第44卷第3期 南京航空航天大学学报 Vol. 44 No. 3
2012年6月 Journal of Nanjing University of Aeronautics & Astronautics Jun. 2012

一种三维度的秘密信息可信释放策略

朱 浩[1,2] 庄 毅[1] 薛 羽[1] 丁卫平[1,2] 梁惺彦[2]
(1.南京航空航天大学计算机科学与技术学院,南京,210016;2.南通大学计算机科学与技术学院,南通,226019)

摘要:秘密信息可信释放策略的研究目前主要集中在内容、时间、地点以及调用主体4个维度上,不同维度的策略侧重于解决可信释放的不同方面,具有一定的局限性。为了确保秘密信息的可信释放,需要综合考虑不同的维度。为此,提出了一种结合内容、地点和调用主体3个维度的可信释放策略。该策略的内容维度限制攻击者不能通过释放机制获取额外的秘密信息,地点维度控制秘密信息仅能在程序中特定语句点释放,而主体维度则限定攻击者不能影响秘密信息释放语句是否被调用执行。通过这3个维度的控制,该策略具有更细的控制粒度,能更好地抵抗信息清洗攻击。此外,建立了策略实施的类型系统,给出了类型系统的可靠性定理及其证明。

关键词:可信释放;秘密性;信息流;三维;无干扰
中图分类号:TP311 **文献标识码**:A **文章编号**:1005-2615(2012)03-0384-09

3-D Policy of Trusted Release of Confidential Information

Zhu Hao[1,2], *Zhuang Yi*[1], *Xue Yu*[1], *Ding Weiping*[1,2], *Liang Xingyan*[2]
(1. College of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing, 210016, China;
2. College of Computer Science and Technology, Nantong University, Nantong, 226019, China)

Abstract: Current study on trusted release policies of confidential information focused on WHAT, WHEN, WHERE and WHO dimensions. Each of them tends to address only one aspect of information release and has some limitations. Hence, it is desirable to combine defense along different dimensions. A trusted release policy combining WHAT, WHERE and WHO dimensions is proposed. The key idea of WHAT dimension of the policy is that attacker is not allowed to increase observations about confidential information by causing misuse of the declassification mechanism. WHERE dimension of the policy controls confidential information is declassified only through the declassification statement, and WHO dimension of the policy prevents the attacker from influencing whether confidential information is released. This release policy has finer granularity of controlling the release of confidential information and can resist the information laundering attack better. Additionally, the type rules are established and proved for the policy enforcement.

Key words: trusted release; confidentiality; information flow; 3-D; noninterference

秘密性是软件的可信性属性之一[1],它指低安全级别实体对高安全级别信息的访问是受限的,以防止不期望的高安全级别信息泄露到低安全级别实体。保护秘密性的常用方法之一是访问控制,它对访问包含秘密信息的文件的权限进行了限制,但是没有对信息的传播进行约束,从而会导致程序有意或无意地泄漏信息。例如登录口令检查程序,该程序需要拒绝错误的口令用户登录,但该拒绝行为却泄漏给试图登录者一定量的信息,登录者通过该信息能排除了一个错误的口令,从而缩小了口令搜

基金项目:航空科学基金(2010ZC13012)资助项目;江苏省普通高校研究生科研创新计划(CXLX11_0205)资助项目。
收稿日期:2011-05-27;**修订日期**:2012-02-22
通讯作者:庄毅,女,教授,博士生导师,1956年生,E-mail:zhuangyi@263.net。

索空间，因此访问控制技术不能完全确保端到端的秘密性；另外，加密技术能从一定程度上确保秘密信息安全地从发送方通过传输信道到达接受方，但是不能保证一旦数据在接收方解密后，计算时仍能保持该数据的秘密性，因此加密技术也不能完全确保端到端的秘密性。为了克服上述问题，近年来，研究人员广泛采用基于语言的信息流策略来研究秘密性问题[2]。

无干扰[3-4]是一个非常重要的信息流策略，它的主要思想就是低安全级别的输出不依赖于高安全级别的输入，使得攻击者无法从低安全级别的输出中推导出高安全级别的输入。但是无干扰策略的约束条件太强，以至于很多软件在遵循该策略的情况下将没有实用意义，例如登录口令检查程序不可避免地违反无干扰策略，因此需要对无干扰的约束条件进行弱化或放松，研究人员提出了若干放松无干扰的策略[5-7]。放松的无干扰策略允许软件系统中存在必要的秘密信息释放，但是需要制定安全策略对秘密信息的释放通道加以控制，否则攻击者就可能滥用它来进一步释放更多的违反安全需求的秘密信息。

目前秘密信息可信释放策略的研究主要集中在 4 个维度[8]：内容、时间、地点以及调用主体。不同维度的策略仅仅阐述了信息释放的不同方面，有一定的局限性。文献[9]提出了一种称为“定界”的释放策略，它从信息释放的内容维度对秘密信息的正常释放和非法释放进行“定界”，但只能识别出针对内容维度的攻击，而不能检测其他维度的攻击。文献[10]针对“定界”策略只考虑了内容维度的局限性，提出了结合内容和地点维度的“定位定界”释放策略，对秘密信息的释放点进行“定位”，能识别出更多的攻击。文献[11,12]则提出基于调用主体和内容维度的健壮性策略，限定攻击者不能影响秘密信息是否被释放以及释放的内容，但该策略忽略了其他维度的攻击。为了避免秘密信息的非法释放，释放策略应该综合考虑多个维度，为此，本文基于“定位定界”策略和“健壮性”策略各自的局限性，提出了一种综合考虑了内容、地点和调用主体 3 个维度的秘密信息可信释放策略，进一步提高了策略的安全性。

1　语言模型

1.1　语法和语义

秘密信息释放策略所研究的程序语言是一种安全类型的顺序式命令语言：其中每个常量和变量都具有一个安全级别 l。基于秘密性和完整性是密切联系的，本文的安全级别 l 同时考虑了这两个属性，用一个二元组描述为 $l=(S(l),I(l))$，其中 $S(l)$ 表示秘密性级别，$I(l)$ 表示完整性级别。语言语法由表达式 e 和命令 c 组成：

$$e::=n \mid v \mid e_1 \mathrm{op}\ e_2 \mid \mathrm{declassify}(e,l)$$

$$c::=\mathrm{skip} \mid v:=e \mid c_1;c_2 \mid \mathrm{if}\ e\ \mathrm{then}\ c_1 \mathrm{else}\ c_2 \mid \mathrm{while}\ e\ \mathrm{do}\ c$$

