给主体的下写提供更多的自由,而且在现实系统中确实有这样的需求,例如:军队中上级指挥官向下级传达命令、一个高级别用户需要更新存放在低级别客体中的个人信息(如使用 passwd 程序修改/etc/shadow 中的密码)等.在具体实施中,安全管理员需要找到可信主体需要下写的所有客体的最低敏感标记,需要上读的所有客体的最高敏感标记,分别作为可信主体的 *a*-min 和 *v*-max,可信主体对所有敏感标记处在[*a*-min,*v*-max]间的客体具有读写权限.敏感标记范围模型之所以采用这种方案,其理由可以用下面的例子说明:

在一个实施 MLS 策略的系统中,$S=\{s_1,s_2,s_3,s_4,s_5\}$,$O=\{o_1,o_2,o_3,o_4,o_5\}$,$L=\{l_1,l_2,l_3,l_4,l_5\}$.主体 s_i,o_i 的敏感标记为

l_i,且 $l_i \le l_j$,$1 \le i \le j \le 5$.现假定系统的正常运行需要 s_3 下写客体 o_1,而 s_3 写入 o_1 的内容又可以被 s_2 所读取,写入 o_2.信息经历了 $s_3 \to o_1 \to s_2 \to o_2$ 的过程,相当于 l_3 级别的信息最终也流到了 l_2 级别.因此,敏感标记范围模型认为可以将 s_3 下写的可信度扩展到包含 l_2 的$[l_1,l_3]$范围上.同理,如果 s_3 可以上读 o_5,它也可以上读 o_4,于是上读的可信范围扩展到$[l_3,l_5]$.

然而在实际系统中,这个理由并不成立,例如以下 3 种情况:

(1) 执行严格*-属性的 IPC 客体

如果 o_1 是 IPC 客体,并且系统在有关 IPC 客体的访问上实施严格*-属性,即要求与 IPC 客体具有同样敏感级别的主体才能访问 IPC 客体,那么,s_3 下写 o_1 之后,s_2 并不能访问 o_1.

(2) 内容加密(content encryption)

如果 s_3 写入 o_1 的内容被加密,那么,即使 s_2 读取到这些内容,如果不能解密,也是一无所获.典型例子如使用 passwd 修改密码,因为密码已经加密,所以不必担心被其他级别的用户读取.

(3) 挥发性客体(volatile object)

如果 o_1 是类似于/dev/null 这样的设备,那么,s_3 写入 o_1 的内容瞬间消失,s_2 根本没有机会读取到这些内容.

在以上 3 种情况下,既然 s_2 不能从 o_1 获得 s_3 下写的内容,那么,有用的信息流就不会必然到达 l_2 级别.因此,将 s_3 的下写范围扩展到包含 l_2 的整个$[l_1,l_3]$区间是不合理的.

而对于 DLS 模型来说,如果对 s_3 的配置合理,在下写 o_1 这个可信请求事件到来时,s_3 的当前敏感标记将会调整为 l_1.在完成对 o_1 的写打开操作后,s_3 又可以通过其他可信请求事件将当前敏感标记调整到其他级别.而且根据规则 1.2 的严格*-安全规则,s_3 的当前敏感标记为 l_1 时,它只能打开敏感标记同为 l_1 的客体.二者在下写上的区别可以用图 2 来表示.图中[a-min,v-max]间的虚线表示在敏感标记范围模型中,可信进程在整个生命周期内可以读写的范围;而实线表示在 DLS 模型中可信主体访问权限的变化情况:如果实施严格*-属性,那么在$[0,t_0]$时段,当前敏感标记为 l_3,只能访问 l_3 级别的客体;在$[t_0,t_1]$时段,只能访问 l_1 级别的客体;在$[t_1,t_2]$时段,跳回 l_3;在 t_2 时刻之后,又可以被不同的可信请求事件触发,当前敏感标记调整为 l_2 或者其他级别.

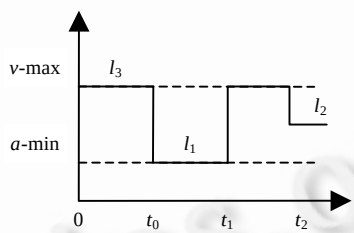

Fig.2 Comparison of trusted subject's right between range model and DLS

图 2 可信主体权限在 DLS 模型和敏感标记范围模型中的区别

需要强调的是,可信主体本身的应用逻辑是确定可信请求事件的唯一依据.在合理配置的可信请求事件的前提下,DLS 模型保证了可信主体在生命周期内以非可信状态序列的形式遵守 MLS 策略内的最小特权原则.仍然以高级别(设为 l_{high})用户运行 passwd 程序修改低级别(设为 l_{low})shadow 文件中的密码为例,定义打开和关闭 shadow 文件为两个可信请求事件,则 passwd 进程的非可信状态序列为:在 open 可信请求之前的敏感标记为 l_{high};之后降为 l_{low};在写入新密码并发出关闭文件请求后,再次升为 l_{high}.这样,即使 passwd 程序被木马替换,只要其破坏行为中没有匹配系统配置的可信请求,则将不会存在多个非可信状态,从而高级别用户不会把敏感信息轻易泄漏出去.

