面向移动 Web 操作系统的 BLP 改进模型及应用

朱大立[1,2], 杨 莹[1,2], 金 昊[1,2], 邵 京[1,2], 冯维淼[1,2]

[1] 中国科学院大学, 北京 中国 100049

[2] 中国科学院信息工程研究第四研究室, 北京 中国 100093

摘要 作为重要的机密性策略经典模型, BLP 模型通过对主体和客体进行分级和标记, 并引入高安全等级的引用监视器, 实现信息系统的强制访问。随着移动智能终端的普及, Web 操作系统因其具有移动性、移植性、高扩展性和跨平台性等优点, 成为移动政务系统的主要解决方案之一, 并越来越受到研究人员的重视。但现有的 Web 操作系统对机密性要求不高, 无法满足移动政务系统对安全保密的需求。本文从安全模型构建入手, 对智能终端的 Web 操作系统进行抽象建模, 并重定义 BLP 模型的元素, 增强主客体的访问控制以提高其机密性。鉴于 BLP 模型缺乏可信主体的最小权限原则和完整性约束, 本文在改进的 BLP 模型当中重新划分主体、客体的安全级, 增加可信级别标记和角色映射函数, 并针对现有的 Web 操作系统进行模型映射, 实现了最小权限原则、主体完整性约束和域间隔离机制, 可有效提高 Web 操作系统机密性等级。

关键词 Web 操作系统； BLP 模型； 移动终端； 操作系统安全； 最小权限原则； 完整性； 隔离

中图分类号:TP309.1 DOI 号 10.19363/j.cnki.cn10-1380/tn.2017.10.002

Research and Application of Improved BLP Model for Mobile Web Operating System

ZHU Dali[1,2], YANG Ying[1,2], JIN Hao[1,2], SHAO Jing[1,2], FENG Weimiao[1,2]

[1]University of Chinese Academy of Sciences, Beijing 100049, China

[2]Institute of Information Engineering, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Abstract BLP model is a classic model of important strategies of confidentiality, which realizes the mandatory access control by classifying and marking the subjects and objects as well as introducing reference monitor of high safety level. With the popularity of mobile devices, web operating system is attracting more and more attentions from researchers for its advantages of cross-platform, high mobility, portability and scalability. And it is used as a system-level solution of mobile e-government. But existing Web operating systems provide low confidentiality, and ambiguous design of system security access control policy, so they cannot meet the security demand of the mobile e-government system. This paper builds the security model based on the abstract modeling of Web operating system, and redefines the model elements, mapping functions, as well as access control policy on both the subject and object to improve its confidentiality. As BLP model is lack of the least privilege principle on trusted subject and integrity constraints, we redraw the security level of the subject and object, add the tag of confidence level and role mapping function which is according to the existing security model of Web operating system. Finally, we implement the principle of least privilege, the integrity constraints on subjects and isolation mechanism between domains, which can effectively improve the security.

Key words Web operating system; BLP model; mobile terminal; operation system security; principle of least privilege; integrity; isolation

1 引言

信息系统的安全需求主要包含: 机密性、完整性和可用性。所谓一个系统是安全的, 就是指系统达到了当初设计时所制定的安全策略[1]。安全模型就是对安全策略所表达的安全需求的简单、抽象和无歧义的描述, 它为安全策略和实现它的机制之间的关联提供了一种框架。安全模型同时也描述了对某个安全策略需要用哪种机制来满足。

Bell-LaPadula(BLP)模型是实现多级安全

通讯作者: 冯维淼, 博士, 工程师, Email: fengweimiao@iie.ac.cn。

本课题得到中国科学院战略性先导专项项目: 重点行业应用系统信息安防关键技术研究(No. XDA06010703)资助。

收稿日期: 2016-05-06; 修改日期: 2016-09-08; 定稿日期: 2017-08-23

(Multilevel Security, MLS)机密性策略的经典模型[2]。为了实现 MLS 策略, BLP 模型定义了两个安全属性, 即简单安全性(ss-属性)和限制性安全性(*-属性)。ss-属性模拟了现实世界的情况, 即主体不能读安全级比自己高的客体中的信息; *-属性的引入则主要是为了解决特洛伊木马问题, 它要求主体只能写安全级高于自己的客体。尽管 BLP 模型存在着一些缺陷, 但仍然是最重要的安全模型[3]。当前对于 BLP 模型的改进研究, 主要根据具体应用场景, 实现权限管理或构建网络环境下的访问控制等方面进行。

目前智能终端操作系统中 Android、iOS 和 Windows Phone(WP)占据了大部分市场份额。iOS 和 WP 的安全依赖于其闭源性和应用溯源等安全机制[4], Android 由于开源, 使恶意软件和黑客非常容易进行权限获取和系统修改, 带来更多的内核层危害。随着网络带宽的增加和网络传输速度的提升, 越来越多的厂商推出了基于 Web 的移动操作系统, 如 Firefox OS、Ubuntu Touch、Samsung Tizen、Google Chrome 等, 这类基于 Web 的移动操作系统又被称为 Web OS[5]。Web OS 具有服务平台特性, 为人们的工作提供了移动性和跨平台性。特别是政府和企业级的用户, 他们尤为注重机密性, 通过基于 Web OS 的智能终端, 不仅可以使用云端存储的办公软件, 还可以将数据储存在守信的内网, 即云端, 这样既解决了移动办公问题, 又保证了信息的安全性。但是如果 Web 操作系统实现高保密性需求, 还需引入机密性安全模型保证其安全性, 来防止信息的未授权访问、修改以及合法用户对系统的不恰当访问等问题。

2 BLP 模型

2.1 模型简介

1973 年, D. Elliott Bell 和 Leonard J. LaPadula 提出的 Bell-LaPadula 模型[6], 简称 BLP 模型, 是第一个可证明安全性的计算机系统形式化模型, 也是一个符合军事安全策略的多级安全数学模型。它形式化定义了系统、系统状态以及系统状态间的转换规则; 定义了安全的概念和一组安全特性, 对系统状态和状态间的转换规则进行限制和约束。使得对于一个系统而言, 如果它的初始状态是安全的, 并且经过的一系列规则转换都保持安全, 那么可以证明该系统的终了也是安全的。

2.1.1 系统状态

系统状态 $v \in V$ 由一个有序四元组 (b, M, f, H) 表示, 其中:

1) 当前访问集 $b \subseteq (S \times O \times A)$ 表示在特定状态下, 哪些主体以何种访问属性访问哪些客体; S 是主体集, O 是客体集, A={r, w, e, a}是访问属性集。

2) M 表示访问矩阵, 其中元素 $M_{ij} \subset A$ 表示主体 S_i 对客体 O_j 具有的访问权限。

3) $f \subset F$ 表示访问类函数, 记作 $f = (f_s, f_o, f_c)$, 其中 f_s 表示主体的最高安全级函数; f_c 表示主体当前有安全级函数; f_o 表示客体的安全级函数。

4) 系统中的客体组成一棵树, H 表示当前的层次结构, 即当前客体的树状结构, $O_j \in H(O)$ 表示在此树形结构中, O_j 为叶子结点, O 为父节点。

2.1.2 状态转换规则

BLP 模型定义了 11 个状态转换规则: $R1$-$R11$, 描述了主体和客体的基本访问规则, 包括只读访问、只写访问、执行访问、读写访问、主体释放对客体访问属性、授予另一个主体对客体访问属性、撤销另一主体对客体访问属性、创建一客体(保持兼容性)、删除一组客体、改变主体当前安全级和改变客体的安全级, 并进行了形式化定义和证明。

规则 ρ 为函数 $\rho: R \times V \to D \times V$, 其中 R 为请求集, V 为状态集, 判定集 D={*yes*, *no*, *error*, ?}, 该函数表示给定一个请求 R 和一个状态 V, 由规则 ρ 决定系统产生的响应 D 和下一个状态 V。

如果系统 $\sum(R, D, W, z_0)$ 是一个**安全系统**, 对每一个时刻 $t \in T$, $(x_t, y_t, z_t, z_{t-1}) \in W$, 系统的每一个状态($z_0$, z_1, $\cdots$, z_n)均为安全状态, 其中 z_0 是初始状态。即当所有的 $\rho(R_k, v) = (D_m, v^*)$, 均有: v 是安全状态 $\Rightarrow v^*$是安全状态。那么规则 ρ 保持系统安全。

2.1.3 安全公理

为了解释什么样的状态是一个安全状态, 什么样的系统是一个安全系统, BLP 定义了一组安全公理。

1) 简单安全性 (Simple-security property)

状态 $v = (b, M, f, H)$ 满足简单安全性, 如果所有的

$$S \in S \Rightarrow [(O \in b(S : \underline{r}, \underline{w}))] \Rightarrow (f_s(S) \propto f_o(O))$$

其中, 符号 $\propto$ 表示前者支配后者, $b(S : x_1, x_2, \cdots, x_n)$ 表示主体 S 对其具有访问权限 $x_i (1 \leqslant i \leqslant n)$ 的所有客体集合。简单安全性表示如果主体对客体具有读和写的权限, 则主体安全级高于客体安全级, 且一个主体只能读不高于自身安全级别的客体。

2) *特性(* property)

S'是 S 的一个子集,状态 $v = (b, M, f, H)$ 满足相对于 S'的*特性(记为* property rel S'),iff 所有的

$$S \in S' \Rightarrow \begin{cases} O \in b(S:\underline{a}) \Rightarrow f_o(O) \propto f_c(S) \\ O \in b(S:\underline{w}) \Rightarrow f_o(O) = f_c(S) \\ O \in b(S:\underline{r}) \Rightarrow f_c(S) \propto f_o(O) \end{cases}$$