式中：n 为常量，v 为变量，用 $\mathrm{Vars}(e)$ 表示表达式 e 中变量的集合；op 是二元的算术或逻辑运算符；l 是安全级别，安全环境是变量到安全级别的映射，例如表示变量的安全级别，本文安全级别 l 中 $S(l)$ 和 $I(l)$ 分别考虑两级：高安全级别 H 和低安全级别 L；释放表达式 $\mathrm{declassify}(e,l)$ 是将表达式 e 的安全级别降低到 l，其中表达式 e 称为被释放表达式，$\mathrm{declassify}(e,l)$ 不允许嵌套使用。命令语法中包括了标准的原子命令和顺序命令 c_1,c_2。

在表达式语义中，表达式配置形式为 $\langle e,m,E\rangle$，其中 e 为表达式，m 为存储（存储是变量到值的映射），E 为被释放表达式集合，即通过 $\mathrm{declassify}(e,l)$ 释放秘密信息的表达式 e 的集合；在不需要考虑 E 时可以省略成 $\langle e,m\rangle$ 形式。表达式求值规则具有形式 $\langle e,m,E\rangle \downarrow \langle n,E'\rangle$，其中 $m(e)=n$。具体表达式语义如图 1 所示。

$$\text{(S-CON)}:\quad \langle n,m,E\rangle \downarrow \langle n,E\rangle$$

$$\text{(S-VAR)}:\quad \langle v,m,E\rangle \downarrow \langle m(v),E\rangle$$

$$\text{(S-OP)}:\quad \frac{\langle e_i,m,E\rangle \downarrow \langle n_i,E_i\rangle}{\langle e_1 \mathrm{op} e_2,m,E\rangle \downarrow \langle \mathrm{op}(n_1,n_2),E_1 \cup E_2\rangle}$$

$$\text{(S-DECL)}:\quad \frac{\langle e,m,E\rangle \downarrow \langle n,E'\rangle}{\langle \mathrm{declassify}(e,l),m,E\rangle \downarrow \langle n,E' \cup \{e\}\rangle}$$

图 1　表达式语义

在命令语义中，命令配置具有形式为 $\langle c,m,E\rangle$，其中 c 是命令，m 和 E 与表达式配置的解释相同，在不需要考虑 E 时可以省略成 $\langle c,m\rangle$ 形式。命令配置的单步转移关系具有形式 $\langle c,m,E\rangle \rightarrow \langle c',m',E'\rangle$，一个特殊单步转移是 $\langle c,m,E\rangle \rightarrow \langle \mathrm{stop},m',E'\rangle$，表示单步转移后程序终止。用 $\rightarrow^*$ 表示 $\rightarrow$ 的自反传递闭包，表示零步或多步转移。迹 $\mathrm{Trace}(\langle c,m,E\rangle)$ 表示从 $\langle c,m,E\rangle$ 起始的命令配置转移序列。$\langle c,m,E\rangle \Downarrow \langle m',E'\rangle$ 表示 $\langle c,m,E\rangle \rightarrow^* \langle \mathrm{stop},m',E'\rangle$，当不考虑终止配置时，可简写为 $\langle c,m,E\rangle \Downarrow$；当仅考虑迹中存储变化时，可从配置转移序列中依次抽取存储构成序列 $[m,m',\cdots]$ 来表示迹 $\mathrm{Trace}(\langle c,m,E\rangle)$。具体命令语义如图 2 所示，其中 $m[x \mapsto n]$ 表示仅将存储 m 中的变量 x 的值更新为 n。

$$\text{(S-SKIP)}:\langle \text{skip},m,E\rangle \rightarrow \langle \text{stop},m,E\rangle$$

$$\text{(S-ASSIGN)}:\frac{\langle e,m,E\rangle \downarrow \langle n,E'\rangle}{\langle x:=e,m,E\rangle \rightarrow \langle \text{stop},m[x\mapsto n,]E'\rangle}$$

$$\text{(S-SEQ)}:\frac{\langle c_1,m,E\rangle \rightarrow \langle c'_1,m',E'\rangle}{\langle c_1;c_2,m,E\rangle \rightarrow \langle c'_1;c_2,m',E'\rangle}$$

$$\text{(S-IF1)}:\frac{\langle e,m,E\rangle \downarrow \langle n,E'\rangle n\neq 0}{\langle \text{if } e \text{ then } c_1 \text{ else } c_2,m,E\rangle \downarrow \langle c_1,m,E'\rangle}$$

$$\text{(S-IF2)}:\frac{\langle e,m,E\rangle \rightarrow \langle n,E'\rangle}{\langle \text{if } e \text{ then } c_1 \text{ else } c_2,m,E\rangle \rightarrow \langle c_2,m,E'\rangle}$$

$$\text{(S-WHILE1)}:$$

$$\frac{\langle e,m,E\rangle \downarrow \langle n,E'\rangle n\neq 0}{\langle \text{while } e \text{ do } c,m,E\rangle \rightarrow \langle c;\text{while } e \text{ do } c,m,E'\rangle}$$

$$\text{(S-WHILE2)}:\frac{\langle e,m,E\rangle \downarrow \langle 0,E'\rangle}{\langle \text{while } e \text{ do } c,m,E\rangle \rightarrow \langle \text{stop},m,E'\rangle}$$

图2 命令语义

1.2 安全格

程序语言中数据的安全级别形成一个安全格 $L=(L_S, L_I)$，它是秘密性安全格 L_S 和完整性安全格 L_I 构成的一个二元组。L_S 上的偏序关系为 $\subseteq_S$，$S\subseteq_S S'$ 表示在秘密性程度上，S 小于等于 S'；L_I 上的偏序关系为 $\subseteq_I$，$I\subseteq_I I'$ 表示在完整性程度上，I' 小于等于 I；L 上的偏序关系为 $\subseteq$，$l_1=(S(l_1),I(l_1))$，$l_2=(S(l_2),I(l_2))$，$l_1\subseteq l_2$ 当且仅当 $S(l_1)\subseteq S(l_2)$ 且 $I(l_1)\subseteq I(l_2)$ 同时成立。实际上，这里的偏序关系表达的是不同安全级别的数据在使用时受到的限制程度之间的关系。高秘密性数据在使用时受到的限制性要比低秘密性数据强，这有利于防止信息的泄漏；低完整性数据在使用时受到的限制性要比高完整性数据强，这有利于防止信息的破坏。图3描述了具有两级安全级别（H 和 L）的安全格 L_{LH}。令集合 $SC=\{L,H\}$，$(SC,\subseteq)$ 构成安全格，$(SC,\bigcup,\bigcap)$ 是格 $(SC,\subseteq)$ 诱导的代数系统，其中，$\bigcup$ 和 $\bigcap$ 是定义在 SC 上的两个二元运算：$\forall\ \sigma_1,\sigma_2\in SC$，$\sigma_1\bigcup\sigma_2$ 表示 σ_1 与 σ_2 的最小上界，$\sigma_1\bigcap\sigma_2$ 表示 σ_1 与 σ_2 的最大下界。例如，表达式 $x+y$ 的秘密性级别为 $\Gamma(x)\bigcup\Gamma(y)$。