4 总 结

在以敏感标记范围模型指导 MLS 系统的开发和配置时我们发现:一些应用程序在设置为可信主体运行后,其可读写的范围被放大,超出了所需权限.本文在分析国内外已有多级安全模型的基础上,针对敏感标记范围模型在可信主体最小特权支持上的不足,通过对可信主体状态序列的分解,提出了基于离散敏感标记序列的多级

安全模型 DLS.DLS 模型具有以下特点:

- 引入非可信状态和可信请求事件的概念,使可信主体权限配置以其应用逻辑为依据;
- 支持可信主体 f_c 在敏感标记序列范围内的动态调整;
- 在 MLS 的策略框架内实现,便于进行高安全等级操作系统所必需的形式化验证.

DLS 模型为可信主体权限的合理设置提供了指导,并且提供了表达力较为丰富的策略配置语言,但是这种配置语言的完备性还需要进一步验证.与大多数 MLS 系统一样,目前可信主体的配置工作仍然需要分析源代码或检查点输出信息.在以后的工作中,我们希望给出一种能自动将可信主体分解为多个非可信状态,并提取出可信请求事件的方法和工具.

References:

[1] Bell D, LaPadual LJ. Secure computer system: Unified exposition and MULTICS interpretation. MTR-2997 Rev.1, Bedford: The MITRE Corporation, 1976.

[2] Bell D. Secure computer systems: A retrospective. In: Proc. of the 1983 IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 1983. 161−162.

[3] Mayer FL. An interpretation of a refined Bell-La Padula model for the TMach kernel. In: Proc. of the 4th Aerospace Computer Security Conf. 1988. 368−378. http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=3362

[4] Lee TMP. Using mandatory integrity to enforce commercial security. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE Computer Society Press, 1988. 140−146. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=8106

[5] Shi WC, Sun YF. History sensitivity of the multilevel security policies. Journal of Software, 2003,14(1):91−96 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/91.htm

[6] Ji QG, Qing SH, He YP. An improved dynamically modified confidentiality policies model. Journal of Software, 2004,15(10): 1547−1557 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1547.htm

[7] Draft standard for information technology—Portable operating system interface (POSIX)—Part 1: System application program interface (API)—Amendment #: Protection, audit and control interfaces. IEEE Computer Society, 1997.

[8] Xie J, Xu F, Huang H. Trust degree based multilevel security policy and its model of state machine. Journal of Software, 2004,15(11):1700−1708 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1700.htm

[9] National Computer Security Center. A Guide to Understanding Security Modeling in Trusted Systems. 1992.

附中文参考文献:

[5] 石文昌,孙玉芳.多级安全性政策的历史敏感性.软件学报,2003,14(1):91−96. http://www.jos.org.cn/1000-9825/14/91.htm

[6] 季庆光,卿斯汉,贺也平.一个改进的可动态调节的机密性策略模型.软件学报,2004,15(10):1547−1557. http://www.jos.org.cn/1000-9825/15/1547.htm

[8] 谢钧,许峰,黄皓.基于可信级别的多级安全策略及其状态机模型.软件学报,2004,15(11):1700−1708. http://www.jos.org.cn/1000-9825/15/1700.htm

附录 A 基于非可信状态和可信请求事件的可信主体配置语言

```
#begin_config
    #begin_prog
        path:程序路径名
        users:可运行此程序的用户 ID 列表
        #begin_state
            stateno:状态编号
            mls_label:{此状态具有的当前敏感标记}
            #begin_tre
```

```
                    type:{某个普通访问或者某个系统调用}
                    param:{与类型相关的参数}
                    canwitchto:{此请求触发的其他可达状态的状态编号}
                #end_tre
                其余可信请求事件…
            #end_state
            其余非可信状态…
        #end_prog
        其余程序…
#end_config
```


武延军(1979－),男,陕西延安人,博士生,主要研究领域为操作系统安全.

梁洪亮(1972－),男,博士,副研究员,主要研究领域为系统软件,信息安全.

赵琛(1967－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为编译技术,软件测试.