受限安全性(*特性)可防止向下写, 以及防止信息向下流动, 即一个主体只能写不低于自身安全级别的客体, 但不适用于可信主体。

3) 自主安全性(Discretionary-security property)

状态 $v=(b,M,f,H)$ 满足自主安全性, iff 所有的 $(S_i,O_j,\underline{x}) \in b \Rightarrow \underline{x} \in M_{ij}$。

2.2 BLP 模型分析

BLP 模型是状态机模型, 以状态在离散时间点上的变化出发, 通过状态迁移函数来描述状态迁移, 而状态迁移函数根据当前状态和输入来定义下一个状态, 也就是可能产生的输出。利用状态机模型 BLP 来设计一个安全的系统[7], 需要定义安全的状态集合, 检查确保所有的状态迁移都是安全的, 检查系统的起始状态是安全的, 如果满足上述性质, 则所有的状态迁移都是安全的, 那么系统将总是安全的。

但是在 BLP 模型中, 可信主体不受*特性约束, 访问权限太大, 必然会威胁到系统中的所有可访问资源的安全, 因此不符合当前操作系统普遍采用的最小特权原则, 应对可信主体的操作权限和应用范围进一步细化。Landwehr[8]指出使用 BLP 模型的系统, 真正执行的安全规则既有 BLP 的安全规则, 又有可信主体超越 BLP 安全规则的规则。因此, ABLP 模型[9]结合一个以 Linux 为基础的安全操作系统, 从理论上构造 BLP 公理的一种新的实施方法, 允许主体的当前敏感标记在主体的活动过程中合理地进行调整, 具有借鉴意义。文献[10]对主体和客体增加最大和最小安全标签, 在主体对客体进行读写操作时根据这几个标签来进行判断, 符合实际系统中的使用习惯。MBLP[11]面向最小特权管理, 对 BLP 模型下的用户类型、操作种类和操作权限进行细分, 约束可信主体的权限, 使其尽可能地符合最小特权原则。

BLP 模型主要注重保密性, 控制信息从低安全级流向高安全级, 但缺少完整性控制, 目前大量的研究针对其完整性缺失问题进行。利用保密性标志和可信度标志构成主客体的访问标志[12], 以此来实现保密性策略和完整性策略的结合, 解决完整性标志难以在实际中找到参照的问题, 但其完整性策略限制不够严格。文献[13]将 Bell 模型中限制可信主体权限的思想和实现非可信主体敏感等级历史变化的特点统一到一起。引入了主客体可信度和主体完整性, 解决多级安全策略中的完整性问题。Liu[14]试图通过用公式给主体和客体一个 0 或 1 的系数, 表示对机密性或完整性不关心或者关心, 来同时解决机密性和完整性的保护问题, 但这种安全级只能体现完整性和机密性相对重要程度, 忽略了主体或客体本身的安全等级, 不适合实际应用。还有一些[15,16]借鉴完整性模型 Biba[17]的思想, 增加对系统中主客体完整性的定义, 但是简单叠加的方法可用性较差[18]。还有基于预授权的机密性和完整性访问控制模型[19], 通过引入预授权机制对一些随机动态的访问活动进行合理控制实现系统机密性和完整性的统一, 具有较高的可用性。

BLP 模型虽然实现了自主访问控制和强制访问控制, 但没有考虑如何对系统中主体的权限进行配置。一些研究者提出基于时间限制的多级安全模型[20], 通过引入时间参数和检查函数, 限制主体的作用范围, 减小主体可能造成的危害。但是在现实系统当中, 存在不同密级的主体和客体, 还应对主体访问空间进行细粒度的划分。

2.3 Web 操作系统安全模型分析

Web OS 是一种基于标准统一的 Web 语言, 设计开发的操作系统, 它将现在的浏览器转化成一个平台, 但是与浏览器不同, 能够为系统资源提供保护域[21]。由于其在框架层和应用层采用的程序开发语言是 HTML5、CCS 和 JavaScript, 与 Android 所使用的 Java、iOS 使用的 Objective-C、Windows Phone 的.Net C#相比, 在采用的安全机制方面存在一些差异[22], 以 Firefox OS(FOS)为例[21,23]: Android 应用很容易将 IMEI/IMSI 泄露出去, 但是 FOS 不同, 不允许第三方应用通过 Web API 获取用户或设备的 ID, 只有预装应用(certified app)可以; 只有预装应用可访问系统的拨号服务; 不允许执行二进制应用(binary app); 虽然 Web app 像一个网站应用的快捷键, 但它的 manifest 文件必须存储在设备上; 不允许进程间直接通信, 需要通过 Web API(b2g 进程)进行通信等。这些安全机制根据 Web 系统特点, 为系统资源提供了全面的保护。

由于采用 Web 技术, 除了传统的堆栈溢出、ROP、0-day 攻击, 还有来自网络的点击劫持、钓鱼、缓存中毒、从存储和全局变量获取信息、跨站点脚本攻击等, 因此带来了更宽的攻击面[24]。如果采用厂商定制策略, 未来可能会遇到碎片化问题, 除此之外还有一些通用的移动终端安全威胁, 如来自移动网络的攻击, 来自内置广告的安全威胁。

目前开源的 Web OS, 如 FOS、Tizen、Chrome OS、Ubuntu Touch 等, 均基于 Linux 内核开发, 采用层级的安全模型, 其安全框架遵循最小权限原则, 即最初只给予最小的权限, 然后只在需要时选择性地授予额外权限。但在这些系统在各自的子系统中实施的安全模型有所不同, 如 FOS[25]的文件系统和内核采用 DAC 模型, 其对系统调用接口 Web API 采用 MAC 模型, 安全框架遵循最小权限原则。Tizen[26]采用混合的系统框架层, 应用程序可以是 Web 应用也可以是本地或混合的, 文件系统采用 DAC 模型, 内核 LSM Smack 采用强制访问控制 MAC。

Web 应用是 Web OS 与传统智能终端操作系统的主要区别所在, 它解锁了终端设备与应用服务之间的紧耦合关系, 应用服务能够通过互联网在不同终端之间互联互通, 使得 Web OS 更像一个服务平台[24,27]。但是 Web 技术的使用, 还存在因完全使用 HTML5、CSS 和 JavaScript 导致的信息泄露或超出预期的行为, 如特权提升等问题。目前的 Web 操作系统还缺乏完整性验证, Web APIs 作为应用访问系统资源的唯一接口和进程通信的通道, 应实施强制访问控制来保护系统资源和进程间的有效隔离。文献[28]分析了几个主流的 Web OS 平台(Ubuntu、Chrome、Windows 和 FOS)的应用安全模型和访问控制策略的实施方式, 通过本地对象和访问控制准则表达, 将应用程序区分为本地原生代码、应用程序的本地 Web 代码、应用程序的远程 Web 代码和第三方 Web 代码, 实现了细粒度的访问控制策略。

本文讨论的模型面向具有更高安全保密需求的智能终端 Web 操作系统。在传统开源 Web 操作系统的基础上, 提出系统保密性需求[29], 例如: 系统中用户需要进行身份认证, 以便进行访问控制; 用户和访问的信息可能都有保密级别, 因此需要进行强制访问控制; 因为保密不仅是安全信息流向的问题, 还有相互之间需要保密, 信息与其创建者的保密级别应该相同, 同一保密级别的用户之间不能相互访问各自的信息, 也就是隔离的问题; 只有信息创建者具有再授权操作的权力, 以便进行自主访问控制; 用户可以查询等于或低于其角色的保密级别的信息, 但不能进行插入、更改、删除、保存操作。因此, 需要增加客体的保密级别, 来实现对每个角色、每条信息进行保密级别标记。综上分析, 本文将 Web 系统的应用分为三种类型: Certified、Privileged 和 Web, 并赋予不同的安全属性和安全级。同时增加客体的保密级别, 来实现对每个角色、每条信息进行保密级别标记。然后构造对保密级别标记的判断条件, 实现按主体、客体的保密级别进行多级保密安全管理的策略。

BLP 模型中信息不能由高安全等级流向低安全等级, 概括为“不上读, 不下写”。这种策略限制了高密级主体向非敏感客体写数据的合理要求, 降低了系统的可用性, 但保证了系统较高的机密性。而 BLP 模型受到争议的根源在于没有调整安全级的相关策略, 即其“宁静性”原则[1]。BLP 模型认为安全等级和访问权限不改变的这种特性称为稳定性(tranquility), 不改变访问权限的操作认为是稳定的, 因此 BLP 模型的“宁静性”原则规定系统的安全除了系统初始状态的安全外, 还依赖于主客体敏感标记在它的整个生命期内的静态不变性。宁静性原则会导致模型在实际应用中缺乏灵活性, 但是本文研究的面向高安全保密等级的 Web 操作系统, 通常情况下访问权限和密级是不变的。因此, 基于以上分析认为利用 BLP 模型构建本文所假设前提的安全模型是适用的。

3 WBLP 安全模型设计

安全模型的设计以达到GB17859-1999安全标记保护级为目标, 实现具有系统审计保护级可信计算基, 提供有关安全策略模型、数据标记以及主体对客体强制访问控制的非形式化描述等[30]。在进行设计时结合 Web OS 的安全机制和安全策略, 基于对 BLP 模型的修改建立的, 记为 WBLP (Web OS based on BLP)。根据 2.2 和 2.3 对 BLP 模型的分析, 可信主体不满足最小权限原则, 因此对可信主体增加限制, 将主体进一步分级分类, 并实施最小权限原则; 在实际系统当中, 存在不同密级的文件, 应增加细粒度的访问控制, 因此对客体安全等级划分; 通过对 Web 操作系统的安全模型和安全机制的分析发现, 还缺乏对主体完整性的验证, 因此本文从以上三个方面对 BLP 模型进行改进。