程序中的信息流分为显式流和隐式流。显式流是由赋值语句引起的，比如 $x:=y$ 语句，变量 y 的信息直接流入了变量 x，这个显式流是合法的当且仅当 $\Gamma(y)\subseteq\Gamma(x)$ 成立。隐式流是由程序的控制结构引起的，例如 if b then $x:=1$ else $x:=0$，条件表达式 b 的信息间接地流入分支语句中被赋值变量 x，这个隐式流是合法的当且仅当 $\Gamma(b)\subseteq\Gamma(x)$ 成立；再如，语句 if $x>y$ then $z:=w$ else $i:=i+1$ 中的隐式流是合法的当且仅当 $(\Gamma(x)\bigcup\Gamma(y))\subseteq(\Gamma(z)\bigcap\Gamma(i))$。循环结构也存在类似的隐式流。

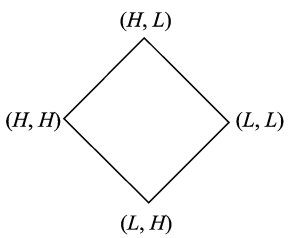

图3 安全格 L_{LH}

2 三维度的释放策略

2.1 攻击模型

任何一种安全策略都是针对某种攻击模型提出的。假设攻击者能够读取的信息秘密性级别不超过 S_A，能修改的信息的完整性级别不超过 I_A，那么攻击者的攻击能力可以被描述为安全格中的一个元素 A，用元组 $(S(A),I(A))$ 表示。基于该元素 A，定义高秘密性区域 $H_S=\{l\,|\,S(A)\subset S(l)\}$，低机密性区域 $L_S=\{l\,|\,S(l)\subseteq S(A)\}$，高完整性区域 $H_I=\{l\,|\,I(l)\subset I(A)\}$ 以及低完整性区域 $L_I=\{l\,|\,I(A)\subseteq I(l)\}$，由此获得4个区域 $LH=L_S\bigcap H_I$，$HH=H_S\bigcap H_I$，$LL=L_S\bigcap L_I$，$HL=H_S\bigcap L_I$。这4个区域如图4所示。对攻击者而言，LH 区域中的数据可以观察但是不能修改，HH 区域的数据不可观察也不能修改，LL 区域的数据可以观察也能修改，HL 区域的数据不能观察但能修改。被动攻击者只能观察安全格中 LL 和 LH 区域中的数据；主攻攻击者除了能观察到 LL 和 LH 区域中的数据外，还能修改 HL 和 LL 区域的数据。

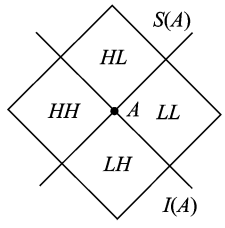

图4 攻击者视图

攻击者通过对程序的观察以及向程序中注入或修改代码来改变程序的可视行为，从而非法获取秘密信息。假设攻击代码满足下列3个条件：(1)不能包含释放表达式 declassify(e,l)，否则攻击者能通过它释放任意秘密信息；(2)不存在从高秘密性区域 H_S 流向低秘密性区域 L_S 的信息流，否则这种攻击代码非常容易被识别；(3)不能修改高完整区域 H_I 中的变量。把满足上述3个限定条件的较隐蔽的攻击称为“信息清洗攻击”。对于这种攻击，需要通过综合考虑不同的维度，制定和实施限制性更强的信息可信释放策略才能识别出来。

本文中秘密性主要针对程序中的变量，而完整

性则涉及到两个方面：程序中的变量和程序代码本身。程序中的高完整性代码是攻击者无法修改的，低完整性代码是攻击者有能力修改的。为了区别表示这两者，把程序中的高完整性代码原样保留，各个低完整性代码区分别用[·]代替；程序中可能存在多处低完整性代码区，那么程序的形式变成了高完整性代码中夹杂了若干低完整性代码区[·]；程序中所有[·]可以用向量形式$[\bar{\cdot}]$表示，把$c[\bar{\cdot}]$称为程序c的高完整性语境，形式化描述如下：

$c[\bar{\cdot}]::=\text{skip}\mid v:=e\mid c_1$

$c_2\mid \text{if } e \text{ then } c_1 \text{ else } c_2\mid \text{while } e \text{ do } c\mid[\cdot]$

可用任意的低完整性代码$[a]$来填充[·]；如果用低完整性代码向量$[\bar{a}]$填充$[\bar{\cdot}]$，则可得到一个完整的程序$c[\bar{a}]$。对于被动攻击者，不能修改低完整性代码，只具有观察能力；对于主动攻击者，可用“信息清洗攻击”的代码填充$[\bar{\cdot}]$。下面是一个信息清洗攻击的例子。在本文后续的所有例子中，变量的下标分别表示其秘密性与完整性级别，如X_{LH}表示变量X的秘密性级别为L，完整性级别为H。考虑程序c_1：

$[\cdot];\ Y_{LL}:=\text{declassify}((S_{1HL}+S_{2HL}+S_{3HL}+S_{4HL})/4,\ LL)$

该程序本意是释放秘密性变量$S_{1HL}\sim S_{4HL}$的平均值到公开变量Y_{LL}中，除此之外不释放任何更多的关于变量$S_{1HL}\sim S_{4HL}$的秘密信息，但是由于变量$S_{1HL}\sim S_{4HL}$是低完整性变量，并且在释放语句前存在完整性代码区[·]，那么主动攻击者可能在[·]中注入如下信息清洗攻击代码

$S_{2HL}=S_{1HL},\ S_{3HL}=S_{1HL},\ S_{4HL}=S_{1HL}$

那么程序c_1执行完后变量Y_{LL}的值等于变量S_{1HL}的值，从而导致变量S_{1HL}完全泄漏到公开变量Y_{LL}。

2.2　不可区分关系

信息释放策略建立在攻击者对系统状态的区分能力的基础上，而这种区分能力是通过存储和配置上的不可区分关系来表述的。存储上的最简单的不可区分关系是$=_{S(l)}$，对于存储m_1,m_2以及安全级别l，$m_1=_{S(l)}m_2$表示$\forall v.\ S(\Gamma(v))\subseteq_{S(l)}\Rightarrow m_1(v)=m_2(v)$；存储$m$的$_{S(l)}$限制记为$m|_{S(l)}$，表示将存储$m$限制在仅对秘密性级别不高于$_{S(l)}$的变量进行映射，从而可知$m_1=_{S(l)}m_2$等价于$m_1|_{S(l)}=m_2|_{S(l)}$。另外一种存储上的不可区分关系是基于被释放表达式集合E的不可区分关系$I(E)$，对于$m_1I(E)m_2$，表示$\forall\ e\in E, m_1(e)=m_2(e)$。下面定义配置上的不可区分关系。

定义 1（迹投影）：令$_{S(l)}$为秘密性级别，迹t为存储序列$[m_1,\cdots,m_i,\cdots]$，则迹t的$_{S(l)}$投影$t|_{S(l)}=[m_1|_{S(l)},\cdots,m_i|_{S(l)},\cdots]$；两个迹$t_1$和$t_2$的$_{S(l)}$投影相等：$t_1|_{S(l)}=t_2|_{S(l)}$，记为$t_1\sim_{S(l)}t_2$。

定义 2（迹不可区分）：当迹t_1和t_2都是可终止时满足$t_1\sim_{S(l)}t_2$，称迹t_1和t_2对于秘密性级别$_{S(l)}$是不可区分的，记为$t_1\approx_{S(l)}t_2$。

定义 3（配置不可区分）：如果成立$\text{Trace}(\langle c_1,m_1\rangle)\approx_{S(l)}\text{Trace}(\langle c_2,m_2\rangle)$，称配置$\langle c_1,m_1\rangle$和$\langle c_2,m_2\rangle$是对于秘密性级别$_{S(l)}$是弱不可区分的，记为$\langle c_1,m_1\rangle\approx_{S(l)}\langle c_2,m_2\rangle$；如果$\langle c_1,m_1\rangle\Downarrow$、$\langle c_2,m_2\rangle\Downarrow$且$\langle c_1,m_1\rangle\approx_{S(l)}\langle c_2,m_2\rangle$，则称配置$\langle c_1,m_1\rangle$和$\langle c_2,m_2\rangle$是对于秘密性级别$_{S(l)}$是强不可区分的，记为$\langle c_1,m_1\rangle\approx_{S(l)}\langle c_2,m_2\rangle$。

定义 4（互模拟）：在命令配置$\langle c,m,E\rangle$集合上的对称关系R_{i_1,i_2}是关于初始存储i_1和i_2上的低秘密性级别L的互模拟关系（记作$\sim_{i_1,i_2}$，简称关于i_1和i_2的互模拟）需满足以下条件：如果$\langle c_1,m_1,E_1\rangle\Downarrow$，$\langle c_2,m_2,E_2\rangle\Downarrow$以及$\langle c_1,m_1,E_1\rangle R_{i_1,i_2}\langle c_2,m_2,E_2\rangle$成立，则：

（1）$i_1I(E_1)i_2$当且仅当$i_1I(E_2)i_2$；

（2）如果$i_1I(E_1)i_2$成立，则：①$m_1=_Lm_2$；②如果$\langle c_1,m_1,E_1\rangle\rightarrow\langle c'_1,m'_1,E'_1\rangle$，则$\langle c_2,m_2,E_2\rangle\rightarrow*\langle c'_1,m'_1,E'_1\rangle R_{i_1,i_2}\langle c'_2,m'_2,E'_2\rangle$。

2.3　策略定义

基于健壮性策略[11-12]和定位定界策略[10]各自的局限性，下面提出了一种结合内容、地点与调用主体维度的释放策略。

定义 5（健壮的定位定界策略）：假设攻击者的攻击能力为$(S(A),I(A))$，对于任意存储m_1和m_2以及低完整性代码向量$[\bar{a}_1]$和$[\bar{a}_2]$，如果同时满足下列两个安全条件，则$c[\bar{\cdot}]$满足健壮的定位定界的策略。

（1）$\langle c[\bar{a}_1],m_1\rangle\approx_{S(A)}\langle c[\bar{a}_1],m_2\rangle\Rightarrow\langle c[\bar{a}_2],m_1\rangle\approx_{S(A)}\langle c[\bar{a}_2],m_2\rangle$；

（2）$m_1=_Lm_2\Rightarrow\langle c[\bar{\cdot}],m_1,\varnothing\rangle\sim_{m_1,m_2},\varnothing\rangle$；

由于本文忽略了时间和终止隐通道，所以仅当配置不能构成弱不可区分关系时，信息才会泄漏，但是由于可能存在不够熟练的攻击者在[·]插入非终止代码而获得比被动攻击者还少的信息，本文忽略这种情况，所以在定义 5 的安全条件（1）的前

提中采用配置的强不可区分关系。另外,上述定义分别从低完整性代码和高完整性代码两个层次进行控制:首先,安全条件(1)从调用主体维度和内容维度对低完整性代码进行约束:限制它不能影响高完整性代码中的释放语句是否执行以及释放内容。由于这里的策略都是针对"信息清洗攻击",所以在低完整性代码中不可能存在从高秘密性区域流向低秘密性区域的信息流,从而在低完整性代码部分不需要考虑地点维度的限制。其次,对高完整性代码而言,由于攻击者无法修改代码,所以不需要考虑调用主体维度的限制,但是可能存在程序设计者无意疏忽或有意违反,使得高完整性部分的代码存在秘密信息从地点和内容维度的非法释放,因此在高完整性代码部分,需要对没有考虑到的信息释放的内容维度和地点维度进行考虑。安全条件(2)限制了高完整代码中不允许出现秘密信息在非法的释放点释放,以及不能在释放点释放超额的秘密信息。下面通过例子对该策略的安全性进行说明,考虑程序c_2:

if X_{LH} then $Y_{LH}:=$ declassify(Z_{HH}, LH) + T_{HH} else skip;

直观上,这个程序是不安全的,原因是当X_{LH}值为真时执行赋值语句,流向变量Y_{LH}的信息包含两部分:一部分是通过释放表达式declassify合法释放的,而另一部分是变量T_{HH}的信息,这部分信息不是通过释放表达式合法释放的。形式上,对于秘密性级别L,设存储m_1和m_2满足$m_1=Lm_2$并且

$$m_1(X_{LH})=m_1(Y_{LH})=1 \quad m_1(Z_{HH})=5$$
$$m_1(T_{HH})=6$$
$$m_2(X_{LH})=m_2(Y_{LH})=1 \quad m_2(Z_{HH})=5$$
$$m_2(T_{HH})=8$$

令c'_2表示$Y_{LH}:=$declassify(Z_{HH}, LH)+T_{HH},则c_2在存储m_1和m_2单步转移可分别表示为:

$$\langle c_2,m_1,\varnothing\rangle \rightarrow \langle c'_2,m_1,\varnothing\rangle,\langle c_2,m_2,\varnothing\rangle \rightarrow \langle c'_2,m_2,\varnothing\rangle,$$

这步转移并没有引起存储状态和被释放表达式集合的变化,c'_2的下一步转移可分别表示为

$$\langle c'_2,m_1,\varnothing\rangle \rightarrow \langle \mathrm{stop},m'_1,Z_{HH}\rangle,\langle c'_2,m_2,\varnothing\rangle \rightarrow \langle \mathrm{stop},m'_2,Z_{HH}\rangle$$