3.1 定义

定义 1 *Dom* 为域函数, 将 S 中的主体或 O 中的客体映射到相应的域。当应用程序作为主体, *Dom* 的定义域是 S, 值域是 $\{S_u, S_p, S_c\}$, $S=(S_u \cup S_p \cup S_c)$, 其中 S_u 表示普通用户域, 代表系统中一般用户的操作, S_p 表示认证用户域或特权用户域, 代表系统中已认证用户的操作, S_c 表示系统特权用户域, 代表可信用户的操作; 对于客体来说, *Dom* 的定义域是 O, 值域是 $\{O_u, O_p, O_s, O_v\}$, $O=(O_u \cup O_p \cup O_s \cup O_v)$, 其中 O_u 代表一般用户空间域, O_p 代表受信任用户空间域,

O_s 代表系统空间域, O_v 代表病毒保护域。

定义 2 访问方式集合 A={r, a, w, e};访问控制矩阵 M; 请求属性集合 R, R={g, r},判定结果集合 D, D={*yes*, *no*, *error*,?}。时刻 T={1, 2,⋯, t,⋯}, 请求序列 X 中的任一元素记为 x, X= R^T; 判定序列 Y, Y=D^T, 其中的任一元素记为 y。$R^{(1)}$: 请求 get-/release-访问, $R^{(2)}$: 请求 give-/rescind-访问, $R^{(3)}$: 请求产生一个新客体, $R^{(4)}$: 请求删除一个客体, $R^{(5)}$: 请求改变安全级。

定义3 f_s是主体最大安全级函数,f_o是客体安全级函数, f_c 是主体当前安全级函数。机密等级分类集合 C, C 的值域为{C_1, C_2,⋯, C_q}, 且 C_1>C_2>⋯>C_q; 非等级类别集合 K, 可信等级分类集合 I, I={*certified*, *privileged*, *web_local*, *web_remote*}, 敏感级标签集合 L=C×K, 客体树状层次集合 H。

定义 4 P 为特权映射函数, 代表可信用户的权限集合, 它将 S_c 映射到不同的项中, P 的定义域为 S_c, 值域为{u_1, u_2, u_3, u_4, ⋯, u_n}, 那么可信主体集合 S_c= u_1∪u_2∪u_3∪u_4∪⋯∪u_n。

定义 5 系统状态由有序四元组 (b,M,f,H) 表示, 其中 $b\subseteq(S\times O\times A)$ 表示在特定状态下哪些主体以何种访问属性访问哪些客体; M 表示访问矩阵, 访问类函数 f, 记作 f=(f_s,f_o,f_c)。

定义 6 RA 为角色映射函数, 即可信用户权限对应的角色的集合, 它将特权用户集合映射到不同的集合中, RA 的定义域为{u_1, u_2, u_3, u_4,⋯, u_n},值域为{U_1, U_2,⋯, U_m}, 其中 U_i={u_j|1⩽j⩽n},1⩽i⩽m, 把 U_i 又称为角色 i。

模型元素的定义部分, 增加对主体安全级别的划分: 因为 Web 操作系统中作为主体的是进程, 对应于不同的应用, 如 Web 应用、可信应用和系统应用, 因此将模型主体分为 S_u、S_p、S_c, 分别给予不同的安全级别。但是 Web 应用分为本地代码和远程代码, Web 应用的远程代码应该是最不可信的, 给予最少的权限。给主体相应的可信等级, 值域为{*certified*, *privileged*, *web_local*, *web_remote*}, 该集合是偏序的。本文将客体存储空间划分为{O_u, O_p, O_s, O_v }, 客体安全级对应系统存储空间不同的分区, 从而实现客体域间隔离。

3.2 安全公理

安全公理用来对安全系统和系统的安全状态进行定义。

公理 1(简单安全公理)

状态 $v=(b,M,f,H)$ 满足简单安全公理, iff $\forall S_i$ 满足:

1) $(S_i\in S_c)and[O\in b(S_i:\underline{r},\underline{w})]\Rightarrow f_s(S_i)\triangleright f_o(O)$;

2) $\left.\begin{array}{l}(S_i\in S_u)or(S_i\in S_p)\\ \&(I(S_i)>I(O))\\ \&[O\in b(S_i:\underline{r},\underline{w})]\end{array}\right\}\Rightarrow f_s(S_i)\triangleright f_o(O)$。

其中, $\triangleright$ 表示前者支配后者, 如 $f_s(S)\triangleright f_o(O)$ 成立, 主体 S 的密级高于客体 O 的密级, S 的范畴包含 O 的范畴。$O\in b(S:\underline{x})$ 表示主体 S 对客体 O 具有 x 访问权限。

当主体对客体具有读和写的访问, 可信主体 S_c 需要具有对客体相应的访问权限, 非可信主体 S_p 和 S_u 除了在访问列表中具有访问权限外, 还需验证其可信等级是否高于客体的可信等级, 为真则主体安全级支配客体安全级, 状态 v 满足简单安全公理。简单安全公理表明一个主体只能读不高于自身安全级别的客体。

根据简单安全公理, 还可以推出如果主体 S_i 同时具有浏览客体 O_1 和修改客体 O_2 的访问权限, 则客体 O_1 的安全级别受客体 O_2 的支配, 即:

$$[(S_i,O_1,\underline{x}),x\in\{r,w\}]\wedge[(S_i,O_2,\underline{x}),x\in\{a,w\}]\Rightarrow f_o(O_2)\triangleright f_o(O_1)$$。

公理 2(*-特性公理)

非可信主体集合 S'是 S 的子集, S'=S_u∪S_p, 状态 $v=(b,M,f,H)$ 满足*特性公理, iff 对所有的 $S_i\in S$'有:

1) $O\in b(S_i:\underline{a})\Rightarrow f_o(O)\triangleright f_c(S_i)$

2) $O\in b(S_i:\underline{w})\Rightarrow f_o(O)=f_c(S_i)$

3) $O\in b(S_i:\underline{r})\Rightarrow f_c(S_i)\triangleright f_o(O)$

*-特性用于防止“向下写”以及防止信息向下流动, 即一个主体只能写不低于自身安全级别的客体。

公理 3(自主安全公理)

状态 $v=(b,M,f,H)$ 满足自主安全公理, iff 对所有的 $S_i\in S_u\cup S_p\cup S_c$, $O_j\in O$, 有

$(S_i,O_j,\underline{x})\in b\Rightarrow\underline{x}\in M_{ij}$。

公理 4(兼容性公理)

状态 $v=(b,M,f,H)$ 满足兼容性公理, iff 对任意的 i, k, 且 $O_j,O_k\in O$, 有

$O_i\in H(O_k)\Rightarrow f_o(O_i)\triangleright f_o(O_k)$

其中, $O_i\in H(O_k)$ 表示在此树形结构中, O_i 为叶子结点, O_k 为父结点。状态兼容性公理表示在树形结构中叶子结点安全级支配其上层结点。

公理 5(完整性公理)

完整性指主体的完整性, 主体完整性与主体所访问的客体无关, 取决于对其进行操作的另一主体, 因此需要对与主体相关的状态转换规则的安全性进行验证, 判断状态转换规则能否执行。经典 BLP 模型中[6,31]的相关规则中, $R5$、$R10$ 对自己的访问属性

和安全级进行修改, 不影响其他主体, *R*6 和 *R*7 是两个主体之间的操作, 因此需重点考虑。

状态 $v=(b,M,f,H)$ 满足完整性公理, iff 对任意主体 $S_i \in S_u \bigcup S_p$, $O_j \in O$, 有

$$\left.\begin{array}{l}(S_\lambda \in S_c)\ or\\ [(S_\lambda \in S_p)\&(f_c(S_\lambda) \triangleright f_s(S_i))]or\\ [(S_\lambda \in S_u)\&(f_c(S_\lambda) \triangleright f_s(S_i))]\end{array}\right\} \rightarrow (S_\lambda, g/r, S_i, O_j, \underline{x})$$
$$(S_\lambda \in S_u)\&(S_i \in S_p) \rightarrow (no, v)$$

WBLP 完整性公理在不增加基本操作类型的前提下, 仅对可能引起主体发生变化的操作进行验证和约束, 实现了一个较为轻量级的完整性保护, 在现实系统中较容易实现且对系统易用性影响较小。

公理 6(域间隔离公理)

状态 $v=(b,M,f,H)$ 满足域间隔离公理, iff 对所有的 $S_i \in S$, $O_j \in O$, $(S_i,O_j,\underline{x}) \in b$ 有:

1) $O_j \in O_u$ *and* $S_i \in S_u$, 那么 $\underline{x} \in \{a,w,r,e\}$;

2) $O_j \in O_p$ *and* $S_i \in S_u$, 那么 $\underline{x} \in \{r,e\}$;

3) $O_j \in O_s$ *and* $S_i \in S_u$, 那么 $\underline{x}=e$;

4) $O_j \in O_p$ *and* $S_i \in S_p$, 那么 $\underline{x} \in \{a,w,r,e\}$;

5) $O_j \in O_u$ *and* $S_i \in S_p$, 那么 $\underline{x} \in \{r,e\}$;

6) $O_j \in O_s$ *and* $S_i \in S_p$,那么 $\underline{x} \in \{r,e\}$;

7) $O_j \in (O_u \bigcup O_p)$*and* $S_i \in S_c$, 那么 $\underline{x} \in \{a,w,r,e\}$;