式中:$m'_1=m_1[Y_{LH}\rightarrow 11]$,$m'_2=m_2[Y_{LH}\rightarrow 13]$,$m'_1\neq L,m'_2$,从而不满足互模拟的条件,即不满足定义5的安全条件(2),从而健壮的定位定界策略判定该程序是不安全的。但是,健壮性策略认为:c_2的条件判断表达式和分支语句中的被释放表达式中的变量都是高完整性的,那么攻击者不能影响释放表达式所在语句是否执行以及被释放的内容,所以判定c_2是安全的。由此可见,健壮性策略具有忽略地点维度的局限性。如果程序c_2改成:

[·]; if X_{LH} then $Y_{LH}:=$ declassify(Z_{HH} + T_{HH}, LH) else skip

则满足定义5的两个安全条件,健壮的定位定界策略判定该程序为安全的。

考虑另外一个程序c_3:

[·]; if X_{LL} then $Y_{LH}:=$ declassify(Z_{HH}, LH); else $Y_{LH}:=$ declassify(T_{HH}, LH)。

直观上,这个程序是不安全的,原因在于该程序中条件判断表达式X_{LL}是低完整变量,那么攻击者可能在低完整性代码区[·]注入攻击代码修改X_{LL}的真假值,从而影响包含释放表达式的不同分支的执行。形式上,对于秘密性级别L,设存储m_1和m_2满足:

$$m_1(X_{LL})=m_1(Y_{LH})=1 \quad m_1(Z_{HH})=5$$
$$m_1(T_{HH})=6$$
$$m_2(X_{LL})=m_2(Y_{LH})=2 \quad m_2(Z_{HH})=5$$
$$m_2(T_{HH})=8$$

设低完整性代码区的攻击代码$[a_1]$为:skip,则可推出$\langle c_3[a_1],m_1\rangle\approx_L\langle c_3[a_1],m_2\rangle$;设攻击代码$[a_2]$为:$X_{LL}:=0$,则可推出$m_1(Y_{LH})=6\neq m_2(Y_{LH})=8$,从而$\langle c_3[a_1],m_1\rangle\approx L\langle c_3[a_1],m_2\rangle$不成立,因此不满足定义5的安全条件(1),健壮的定位定界策略判定该程序是不安全的。但是定位定界策略却判定该程序是安全的,原因是它具有忽略了主体维度的局限性。如果将程序c_3中的条件判断表达式X_{LL}改成X_{LH},其他代码原样不变,那么修改后的程序满足定义5的两个安全条件,健壮的定位定界策略将判定该程序为安全的。

通过上述例子分析得出,健壮的定位定界策略是从不同层次对健壮性策略和定位定界策略进行了集成,具有更细的控制粒度,能识别出原来的策略无法识别的攻击。

3 策略的实施

3.1 类型规则

本节给出了一个实施健壮的定位定界策略的类型规则,该规则如图5所示。表达式规则的一般形式:$\Gamma\vdash e:l,D$,它的含义是在安全环境Γ下,e是

安全级别为l的良类型(well-typed)表达式,D表示一个集合,它包含的是e中可能存在的被释放表达式中的变量。命令类型规则的一般形式为$\Gamma, pc \vdash c : U, D$,它表示在安全环境和程序计数器标签($pc$)下命令$c$是良类型,$c$的执行产生了效果$U$和$D$,分别表示在$c$执行到当前程序点时所有在高完整性语境$c[\cdot]$中被更新的高完整性变量的集合($U$)和所有被释放表达式中的变量的集合($D$)。为了跟踪程序中的隐式流,引入了$pc$,它的取值是一个安全级别,表示所有影响当前程序点的隐式流的安全级别的最小上界。比如程序:

$$\cdots;\text{if } h \text{ then } x := 1 \text{ else } x := 0;\cdots$$

式中:if语句中条件表达式h的信息隐式地流入了各个分支命令($x:=1$或$x:=0$),所以各个分支命令的pc值至少为$\Gamma(h)$,但是可能存在if语句嵌套if语句或if语句嵌套循环语句,那么流入最内层if语句各个分支的隐式流的数据源有多个,分别是嵌套的if语句或循环语句的各个条件表达式。最内层的if语句各个分支命令的pc值是最外层的pc值与这些条件表达式的安全级别的最小上界。上述程序中if语句各个分支的pc值是if语句的pc值与if条件表达式h的最小上界。如果类型规则能应用到程序的每个表达式和命令上,那么类型规则判定该程序是良类型,即是安全的程序。下面对规则作较详细的阐述。

(T-CON):$\Gamma \vdash n : l, \varnothing$　(T-VAR):$\Gamma \vdash v : \Gamma(v) : \varnothing$

(T-OP):$$\frac{\Gamma \vdash e_1 : l_1, D_1 \quad \Gamma \vdash e_2 : l_2, D_2}{\Gamma \vdash e_1 \text{ op } e_2 : l_2 \cup l_2, D_1 \cup D_2}$$

(T-DECL):$$\frac{\Gamma \vdash e : l_1, D \quad D = \varnothing}{\Gamma \vdash \text{declassify}(e, l_2) : l_2, \text{Vars}(e)}$$

(T-SKIP):$\Gamma, pc \vdash \text{skip} : \varnothing, \varnothing$

(T-ASSI):

$$\frac{\Gamma \vdash e : l, D \quad l \cup pc \subseteq \Gamma(x) \quad \forall\, x \in D.\, \Gamma(x) \subseteq (H, H) \quad D \neq 0 \Rightarrow pc \subseteq (L, H) \quad I(v) = H \Rightarrow U = \{v\}}{\Gamma, pc \vdash v := e : U, D}$$

(T-SEQ):

$$\frac{\Gamma, pc \vdash c_1 : U_1, D_1 \quad \Gamma, pc \vdash c_2 : U_2, D_2 \quad U_1 \cap D_2 = \varnothing}{\Gamma, pc \vdash c_1 c_2 : U_1 \cup U_2, D_1 \cup D_2}$$

(T-IF):

$$\frac{\Gamma \vdash e : l, D \quad \Gamma, l \cup pc \vdash c_1 : U_1, D_1 \quad l \cup pc \vdash c_2 : U_2, D_2 \quad \forall\, x \in D.\, \Gamma(x) \subseteq (H, H) \quad D \neq \varnothing \Rightarrow pc \subseteq (L, H)}{\Gamma, pc \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : U_1 \cup U_2, D \cup D_1 \cup D_2}$$

(T-WHILE):

$$\frac{\Gamma \vdash e : l, D \quad \Gamma, l \cup pc \vdash c : U_1, D_1 \quad U_1 \cap (D \cup D_1) = \varnothing \quad \forall\, x \in D.\, \Gamma(x) \subseteq (H, H) \quad D \neq \varnothing \Rightarrow pc \subseteq (L, H)}{\Gamma, pc \vdash \text{while } e \text{ do } c : U_1 D \cup D_1}$$

(T-HOLE):$$\frac{S(pc) \subseteq L}{\Gamma, pc \vdash [\cdot]}$$

图5　类型规则

规则(T-CON)表示常量的安全级别是l,D为空集;规则(T-VAR)表示变量v的安全级别是$\Gamma(v)$,D为空集;规则(T-OP)表示表达式e_1和e_2的op运算值的安全级别是它们各自安全级别的最小上界,D为e_1和e_2的各自的D集的并集:$D_1 \cup D_2$;规则(T-DECL)中的前提条件$D = \varnothing$是用来限定declassify表达式不允许嵌套,把Vars(e)即表达式e中的变量集合作为释放表达式declassify(e, l_2)的D集。

规则(T-SKIP)表示skip命令是良类型,它的执行不修改效果集合U和D。

规则(T-ASSI)的前提条件中,$l \cup pc \subseteq \Gamma(x)$表示赋值语句右侧表达式$e$的安全级别和程序计数器标签$pc$的最小上界必须不高于赋值语句左侧的变量$v$的安全级别(目的是防止非法的隐式和显式流);$\forall\, x \in D.\, \Gamma(x) \subseteq (H, H)$表示对于被释放表达式中的所有变量,其完整性级别必须大于等于H,即取值只可能是H级(只考虑两级安全),秘密性级别必须小于等于H,即取值可以是H或L;$D \neq \varnothing \Rightarrow pc \subseteq (L, H)$表示当赋值语句右侧表达式$e$中存在释放表达式时,$pc$的秘密性级别取值只能是$L$级(目的是防止非法的隐式流),$pc$的完整性级别只能是$H$级,即要求选择(或循环)语句的判断表达式是高完整性(目的是如果释放表达式语句出现在判断表达式为低完整性的分支中,那么攻击者就能影响释放表达式所在的命令是否被执行);$I(v) = H \Rightarrow U = v$表示如果被赋值变量是高完整性级别,那么将该变量加入到U集合中;只有满足上述所有前提条件,才能保证赋值语句$v := e$是良类型。

规则(T-SEQ)表示当顺序语句$c_1; c_2$中的两个命令都是良类型,并且命令c_2中的所有被释放表达式的变量在命令c_1中没有被更新(前提条件$U_1 \cap D_2 = \varnothing$说明这点),那么顺序命令$c_1; c_2$是良类型,顺序语句的执行将引起命令$c_1$和$c_2$相应效果集合的叠加。

规则(T-IF)表示IF选择语句是良类型必须满足以下前提条件:(1)各个分支命令在安全环境Γ和程序计数器标签值$l \cup pc$下是良类型(防止由于选择语句自身的嵌套以及选择语句嵌套循环语句所引发的非法隐式信息流);(2)如果判断表达式中出现释放表达式,那么它的被释放表达式中的变量是高完整性并且pc的秘密性级别取值是L。另外,

由于类型规则需要考虑所有可能的执行路径,因此选择语句的效果集合是各个分支命令相应效果集合的叠加。

规则(T-WHILE)表示 while 循环语句是良类型必须满足以下前提条件:(1)循环体在安全环境和程序计数器标签值 $l\cup pc$ 下是良类型并且循环条件和循环体内所有被释放表达式中的变量没有在循环体内被更新时(循环体 c 的循环执行可看成若干次顺序命令 $c;c$ 形式);(2)同规则(T-IF)的条件(2)。

规则(T-HOLE)限制低完整性代码区[·]的 pc 秘密性级别的取值必须为 L,这是为了防止非法的隐式流流入到攻击者可以控制的低完整性代码区。

3.2 类型规则的可靠性

定理(可靠性定理):如果图5所示的类型规则能推导出程序 c 是良类型,即 $\Gamma,pc\vdash c:U,D$,则程序 c 满足健壮的定位定界策略。

证明:对 $\Gamma,pc\vdash c:U,D$ 导出结构的深度作归纳证明。

首先证明 c 满足定义5的安全条件(2):设 $m_1=_L m_2$,$\langle c,m_1,\varnothing\rangle\Downarrow\langle m'_1,E_1\rangle$,$\langle c,m_2,\varnothing\rangle\Downarrow\langle m'_2,E_2\rangle$,$\langle c,m_1,E_1\rangle R_{m_1,m_2}\langle c,m_2,E_2\rangle$ 要推出 R_{m_1,m_2} 满足关于 m_1 和 m_2 的互模拟条件,对每条命令规则证明如下:

(1)(T-SKIP):$R_{m_1,m_2}=\{(\langle\text{skip},m_1,\varnothing\rangle,\langle\text{skip},m_1,\varnothing\rangle,\langle\text{stop},m_2,\varnothing\rangle)\}$,skip 的执行没有影响存储 m_1 和 m_2,被释放表达式集合仍为空集,R_{m_1,m_2} 满足关于 m_1 和 m_2 的互模拟条件。

(2)(T-ASSI):$R_{m_1,m_2}=\{(\langle v:=e,m_1,\varnothing\rangle,\langle v:=e,m_2,\varnothing\rangle),(\langle\text{stop},m'_1,E\rangle,\langle\text{stop},m'_2,E\rangle)\}$,有 $E_1=E_2=E$,从而定义4互模拟条件(1)成立。如果 v 是高秘密性的变量,则 $m'_1=_L m'_2$,则定义4互模拟条件(2)中的①成立;如果 v 是低秘密性的变量,设 $m_1I(E)m_2$,则有 $m'_1=_L m'_2$,同样定义4互模拟条件(2)中的①成立。另外,互模拟关系条件(2)中的②显然成立。因此,对赋值规则,R_{m_1,m_2} 满足关于 m_1 和 m_2 的互模拟条件。

(3)(T-SEQ):首先把命令 $c_1;c_2$ 拆成两个单独的命令 c_1 和 c_2 来分析:

设 $\langle c_1,m_1,\varnothing\rangle\Downarrow\langle m_{11},E_{11}\rangle$,$\langle c_1,m_2,\varnothing\rangle\Downarrow\langle m_{21},E_{12}\rangle$,由本规则的前提 $\Gamma,pc\vdash c_1:U_1,D_1$ 可合理假设命令 c_1 满足关于 m_1 和 m_2 的互模拟关系:$R^1_{m_1,m_2}$:$\langle c_1,m_1,\varnothing\rangle R^1_{m_1,m_2}\langle c_1,m_2,\varnothing\rangle$,从而可得:$\langle\text{stop},m_{11},E_{11}\rangle R^1_{m_1,m_2}\langle\text{stop},m_{21},E_{12}\rangle$;$m_1I(E_{11})m_2\Leftrightarrow m_1I(E_{12})m_2$;如果 $m_1I(E_{11})m_2$,从而 $m_{11}=Lm_{21}$。