8) $O_j \in O_v$, 当 $S_i=U_n$, n 为特定值, 有 $\underline{x} \in \{a,w\}$;

9) $O_j \in O_s$, 当 $S_i=U_m$, m 为特定值, 有 $\underline{x} \in \{a,w\}$。

域间隔离公理对不同安全级的主体所能访问的客体域进行划分, 从而实现不同类型主体和客体的安全隔离。规则 8)、9)说明只有具有指定权限的特定主体才能在病毒防护区增加客体和对其客体进行修改操作。因此, WBLP 起到了阻止应用型病毒的感染。另外, 用户空间和系统空间的隔离, 使得用户的操作无法影响系统内核的安全。模型对于主体类型的划分, 与 Web 操作系统中存在的三种应用类型相对应, 实现系统中不同类型应用的隔离。

公理 7(最小权限原则)

*特性公理用于防止向下写来防止信息向下流动, 只适用于非可信主体。还需对可信主体进一步约束, 因此提出最小权限原则。

不同的主体在其被创建时都被授予一个可信等级, 为了保证系统的安全性, 这个可信等级应不再变更。根据可信等级, 将主体映射到不同的权限组, 每一组对应相同的权限。对于特权操作 $S_t = u_1 \cup u_2 \cup u_3 \cup u_4 \cup \cdots \cup u_n$, 由角色映射函数 *RA* 将它们映射到不同的角色中, 并把这些“角色”赋予系统中的指定用户, 这样, 操作系统中就存在有若干个特权用户, 这些特权用户共同完成系统的特权操作。这样, 每个特权用户只有完成其工作所需的最小特权, 而不能独自控制整个系统。

WBLP 模型公理中, 1-4 是基本安全公理, 实现了强制存取控制和自主存取控制两种访问控制方式。对于特权管理, WBLP 也指定了相应的规则, 如, 主体的安全级等于客体的安全级或具有指定特权的主体, 且在访问控制列表 *M* 中才能执行写访问。公理 5-7 分别实现了主体完整性的约束, 域间隔离和最小权限原则, 这几个公理是在 BLP 模型基础上, 根据 Web 操作系统的安全架构进行的改进。

经典 BLP 模型中存在隐蔽通道问题, 指的是敌手可以利用访问控制机制本身构造一个隐蔽通道, 使信息从高安全等级流向一个较低的安全等级。如一个低级别的主体在其安全等级上创建一个客体, 它的高安全等级的同谋(一般为特洛伊木马)提高或不改变客体的安全等级, 当较低安全等级的主体试图读客体, 无论该请求成功或失败, 都暴露了高安全等级的主体的行为, 那么一个比特的信息就从高安全等级流向了较低安全等级。

WBLP 模型将其中的强制性安全规则加以修改, 对创建和删除操作增加约束, 以防产生隐蔽通道。在简单安全公理中, 加入了可信等级 $I(S_i) > I(O)$ 的判断, 对读写操作增加判断, 修改为:

$$iff\ I(S_i) >= I(O) \rightarrow O \in b(S_i : \underline{r}, \underline{e})$$
$$iff\ I(S_i) = I(O) \rightarrow O \in b(S_i : \underline{a}, \underline{w})$$

如果满足, 则可以执行只读或读写的操作。因此 WBLP 模型能够限制隐蔽通道的产生。

3.3 状态转换规则

在 WBLP 模型中, 一个系统 $\sum(R,D,W,z_0)$ 是安全的, 当系统初始状态 z_0 保持系统安全, 且系统每一个请求 *R* 和产生的下一个状态 *v**也满足安全公理。根据 WBLP 模型的域间隔离公理, 在 11 个基本的状态转换操作中, 需要重新构造只读访问、只写访问、读写访问; *R6*、*R7* 在满足完整性公理的情况下, 才能执行; *R*9、*R*11 根据域间隔离公理和完整性公理, 对于 O_s 和 O_v 区域的客体不适用。

***R*1** 当符合以下条件时, 主体 S_i 对客体 O_j 进行只读访问, 即 $R1(R_k,v)=(D_m,v^*)$ 保持系统安全状态。

*特性函数: $*1(R_k,v)=true \Leftrightarrow f_c(S_i) \triangleright f_o(O_j)$

$$R1(R_k,v)=\begin{cases}(?,v), & iff\ R_k\notin dom(R1)\\ (yes,(b\cup(S_i,O_j,\underline{r}),\\ M,f,H)), & iff\ [R_k\in dom(R1)]\\ & \&[\underline{r}\in M_{ij}]\\ & \&[f_s(S_i)\triangleright f_o(O_j)]\\ & \&[O_j\notin O_v\cup O_s]\\ & \&[S_i\in S_c\ or\ *1(R_k,v)]\\ (no,v), & else\end{cases}$$

其中, $dom(R_i)$为 R_i的定义域。受域间隔离公理约束, 主体不能访问系统管理区 O_s和病毒保护区 O_v中的客体。当主体的访问属性中有对客体的只读访问, 且主体的安全级支配客体的安全级, 主体是可信主体或主体的安全级支配客体的安全级时满足上述条件的 $R1$ 引起的状态改变是安全的。

R2 当符合以下条件时, 主体 S_i对客体 O_j进行只写访问:

*特性函数: $*2(R_k,v)=true\Leftrightarrow f_o(O_j)\triangleright f_c(S_i)$

$$R2(R_k,v)=\begin{cases}(?,v), & iff\ R_k\notin dom(R2)\\ (yes,(b\cup(S_i,O_j,\underline{a}),\\ M,f,H)), & iff\ [R_k\in dom(R2)]\\ & \&[\underline{a}\in M_{ij}]\\ & \&[O_j\notin(O_v\cup O_s)]\\ & \&[S_i\in S_c\ \text{or}\ *2(R_k,v)]\\ (no,v), & else\end{cases}$$

域间隔离公理约束, 只有特定主体能向系统特权用户域 O_s和病毒保护域 O_v中的客体进行只写访问, 因此满足上述条件的 $R2$ 引起的状态改变是安全的。

R3 是主体对某个客体的执行访问, 不会影响系统内其他主体和客体, 因此满足 WBLP 模型的公理约束。

R4 根据域间隔离公理, 只有特定的主体能对系统特权用户域 O_s和病毒保护域 O_v中的客体进行只写和读写访问。当符合以下条件时, 主体 S_i 对客体 O_j 进行读写访问:

*特性函数: $*4(R_k,v)=true\Leftrightarrow f_o(O_j)=f_c(S_i)$

$$R4(R_k,v)=\begin{cases}(?,v),iff\ R_k\notin dom(R4)\\ (yes,(b\cup(S_i,O_j,\underline{w}),\\ M,f,H)), & iff\ [R_k\in dom(R4)]\\ & \&[\underline{w}\in M_{ij}]\\ & \&[f_s(S_i)\triangleright f_o(O_j)]\\ & \&[O_j\notin(O_v\cup O_s)]\\ & \&[S_i\in S_c\ \text{or}\ *4(R_k,v)]\\ (no,v), & else\end{cases}$$

R5 是主体对自身对某个客体访问属性的撤销操作, 不会影响系统内其他主体, 因此满足 WBLP 模型的公理约束。

R6、***R7*** 是一个主体对另一主体关于客体的访问属性的添加或撤销操作, BLP 模型中对该操作类型的约束, 是针对客体 O_j 及其父结点 $O_{s(j)}$是否为根结点可能产生的几种不同情况, 和每种情况需要分别满足的条件进行分析判断。根据 WBLP 完整性公理和域间隔离公理, 需要满足以下条件:

$$R6(R_k,v)=\begin{cases}(?,v), & iff\ R_k\notin dom(R6)\\ (yes,(b,M\backslash M_{ij}\cup\{x\},f,H)),\\ & iff[R_k\in dom(R6)]\\ & \&[(S_\lambda\in S_c)and(S_i\notin S_c)\\ & \quad and\ f_c(S_\lambda)\triangleright f_s(S_i)]\\ & \&[<[O_j\neq O_R]\&[O_{S(j)}\neq O_R]\\ & \&[O_{S(j)}\in b(S_\lambda:\underline{w})]>\\ & \text{or}<[O_{S(j)}=O_R]\\ & \&[GIVE(S_\lambda,O_j,v)]>\\ & \text{or}<[O_j=O_R]\\ & \&[GIVE(S_\lambda,O_R,v)]>]\\ (no,v), & else\end{cases}$$

其中, $R_k=(S_\lambda,r,S_i,O_j,\underline{x})\in R^{(2)}$, $\underline{x}\in A$; 当 O_k=O_R 或 $O_{s(k)}$=O_R 时, 表达式 $GIVE(S_\lambda,O_k,v)$ 为真, 表示 S_λ 在状态 v 下能够授予对 O_k的访问权; $M\backslash M_{ij}\cup\{x\}$ 表示对访问矩阵 M 中的元素 M_{ij} 进行修改, 并以元素 $M_{ij}\cup\{x\}$ 代替 M。

规则 $R6$ 的解释: 1)当客体 O_j及其父结点 $O_{s(j)}$均不是层次树的根结点, 即$[O_j\neq O_R]\&[O_{S(j)}\neq O_R]$, 主体的访问属性中有对父结点的读写权限; 2)当客体 O_j父结点 $O_{s(j)}$是层次树的根结点, 且S_λ在状态 v 下能够授予对 O_k 的访问权, 即 $GIVE(S_\lambda,O_j,v)$ 为真; 3)当客体是层次树的根节点时, 且$GIVE(S_\lambda,O_R,v)$ 为真; 在满足上述条件时, 主体S_λ可以授予另一主体S_i对客体O_j的访问权限。