设 $\langle c_2,m_{11},\varnothing\rangle\Downarrow\langle m_{12},E_{21}\rangle$,$\langle c_2,m_{21},\varnothing\rangle\Downarrow\langle m_{22},E_{22}\rangle$,由本规则的前提 $\Gamma,pc\vdash c_2:U_2,D_2$ 可合理假设 c_2 满足关于 m_{11} 和 m_{21} 的互模拟关系 $R^2_{m_{11},m_{12}}\langle c_2,m_{21},\varnothing\rangle$。然后,作为整体来考虑顺序命令 $c_1;c_2$,此时需要考虑 c_1 和 c_2 之间的衔接问题。构造关于命令 $c_1;c_2$ 的配置上的关系 R_{m_1,m_2}:

$$R_{m_1,m_2}=\{(\langle c_1^1;c_2,m_1^1,E_1^1\rangle,\langle c_1^2,c_2,m_1^1,E_1^2\rangle)\mid\langle c_1,m_1,\varnothing\rangle\rightarrow*\langle c_1^1,m_1^1,E_1^1\rangle\wedge\langle c_1,m_2,\varnothing\rangle\rightarrow*\langle c_1^2,m_2^1,E_1^2\rangle\wedge\langle c_1^1,m_1^1,E_1^1\rangle R^1_{m_1,m_2}\langle c_1^2,m_2^1,E_1^2\rangle\}\cup\{(\langle c_2^1,m_{11}^1,E_{11}\cup E_2^1\rangle,\langle c_2^2,m_{21}^1,E_{12}\cup E_2^2\rangle)\mid\langle c_2,m_{11},\varnothing\rangle\rightarrow*\langle c_2^1,m_{11}^1,E_2^1\rangle\wedge\langle c_2,m_{21},\varnothing\rangle\rightarrow*\langle c_2^2,m_{21}^1,E_2^2\rangle\wedge\langle c_2^1,m_{11}^1,E_2^1\rangle R^2_{m_{11},m_{21}}\langle c_2^2,m_{21}^1,E_2^2\rangle\}$$

目标是要证明 R_{m_1,m_2} 是关于 m_1 和 m_2 的互模拟关系,假设两个可终止配置 cfg_1 和 cfg_2,$cfg_1R_{m_1,m_2}cfg_2$,则要证明互模拟的两个条件是满足的。分两种情况对关系 R_{m_1,m_2} 进行讨论:

①令 $cfg_1=\langle c_1^1;c_2,m_1^1,E_1^1\rangle$,$cfg_2=\langle c_1^2;c_2,m_2^1,E_1^2\rangle$,其中 $\langle c_1,m_1,\varnothing\rangle\rightarrow*\langle c_1^1,m_1^1,E_1^1\rangle$,$\langle c_1,m_2,\varnothing\rangle\rightarrow*\langle c_1^2,m_2^1,E_1^2\rangle$,且 $\langle c_1^1,m_1^1,E_1^1\rangle R^1_{m_1,m_2}\langle c_1^2,m_2^1,E_1^2\rangle$,由规则的前提可知互模拟的两个条件满足。

②令 $cfg_1=\langle c_2^1,m_{11}^1,E_{11}\cup E_2^1\rangle$,$cfg_2=\langle c_2^2,m_{21}^1,E_{12}\cup E_2^2\rangle$,其中,$\langle c_2,m_{11},\varnothing\rangle\rightarrow*\langle c_2^1,m_{11}^1,E_2^1\rangle$,$\langle c_2,m_{21},\varnothing\rangle\rightarrow*\langle c_2^2,m_{21}^1,E_2^2\rangle$,$\langle c_2^1,m_{21}^1,E_2^2\rangle$,$\langle c_2^1,m_{11}^1,E_2^1\rangle R^2_{m_{11},m_{21}}\langle c_2^2,m_{21}^1,E_2^2\rangle$,由 $U_1\cap D_2=\varnothing$ 可得:$m_1I(E_2^i)m_2\Leftrightarrow m_{11}I(E_2^i)m_{21}$,$i=1,2$。

为了证明定义4互模拟的条件(1)满足,假设 $m_1I(E_{11}\cup E_2^1)m_2$,则 $m_1I(E_{11}\cup E_2^1)m_2\Leftrightarrow m_1I(E_{11})m_2\wedge m_1I(E_2^1)m_2$,由 $\Gamma,pc\vdash c_1:U_1,D_1$ 得 $m_1I(E_{11})m_2\Leftrightarrow m_1I(E_{12})m_2$,又由 $\Gamma,pc\vdash c_2:U_2,D_2$ 可得 $m_{11}I(E_2^1)I(E_2^1)m_{21}\Leftrightarrow m_{11}I(E_2^2)m_{21}$,再结合上述情况②的结论可得:$m_1I(E_2^1)m_2\Leftrightarrow m_{11}I(E_2^1)m_{21}\Leftrightarrow m_{11}I(E_2^2)m_{21}\Leftrightarrow m_1I(E_2^2)m_2$,从而可得:$m_1I(E_{12})m_2\wedge m_1I(E_2^2)=m_1I(E_2^2)m_2=m_1I(E_{12}\cup E_2^2)m_2$;另外,假设 $m_1I(E_{11}\cup E_2^1)m_2$,由规则的前提以及 $R^2_{m_{11},m_{21}}$ 可得互模拟的条件(2)是满足的。

(4)(T-IF):分为两种情况:

①当 $l\cup pc=\text{high}$

根据前提 $\Gamma\vdash e:l,D$ 和 $\Gamma,l\cup pc\vdash c_1:U_1,D_1$ 以

及$l\cup pc\vdash c_2:U_2,D_2$可知:命令$c_1$和$c_2$中不存在低秘密性级别的变量被更新,而且$D_1=D_2=\varnothing$。

令$R_{m_1,m_2}=\{(\langle c'_i,m'_1,E\rangle,\langle c'_i,m'_2,E\rangle)\mid\langle$if e then $c_1$else $c_2,m_k,\varnothing\rangle\rightarrow*\langle c'_i,m'_k,E\rangle\wedge\langle c'_i,m'_1,E\rangle R^i_{m_1,m_2}\langle c'_i,m'_2,E\rangle\wedge(i,k\in\{1,2\})\}$,其中$E$是条件判断$e$中的被释放表达式的集合;$R^i_{m_1,m_2}$表示分支$c_i$的互模拟关系。定义4中互模拟的条件(1)显然满足;为了证明其中的条件(2)也满足,假设$m_1 I(E)m_2$,由于不允许在命令c_1和c_2中有低秘密性级别变量的更新,因此$m_1=Lm_2$始终是成立的,即定义4互模拟条件(2)中的①满足,又由$R^i_{m_1,m_2}$可得条件(2)的②满足。

②当$l\cup pc=$low

令$R_{m_1,m_2}=\{(\langle c'_i,m'_1,E\cup E'_{i1}\rangle,\langle c'_i,m'_2,E\cup E'_{i2}\rangle)\mid\langle c_i,m_k,0\rangle\rightarrow*\langle c'_i,m'_k,E'_{i_k}\rangle\wedge\langle c'_i,m'_1,E'_{i_1}\rangle R^i_{m_1,m_2}\langle c'_i,m'_2,E'_{i2}\rangle\wedge(i,k\in\{1,2\})\}$,其中$E$和$R^i_{m_1,m2}$与情况①中含义相同;由规则前提得:$m_1 I(E\cup E'_{i_1})m_2\Leftrightarrow m_1 I(E)m_2\wedge m_1 I(E'_{i_1})m_2\Leftrightarrow m_1 I(E)m_2\wedge m_1 I(E'_{i2})m_2\Leftrightarrow m_1 I(E\cup E'_{i_2})m_2$,从而定义4的互模拟的条件(1)满足,为了证明其中的条件(2)满足,假设$m_1 I(E\cup E'_{i_1})m_2$,则$m_1 I(E)m_2$和$m_1 I(E'_{i_1})m_2$,$m_1 I(E)m_2$,说明了在不同存储下命令的两次运行中,条件表达式的值是相同的,从而进入同样的条件分支c_i,然后由$m_1 I(E'_{i_1})m_2$可得$m_1=Lm_2$,即互模拟条件(2)中的①满足,又由$R^i_{m_1,m_2}$可得条件(2)的②满足。

综上,对条件规则,R_{m_1,m_2}满足关于m_1和m_2的互模拟条件。

(5)(T-WHILE):该规则的证明是(T-SEQ)规则和(T-IF)规则的组合情况。

其次,证明命令c满足定义(5)的安全条件(1)。证明思路与上述类似,分别对每一条命令规则进行类型推导。本文为了节省篇幅,仅从总体上分析如下:定义(5)的安全条件(1)包括了两点限制:第一,攻击者的低完整性代码不能影响信息是否被释放,即影响释放表达式declassify(e,l)所在的命令是否被执行;第二,攻击者的低完整性代码不能影响信息释放的内容,即不能修改declassify(e,l)中的e表达式内容。可见,类型规则必须确保这两点成立。在本文的类型规则中,通过规则前提$D\neq\varnothing\Rightarrow pc\sqsubseteq(L,H)$确保了释放表达式declassify$(e,l)$所在命令仅能在高完整环境中被执行,从而保证了攻击者无法影响信息是否被释放;另外,通过规则前提$\forall\ x\in D.\Gamma(x)\sqsubseteq(H,H)$确保了declassify$(e,l)$中的$e$表达式的高完整性,从而保证了攻击者无法影响信息释放的内容。因此,该类型规则能确保上述两点成立,从而满足定义(5)的安全条件(1)。证毕。

3.3　类型规则的约束性

3.2节给出了类型规则的可靠性证明,即满足类型规则的程序一定满足健壮的定位定界策略。但反之未必,即满足健壮的定位定界策略的程序不一定满足该类型规则。这个关系如图6所示。例如,对于程序:

$$[\cdot];t_{HH}:=h1_{HH};h2_{HH}:=h1_{HH};h1_{HH}:=t_{HH};\mathrm{avg}_{LH}:=\mathrm{declassify}((h1_{HH}+h2_{HH})/2)$$

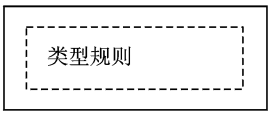

图6　策略与规则的关系

该程序满足健壮的定位定界策略的两个安全条件,但是图5所示的类型规则却判定该程序是不安全的,原因在于类型规则限制被释放表达式中的变量在前面的命令中不允许被修改,而上述例子却对这些变量作了更新。可见本文提出的类型规则虽然满足可靠性定理,但是类型规则的约束性比策略的约束性更强,提出更加宽容的类型规则是未来的工作之一。

4　结束语

秘密信息的可信释放是信息流安全的关键挑战之一。不同维度的信息释放策略存在一定的局限性,而不同维度策略的集成能有效地突破这个局限性。本文提出了一种健壮的定位定界策略,该策略同时考虑了3个维度:内容、地点和调用主体,从而限定攻击者不能释放额外的秘密信息,不能在非法的程序点释放秘密信息以及不能控制秘密信息是否释放。另外,建立了静态实施该策略的类型规则系统,并进行了该规则系统的可靠性分析和证明。存在的问题是静态策略实施机制的限制性太强,虽然能保证可靠性,但是完备性是不能保证的,所以下一步着重研究策略的动态实施机制,从一定程度上对静态实施机制的限制性进行放松,从而获得更宽容的策略实施机制。

参考文献：

[1] 梅宏，曹东刚. 软件可信性：互联网带来的挑战[J]. 中国计算机学会通讯，2010，6(2)：20-23.
Mei Hong，Cao Donggang. Reliability of software：The challenges of the internert [J]. Communication of the CCF，2010，6(2)：20-24.

[2] Sabelfeld A，Myers A C. Language-based information flow security [J]. Selected Areas in Communications，2003，21(1)：5-19.

[3] Goguen J A，Meseguer J. Security policies and security models [C]//IEEE Symposium on Security and Privacy. Oakland，California，USA：IEEE Computer Society，1982：11-20.

[4] Goguen J A，Meseguer J. Unwinding and inference control [C]//IEEE Symposium on Security and Privacy. Oakland，California，USA：IEEE Computer Society，1984：75-87.

[5] Di Pierro A，Hankin C，Wiklicky H. Approximate non-interference [J]. Journal of Computer Security，2004，12(1)：37-81.

[6] Peng L，Steve Z. Downgrading policies and relaxed noninterference [C]//Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Language. New York，USA：ACM，2005：158-170.

[7] Hicks B，King D，McDaniel P，et al. Trusted declassification：high-level policy for a security-typed language[C]//Proceedings of the 2006 Workshop on Programming Languages and Analysis for Security. NY，USA：ACM，2006：65-74.

[8] Sabelfeld A，Sands D. Declassification：dimensions and principles [J]. Journal of Computer Security，2009，17(5)：517-548.

[9] Sabelfeld A，Myers A. A model for delimited information release [J]. Software Security-Theories and Systems，Lecture Notes in Computer Science，2004，3233：174-191.

[10] Askarov A，Sabelfeld A. Localized delimited release：combining the what and where dimensions of information release [C]//Proceedings of the 2007 Workshop on Programming Languages and Analysis for Security. NY，USA：ACM，2007：53-60.

[11] Myers A C，Sabelfeld A. Enforcing robust declassification and qualified robustness[J]. Journal of Computer Security，2006，14(2)：157-196.

[12] Askarov A，Myers A C. A semantic framework for declassification and endorsement [J]. Programming Languages and Systems，Lecture Notes in Computer Science，2010，6012：64-84.