类似的 ***R7***, $R_k=(S_\lambda,r,S_i,O_j,\underline{x})\in R^{(2)}$, $\underline{x}\in A$;

$$R7(R_k,v)=\begin{cases}(?,v), & iff\ R_k\notin dom(R7)\\ (yes,(b-(S_i,O_j,\underline{x}),M\backslash M_{ij}-\{x\},f,H)),\\ & iff[R_k\in dom(R7)]\\ & \&[S_\lambda\in S_c\text{or}\ (S_i\in S_u\cup S_p\\ & \quad and\ f_c(S_\lambda)\triangleright f_s(S_i))]\\ & \&[<[O_j\neq O_R]\\ & \&[O_{S(j)}\in b(S_\lambda:\underline{w})]>\\ & \text{or}<[O_j=O_R]\\ & \&[RES(S_\lambda,O_j,v)]>]\\ (no,v), & else\end{cases}$$

其中, 当 S_λ 在状态 v 下能够撤销对 O_j 的访问权, $RES(S_\lambda, O_j, v)$ 为真。

根据 WBLP 模型的域间隔离公理和完整性公理, ***R*8**、***R*9** 不能创建以 O_s 和 O_v 区域的客体为父结点的新客体, 因为该操作将破坏这两个区域的完整性。因此在满足 $O_j \notin (O_s \cup O_v)$ 时, 才能进行规则 ***R*8**、***R*9** 的状态转换。

规则 ***R*10** 为主体改变自身的当前安全级 f_c, 这一操作并不影响主体最大安全级 f_s, 因此, 符合 WBLP 模型的安全公理。

规则 ***R*11** 受域间隔离公理约束, 当符合以下条件时, 主体 S_i 可以改变客体 O_j 的安全级至 L_u:

- 客体不在系统管理区 O_s 和病毒保护区 O_v 中的;
- S_i 是可信主体, 并且主体当前安全级、客体 O_j 的安全级和 L_u 为偏序关系;
- 主体 S_i 的当前安全级支配 L_u, 主体 S_i 能以只读或读写模式访问客体 O_j;
- 客体 O_j 的安全级被改变为 L_u, 且导致的状态满足*特性;
- 客体 O_j 的安全级被改变为 L_u, 且导致的状态满足兼容性;
- 主体 S_i 有权改变客体的安全级。

即: $R_k = (r, S_i, O_j, L_u) \in R^{(3)}$,

$$
R11(R_k, v) = \begin{cases}
(?, v), iff\ R_k \notin dom(R11) \\
(yes, b, M, f \setminus f_o(O_j) \leftarrow L_u, H), \\
\qquad iff[R_k \in dom(R11)] \\
\qquad \&[O_j \notin O_v \cup O_s] \&[<(S_i \in S_c) \\
\qquad \& f_c(S_i) \triangleright f_o(O_j) > \\
\qquad \text{or} < f_c(S_i) \triangleright L_u \triangleright f_o(O_j) >] \\
\qquad \&[\forall S_i \in S[(O_j \in b(S : \underline{r}, \underline{w})) \\
\qquad\qquad \Rightarrow f_s(S) \triangleright L_u]] \\
\qquad \&[*11(R_k, v)] \& CPT(v, O_j, L_u) \\
\qquad \& CHG(v, O_j, L_u) \\
(no, v), else
\end{cases}
$$

其中, $CPT(v, O_j, L_u)$ 为真, 那么对任意的 $O_k \in H(O_j)$ 有 $L_u \triangleright f_o(S_{S(j)}) \& f_o(O_k) \triangleright L_u$; $CHG(v, O_j, L_u)$ 为真, 表示 S_i 可以改变 O_j 的安全级。

4 WBLP 安全模型在 Web 操作系统的应用

根据2.3中对开源Web OS的分析和总结, 其通用的架构采用的是基于 Linux 内核, 甚至是来自于 ASOP(Android Open Source Project)层级的安全模型[25]。由于移动终端硬件资源的特性和类 UNIX 操作系统的特点, Web OS 的文件系统采用 ext2、jfs、ext4 等, 其启动过程、进程、中断等与 Android 和 Linux 类似。下文首先将介绍模型元素与具体系统组件之间的对应关系, 然后描述安全公理和状态转换规则在系统中的对应关系, 最后指出最小权限原则, 在 Web OS 系统中的解释。

4.1 模型元素与 Web 操作系统的对应

在 Web 操作系统中, 进程是唯一的主体, 用 S 表示, 它可以在用户登录系统初启时或被其他进程创建。每个主体都有一个描述符段(Descriptor segment), 包含进程的相关信息, 以及进程当前访问客体的相关信息。一个进程被赋予一个唯一的进程标识符(PID)、用户标识符(UID)和用户组标识符(GID)。每个进程还被赋予相应的安全级标识, 用于强制存取控制检查。系统中的文件、目录、特殊文件、共享内存、消息、信号量、流、管道、进程都可以作为客体, 用 O 表示。对每一个客体, 在每个主体的描述符段中都有一个段描述符字(Segment descriptor word, SDW), SDW 包含了客体的名字, 执行客体的指针, 以用于读、执行、写的指示器(indicator)标志。

实际系统的访问权限集则由读、写、执行和-组成, 安全模型中的访问权限集与 BLP 一样, 由 e (执行)、r(读)、a (追加写)、w (写)和-(空)组成, 它们与模型元素的对应关系为:

表 1 模型与系统中访问权限对应关系

模型	系统
r	读
e, r	执行
w	读和写
a	写

访问矩阵 M 中的 a_{ij} 表示主体 S_i 对客体 O_j 具有的访问权限, 在实际系统当中, 这一信息保存在访问控制列表(ACL)中。ACL 存储在客体的父级目录中, 每个 ACL 表项都指定了一个进程和这个进程在该客体的访问权限。

在系统中, 主体和客体均具有一定的安全级别或密级, 在模型中抽象描述为 f, 由三个元素组成记作 $f = (f_s, f_o, f_c)$。主体的安全级别存放在进程级别表(process level table)和当前级别表(current level table), 而客体的安全级别存储在其父级目录中。模型中密级的集合为 C, 取值 $\{C_1, C_2, C_3, C_4\}$, 且 $C_1 > C_2 > C_3 > C_q$, 在实际系统的值是 C_1=Top Secret,

C_2=Secret, C_3=Confidential, C_4=Unclassified。Web 系统三种类型应用中, web 应用是分为本地存储和远程服务器端存储两种情况, 因此可信等级分类集合 I={*certified*, *privileged*, *web_local*, *web_remote*}。

模型中的系统状态由一个有序四元组 (b, M, f, H) 组成, 与实际系统的对应关系如表 2 所示:

表 2 状态元素与系统对应关系

模型元素	Web 操作系统中对应
b	描述符字段
M	访问控制列表
f	段目录信息和特定进程安全级表
H	分支

$b \subseteq (S \times O \times A)$ 表示在特定状态下, 哪些主体以何种访问属性访问哪些客体, b 存储在活动进程描述符段中的 SDW 中, 在活动段表中可找到活动进程。DSBR(descriptor segment base register)指针指向当前进程的描述符段。在模型中$(S_i, O_j, \underline{x})$表示主体 S_i 当前以 $\underline{x}$ 模式访问客体 O_j, 在实际的系统中, 该信息分别包含在 DSBR, 临时指针寄存器(Temporary pointer register, TPR)和 SDW 中, 描述符段包含进程的相关信息, 以及进程当前访问客体的相关信息。

模型的层次结构 H 与系统中树结构对应, 由文件系统目录表示。文件系统表示的客体还可以是文件、特殊文件、管道和目录, 它们的安全级等于其创建进程的安全级, 并且根据树形结构的特性, 目录的安全级等于或高于其父目录的安全级, 这样就维持了目录结构的不降级的特性。

WBLP 模型在 BLP 模型的基础上定义了一些新元素, 它们与 Web 系统中的对应关系如表 3 所示:

表 3 WBLP 新增元素与 Web 系统的对应

WBLP	系统
S_u, S_p, S_c	系统的三种类型应用: Certified、Privileged 和 Web。
O_u, O_p, O_s, O_v	根据不同保护级对存储空间的划分
Dom	用户(普通用户、特权用户、系统用户)到操作(普通用户操作、特权用户操作和系统用户操作)的一个映射关系
P	可信用户的权限集合
RA	可信用户权限对应的角色的集合
U_j	特权操作的集合, 又称为角色
u_i	某个特权操作

主体的安全级别存放在进程级别表和当前级别表中。对每一个客体, 在每个主体的描述符段中都有一个段描述符字 SDW, 其中包含了客体名称, 执行客体的指针, 以及用于读、执行、写的指示器标志。一个客体的安全级别或访问控制列表信息, 保存在客体的父级目录中, 因此, 改变一个客体的访问控制参数、创建或删除一个客体, 需要对其父级目录写或追加访问控制权限。而访问一个客体, 实际上进程需要从根目录遍历目录树到达目标客体。如果路径中有任意一个目录不能被这个进程访问, 那么目标客体也不能被访问。因此, 一个客体的安全级别能够支配其父级目录的安全级别, 而将客体放入一个更高安全级别的目录中是毫无意义的。

段目录中的一个分支对应一个客体, 该客体的信息由访问控制列表和物理地址等信息组成。对应关系如图 1 所示:

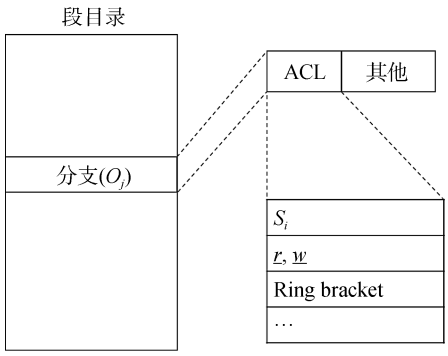

图 1 段目录

4.2 安全公理与 Web 操作系统的对应

1) 域间隔离性

根据模型的域间隔离公理, 在实际系统中, 将存储划分为四个区域: 普通用户区(Ordinary user area), 认证用户区(Authentic user area), 系统管理区(System management area)和病毒保护区(Virus protection area)。如图 2 所示, 普通用户区存储普通用户的数据和应用, 用户 Ua 可以进行读或写的操作。认证用户区存储的是经过系统认证的第三方应用, 存储较可信的用户(如 Ub)的数据和应用, 普通用户 Ua 对该区不具有写的权限。系统管理区不能被普通用户和认证用户读和写, 只有具有特殊权限的用户(U_m)才能进行修改。病毒保护区包含的数据、文件不能被用户空间的进程读写, 只能被系统空间的特殊用户(U_n)访问。利用上述隔离方法, 可将系统的可信计算基、审计信息等存储于病毒保护区, 以保护系统核心部件的安全, 使得系统更可信。

利用隔离机制将系统中的用户划分为两类, 可信用户(Us、U_m、U_n)和非可信用户(Ua、Ub)。可信用户运行在系统空间, 一般是系统管理员, 或系统服务、进程, 它们对系统管理区可读, 并且只有特定的可信用户(U_m、U_n)才具有写访问权限。非可信用

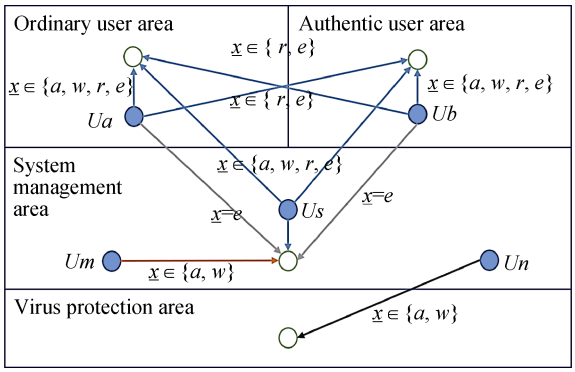

图 2　系统安全隔离域

户指普通用户应用, 位于用户空间, 一般指用户下载安装的第三方应用和经过认证的第三方应用。通过域间隔离公理, 限制不同类型应用之间的访问。同一隔离区域的用户之间, 通过安全级的判定和 ACL 的方式实现互相访问, 这种隔离强化了应用的沙箱机制。

2) 强制访问控制 MAC(Mandatory access control)

经典 BLP 模型通过简单安全性和*特性实现强制访问控制。类似的, 根据 WBLP 模型基本安全公理, 主体能只读或执行客体, 其安全级必须支配客体的安全级, 或主体是具有特权的(当客体是系统管理区中的客体)。若主体写访问客体, 主体的安全级必须等于客体的安全级或主体具有写特权(客体属于系统管理区中的客体)。这种对读写访问的强制性约束, 符合强制访问控制原则。

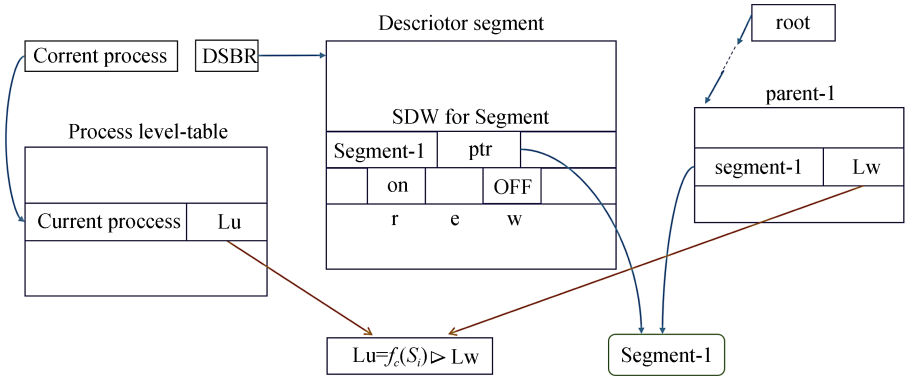

图 3　*特性公理实现 Web OS 中的只读访问

在模型中, *特性公理基于主体当前安全级值, 实现对访问状态(S, O, $\underline{x}$)的约束。在系统中, *特性公理用于活动的进程, 对非活动的进程, 通过简单安全性公理约束。对于一个活动进程的描述符段中的 SDW, 如果读(r)或者执行(e)指示器处于开启状态(ON)而写(w)指示器处于关闭状态(OFF), 那么进程的当前安全级支配段级别; 如果 r 指示器 OFF 而写指示器 ON, 那么进程的当前安全级就被段级别支配; 如果 r 和 w 指示器同时为 ON, 进程的当前安全级就等于段级别。

如图 3 所示, *特性对只读访问的约束在系统中实现为: 在当前活跃进程(由 current process 指向)的描述字段(由 DSBR 指向)的 SDW 中, 所要访问数据段(segment-1)的 r 指示器为 ON 和 w 指示器为 OFF, 且该进程的当前安全级 L_u, 必须支配该数据段的安全级(L_w), 这里, 该数据段的安全级保存在其父结点(parent-1)中。类似的, 读写访问的实现如图 4 所示, SDW 中读和写指示器均为 ON, 且进程的当前安全级等于数据段(segment-2)的安全级。

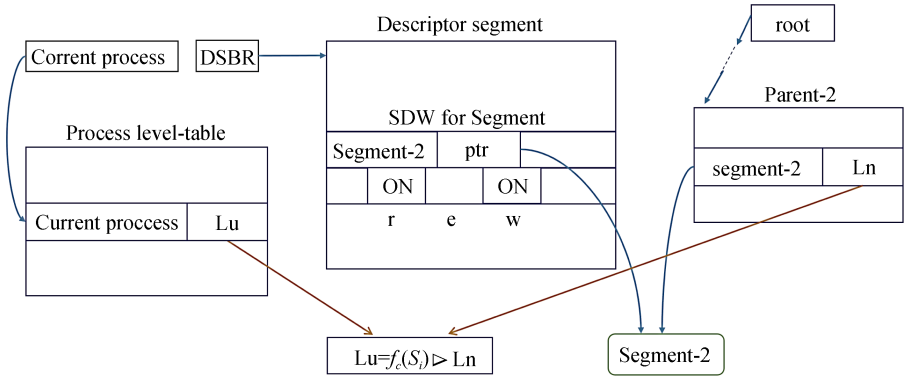

图 4　*特性公理实现 Web OS 中的读写访问

3) 自主访问控制 DAC(Discretionary access control)

根据 WBLP 模型自主安全公理, 每一个当前访问 $((S_i, O_j, \underline{x}) \in b)$由访问控制矩阵 M 判定。若进程以权限 $\underline{x}$ 访问客体, $\underline{x}$ 必须在客体对应的访问控制列表中, 即 $\underline{x} \in M_{ij}$。这种访问控制的实现方式与自主访问控制原则相符。如图 5 所示, 在实际系统中, SDW 中指示器为 ON 的访问方式, 要与 SDW 中 ptr 指向的数据段(segemt-3)的 ACL 指向相同的访问方式(数据段的 ACL 储存在其同一分支的父结点中)。如, a1=ON, 那么 b1=ON; a2=ON 则 b2=ON; a3=ON 则 b3=ON。特别的, 当(a1, a2, a3)=(ON, OFF, OFF)时, (b1, b2, b3)=(ON, ON, ON)也满足自主安全性。因为, 允许的最大访问权限不需要出现在 SDW, 而一个非活动进程是被匿名描述, 这样在激活时上述情况是成立的。

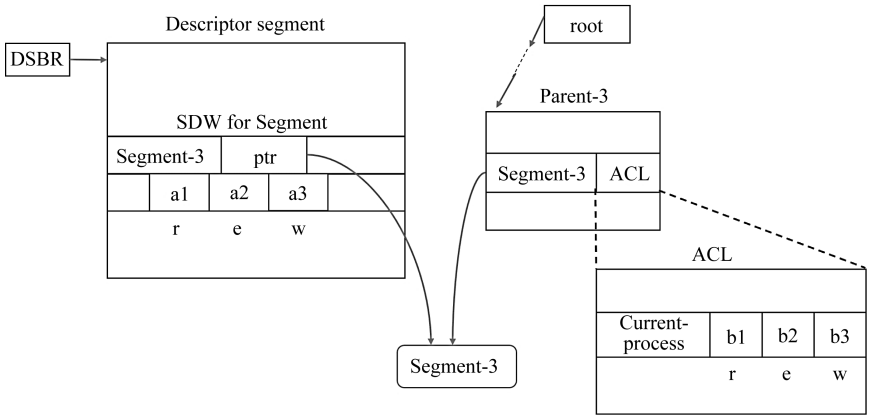

图 5 自主安全性公理在 Web OS 中的实现

4) 状态兼容性

模型还对系统的兼容性进行规定, 当主体创建文件类型的客体时, 客体的安全级等于其父目录的安全级; 当主体创建目录类型客体时, 客体的安全级支配其所在父目录的安全级。

4.3 状态转换规则与 Web 操作系统的对应

Web OS 的状态转换是由内核调用和其返回值定义的, 对于 WBLP 模型规则集 $\rho: R \times V \to D \times V$ 中的任一规则 ρ_i 有:

- 任意请求 $R_j \in R$ 表示一个指定的系统调用或可信进程调用, R 为所有系统调用和可信调用的集合, R_j 的输入参数则来自于当前系统状态 V。
- 任一判定 $D_m \in D = \{yes, no, ?, error\}$ 由一个系统调用或可信进程调用的返回值表示。若 $D_m \neq no$, $D_m \neq ?$, $D_m \neq error$, 则 R_j 输出一个新状态 v^*, 它将包含新的客体和一个新的客体结构, 同时也可以从以前状态中排除某些客体和访问权限。
- 规则 ρ 保持了系统的安全状态, 即当 v 是安全状态时, 那么 v^*是安全状态, 这由 WBLP 模型的安全公理及操作规则来保障。

1) 初始状态

操作系统的安全初始状态由一个初始化过程设置[32], 包括如下步骤: 系统的构造和生成, 包括审计机制、MAC 机制, DAC 机制和权限管理等的安装和初始化; 系统中用户安全文档的定义, 根据安全策略给每个用户赋予相应的安全级和角色; 系统中客体初始安全级设置, 即系统用户空间区、认证用户区、系统管理区、病毒防护区的划分及建立; 最后系统启动。

2) 状态转换规则在系统中的对应

模型的状态转换由一系列的内核调用和内核原语完成的。在模型中, 获取访问规则(get access)分解为获取只读访问、只写访问和执行访问, 但在实际系统中, 这一功能在某些情况下通过一个函数来实现: 当一个分段错误发生时, 如导致加载或存储, 一个 SDW 被创建, 可能导致该用户的 ACL 中所有指针(r, e, w)将都为 ON。

规则 ***R*1** 为获取只读访问, 在系统中实现为允许进程(PID)以只读方式访问数据段(Segment-ID, SID)的请求。这一过程中, 需要检查: ACL 中对应该 PID 的访问模式是否为 r; 安全级列表中 PID 的安全级是否支配 SID 的安全级; PID 是否是可信主体, 或 PID 的当前安全级支配 SID 的安全级; 且该数据段不能存储在病毒保护区 O_v 中。当满足以上条件时, SID 被加入到 PID 的段描述字段, 标记为只读, 并由一个 ptr 指向 SID。

类似的, ***R*2** 在系统中实现为允许进程(PID)以只写方式访问数据段(SID)的请求。这一过程中, 需要

检查只写访问是否存在于 ACL 中, 该进程是否为可信进程, 进程的安全级是否支配数据段安全级, 且数据段不能是系统管理区 O_s 和病毒保护区 O_v 中的。***R3*** 实现为允许进程(PID)执行数据段(SID)的内容。

对于请求读写访问(***R4***), 实现为先请求只读访问(***R1***), 当该请求为真后, 再请求只写访问(***R2***)。

主体 S_i 请求释放客体 O_j 的某个访问权限(***R5***), 在系统中实现为将 ptr 指向的 SDW 中的 *e* 指示器改为 OFF, 如果其他的指示器都为 OFF, 那么从 PID 的段描述符字段中删除该 SID, 如果请求不在定义范围内, 则不会发生状态的变化。

若一个进程(PID_1)给予另一个进程(PID_2)对某个数据段(SID)的访问, 该进程需要对数据段的上一层结点或根节点具有写权限, 并且 PID_1 的当前安全级支配 PID_2 安全级, 记为 ***R6***。规则 *R6* 在状态转换中信息流图走向如图 6 所示。

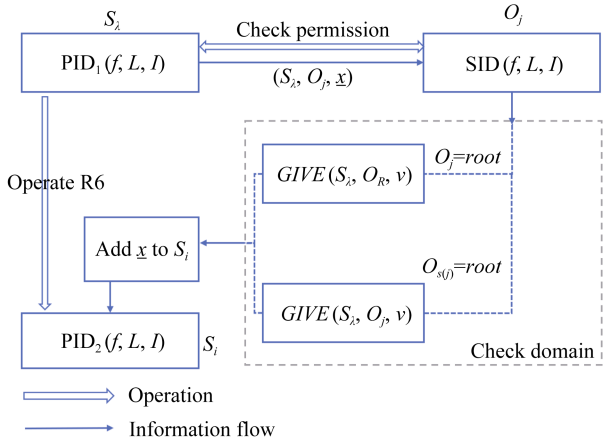

图 6　状态转换规则 *R6* 的信息流图

规则 *R6* 还可以采用如下方式描述:

if $R_k \notin Dom(R6)$ then $R6(R_k, v)$ =(?, v);

else if $[(S_\lambda \in S_c)\&(S_i \notin S_c)\&(f_c(S_\lambda) \triangleright f_s(S_i))]$and[<[$O_j \neq O_R$]and[$O_{s(j)} \neq O_R$]and[$O_{s(j)} \in b(S_\lambda: \underline{x})$]>or <[$O_{s(j)}=O_R$]&[$GIVE(S_\lambda, O_j, v)$]>

or<[$O_j=O_R$]& $GIVE(S_\lambda, O_R, v)$>]

then $R6(R_k, v)$ =(yes, (b, $M \backslash M_{ij} \cup \{x\}$, f, H));

else $R6(R_k, v)$ = (no, v);

end;

若一个进程(PID_1)撤销另一进程(PID_2)对数据段的访问属性, 该进程对数据段的上一层结点或根节点具有写权限, 并且进程的安全级支配 PID_2 的当前安全级, 记为 ***R7***, 其实现方式与 ***R6*** 类似。如果没有其他对该数据段的访问权限, 那么从 PID_2 的段描述符字段中删除对该 SID 的指向。

若一个进程(PID)创建一个数据段时, 实际上是创建一个叶子段, 需要满足: 该叶子段的父结点(SID)不能是系统管理区和病毒防护区中的; 描述符字段的 SDW 对应目录中 SID 的 *w* 指示器为 ON; 新叶子段的安全级支配 SID 的安全级(存储在其父目录中), 记为 ***R8***。

若一个进程(PID)删除一个非系统管理区和病毒防护区中的数据段(SID)时, 如果进程对该数据段的上一层结点具有写权限, 那么调用一个递归的函数: 设置当前数据段 ID(Current-Segment-ID, C_SID)为 SID, 如果 C_SID 没有分支叶子结点, 那么, 删除所有指向 C_SID 的 SDW, 然后在层次结构中删除 C_SID, 以及其父目录中对应的信息, 设置 C_SID 为删除结点的父结点(P_SID), 当 P_SID=SID 时, 循环结束, 操作完成, 记为 ***R9***。

R10 一个进程(PID)改变自己的当前安全级 $f_c(S)$ 为 L_u, 需要满足的条件是: 该进程的 PID 在系统特权进程列表中或每个描述符段中的 SDW 都为 PID, 并且进程安全级 $f_s(S)$支配 L_u; 如果 *r* 指示器 ON, 需要 L_u 支配 $f_c(S)$, 如果 *w* 指示器 ON, 那么 $f_c(S)$支配 L_u; 改变后的状态仍是安全状态, 并且改变后的安全级不高于其最大安全级。

一个进程(PID)改变某数据段(SID)的安全级为 L_n, 需要检查: 进程是否为可信进程, 并且其当前安全级支配数据段的安全级; 若 SDW 中对应对 SID 的 *r* 指示器为 ON, 则需进程的当前安全级支配 L_n; 若与 SID 对应的 *w* 指示器 ON, 其当前安全级受 L_n 支配; SID 的每个分支结点的安全级支配 L_n, 且 L_n 支配其父结点安全级, 那么允许进程改变该数据段安全级, 该操作记为 ***R11***。

4.4　最小权限原则在系统中的实现

最小特权在实际系统中实现为, 在进程被创建时, 根据该进程(PID)所属的类型(certified、privileged 或 web)只给予其最小的权限集合 $u_{\min}$。同时, 将系统可信用户(S_c)的权限集分解成 *n* 个特权角色{u_1, u_2, u_3, u_4, …, u_n}, 这些特权角色的集合, 共同完成原先由一个系统特权应用或进程完成的工作。在根据需要, 为特权用户集合(U_1, U_2, …, U_m)中的某个用户任意组合特权角色, 但每个特权用户只拥有能完成其工作所需的最小的特权角色组合, 即 $U_i = \min(\sum u_j \mid 1 \leqslant j \leqslant n), 1 \leqslant i \leqslant m$。

5　小结

本文通过对 BLP 模型进行分析, 并结合现有 Web 操作系统安全模型的特点和存在问题, 提出了针对 Web 操作系统的 WBLP 模型。通过对 BLP 模型

元素、安全公理和状态转换规则重定义, 在模型当中重新划分主体、客体的安全级, 增加可信级别标记和角色映射, 并针对现有的 Web 操作系统进行模型映射, 实现了最小权限原则、主体完整性约束和域间隔离机制, 以改进其机密性。最后阐述了 WBLP 模型应用到移动终端的 Web 操作系统的对应关系, 并说明模型在 Web 操作系统上的适用性。本文提出的对完整性的改进只针对主体, 未考虑主客体完整性的安全需求, 对主客体的完整性保护还需引入完整性模型, 机密性模型与完整性模型相结合的方法是下一步研究工作。同时, 利用最小权限原则的思想, 对可信用户中的 Linux 超级用户(root)权限进一步分解, 原先 root 用户完成的工作, 可实现为由几个特权用户共同完成, 进而实现 root 分权, 这样即使终端设备恶意软件获取最高权限后, 只能破坏系统的某一块功能, 不能影响整个设备。

参考文献

[1] He J. B., Qing S. H., and Wang C., "Analysis of Two Improved BLP Models," *Journal of Software*, vol.18, no.6, pp. 251-259. 2007.
(何建波, 卿斯汉, 王超, "对两个改进的 BLP 模型的分析," *软件学报*, 2007, 18(6), pp. 251-259。)

[2] FEIERTAG R.J., LEVITT K.N., and ROBINSON L., "Proving Multilevel Security of a System Design," *ACM SIGOPS Operating Systems Review*, vol.11, no.5, pp: 57-65, 1977.

[3] Bell D E. "Looking back at the Bell-La Padula model," *IEEE Annual Computer Security Applications Conference (ACSAC)*, vol.15, pp. -351, 2005.

[4] Doherty J., and S. P. Oriyano. "Wireless and Mobile Device Security." *Jones & Bartlett Pub*, 2015.

[5] 李慧云, 陆钢, 梁柏青等, "Web OS 现状和发展趋势分析," *信息通信技术*, 2014(2): 57-62。

[6] Bell, D.E., "Secure Computer Systems: A Refinement of the Mathematical Model," *Secure Computer Systems A Refinement of the Mathematical Model*, 1974.

[7] Mclean J., "A comment on the basic security theorem of Bell and LaPadula," *Information Processing Letters*, vol.20, no.85, pp. 67–70, 1985.

[8] Landwehr CE. Heitmeyer CL. and McLean J., "A security model for military message systems," *ACM Transaction on Computer System*, vol.2, no.3, pp.198-222, 1984.

[9] Shi W.C., Sun Y. F., and Liang H.L., "AN ADAPTABLE LABELING ENFORCEMENT APPROACH AND ITS CORRECTNESS FOR THE CLASSICAL BLP SECURITY AXIOMS," *Journal of computer research and development*, vol.38, no.11, pp.1366-1372, 2001.
(石文昌, 孙玉芳, 梁洪亮, "经典 BLP 安全公理的一种适应性标记实施方法及其正确性," *计算机研究与发展*, 2001, 38(11): 1366-1372。)

[10] Xu L., and Tan H., "Formal Description and Automated Verification of Improved BLP Model," *Computer Engineering*, vol.12, pp.130-135, 2013.
(徐亮, 谭煌, "BLP 改进模型的形式化描述及自动化验证, "*计算机工程*," 2013, (12): 130-135。)

[11] Liu W. Q., Qing. S.H., and Liu H. F., "Design of a Modified BLP Security Model and Its Application to SecLinux," *Journal of Software*, vol.13, no.4, pp.567-573, 2002.
(刘文清, 卿斯汉, 刘海峰, "一个修改 BLP 安全模型的设计及在 SecLinux 上的应用," *软件学报*, 2002, 13(4): 567-573。)

[12] Cai Y., Zheng Z.R., and Shen C. X., "A Planar Attributes Model Based on Multi Level Security Policy," *Chinese Journal of computers*, vol.27, no.5, pp. 619-624, 2004.
(蔡谊, 郑志蓉, 沈昌祥, "基于多级安全策略的二维标识模型," *计算机学报*, 2004, 27(5) : 619-624。)

[13] Hu Y.Q., Wu H.B., Yu H.Y., and Long R., et al., "Extended BLP Model and Its Application," *Computer Engineering*, vol.36, no.8, pp.123-125, 2010.
(胡勇强, 伍红兵, 俞海英等, "扩展的 BLP 模型及其应用,"*计算机工程*, 2010, 36(8): 123-125。)

[14] Y.H. Liu, and X. Chen, "A new information security model based on BLP model and Biba model," *Proceedings of International Conference on Signal Processing (ICSP)*, vol.3, pp.2643-2646, 2004.

[15] Liu Y.M., Dong Q.K., and Li X.P., "Study on enhancing integrity for BLP model," *Journal on Communications*, vol.31, no.2, pp:100-106, 2010.
(刘彦明, 董庆宽, 李小平, "BLP 模型的完整性增强研究," *通信学报*, 2010, 31(2): 100-106。)

[16] Zhang J., Zhou Z., Li J., and Liu Y., et al., "Confidentiality and integrity dynamic union model based on MLS policy". *Computer Engineering and Applications*, vol.44, no.12, pp.19-21, 2008.
(张俊, 周正, 李建等, "基于 MLS 策略的机密性和完整性动态统一模型," *计算机工程与应用*, 2008, 44(22): 19-21。)

[17] Biba K J., "Integrity Considerations for Secure Computer Systems," *Electronic Systems Div Air Force Hanscom Afb*, 1977.

[18] Liu B., Chen S.H., and Deng J.S., "Survey of Bell-LaPadula model," *Application Research of Computers*, vol.30, no.3, pp. 656-660, 2013.
(刘波, 陈曙晖, 邓劲生, "Bell-LaPadula 模型研究综述," *计算机应用研究*, 2013, 30(3): 656-660。)

[19] Zhang J., Xu L., and Meng Q.D., et al., "Confidentiality and integrity dynamic union model based on pre-authorization mechanisms," *Journal of* National University of Defense Technology, vol.36, no.1, pp.167-171, Feb. 2004.

(张俊, 徐鲁威, 孟庆德等, “基于预授权的机密性和完整性动态统一模型,” *国防科技大学学报*, 2014.2, 36(1), pp.167-171。)

[20] Fan Y.F., Han Z., Cao X., and He Y.Z., et al., “A Multilevel Security Model Based on Time Limit,” *Journal of computer research and development*, vol.47, no.3, pp.508-514, 2010.
(范艳芳, 韩臻, 曹香港等, “基于时间限制的多级安全模型,” *计算机研究与发展*, 2010, 47(3) : 508-514。)

[21] Defreez, Daniel, et al., “A First Look at Firefox OS Security,” *Computer Science*, 2014.

[22] Tor-Morten Grønli, Jarle Hansen, Gheorghita Ghine, and Muhammad Younas, “Mobile application platform heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS,” *IEEE 28th International Conference on Advanced Information Networking and Applications* (*AINA.2014*), pp.635-641, 2014.

[23] 杨彦格, 周晓龙, “Firefox OS 技术特征及优劣势浅析,” *移动通信*, 2014(3): 85-88。

[24] Piekarska M, Shastry B, and Borgaonkar R., “What Does the Fox Say? On the Security Architecture of Firefox OS,” *IEEE 2014 Ninth International Conference on Availability, Reliability and Security (ARES)*, pp.172-177, 2014.

[25] “Firefox OS security overview,” Mozilla, https://developer.mozilla.org/en-US/Firefox_OS/Security/Security_model, Feb 10, 2016.

[26] “Tizen Architecture,” wiki, https://wiki.tizen.org/wiki/Porting_Guide#Tizen_Architecture, 14 March 2016.

[27] Chen B., Ming W. S., and Huang Y. L., “An Anomaly Detection Module for Firefox OS,” *IEEE Eighth International Conference on Software Security and Reliability-Companion (SERE-C)*, pp. 176-184, 2014.

[28] Georgiev, Martin, S. Jana, and V. Shmatikov, “Rethinking Security of Web-Based System Applications.” *Proceedings of the 24th International Conference on World Wide Web (WWW 2015)*, pp. 366-376, 2015.

[29] Hu X., and Lu C., “A Solution for Management Security of Secret Information Based on Access Controlling.” *Computer technology and development*, vol.8, pp.131-134, 2014.
(胡欣杰, 路川, “基于访问控制的涉密信息管理安全解决方案, ” *计算机技术与发展*, 2014, 8:131-134。)

[30] Qing. S.H., and Shen C. X., “Design of high security level operating system,” *SCIENTIA SINICA Informationis*, vol.37, no.2, pp.238-253, 2007.
(卿斯汉, 沈昌祥, “高等级安全操作系统的设计,” *中国科学*, 2007, 37(2): 238-253。)

[31] Bell, D. Elliott, and L. J. L. Padula, “Secure Computer System: Unified Exposition and Multics Interpretation,” *Secure Computer System Unified Exposition & Multics Interpretation*, pp.161-161, 1976.

[32] Tanenbaum, Andrew S., “Modern Operating Systems, Third Edition,” *Pearson Education*, 2009. (Simplified Chinese edition by China Machine Press, 2014.)

朱大立 于 2007 年在华中科技大学获得计算机应用技术专业博士学位。现任中国科学院信息工程研究所正研级高级工程师, 博士生导师。研究领域移动互联网安全和无线网络攻防技术, 研究兴趣包括: 智能终端安全、应用安全、无线管控等技术。Email: zhudali@iie.ac.cn
Email: caochen11@mails.ucas.ac.cn

金昊 于 2013 年在合肥工业大学计算机科学与技术专业获得学士学位。现在中国科学院信息工程研究所通信与信息系统专业攻读博士学位。研究领域为移动互联网安全。研究兴趣包括: 移动恶意代码检测。Email: jinhao@iie.ac.cn

冯维淼 于 2008 年在北京大学计算机应用技术专业获得硕士学位, 现任中科院信息工程研究所四室工程师, 研究领域为移动安全, 研究兴趣包括木马攻防, 应用安全。Email: fengweimiao@iie.ac.cn

杨莹 于 2008 年在广西师范大学计算机应用技术专业获得工学硕士学位, 现在中国科学院信息工程研究所信息安全专业攻读博士学位。研究领域为信息安全。研究兴趣包括: 操作系统安全、移动智能终端安全。Email: yangying@iie.ac.cn

邵京 于2013年在北京理工大学电子信息科学与技术专业获得学士学位。现在中国科学院信息工程研究所信息安全专业攻读硕士学位。研究领域为移动互联网安全。研究兴趣包括: 移动网络安全、基于基站的手机管控技术。Email: shaojing@iie.ac.cn