# 基于自动机监控的二维降密策略

金 丽[1] 朱 浩[2,3]
(南通大学江苏省专用集成电路设计重点实验室 南通 226019)[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)[3]

**摘 要** 降密策略静态实施机制具有限制性过强的缺陷:它将降密策略语义条件判定为安全的程序排斥在外。为了建立更加宽容的实施机制,基于自动机理论,建立了二维降密策略的动态监控机制。程序执行中的命令事件被抽象为自动机的输入,自动机根据这些输入信息跟踪程序执行过程中的信息流,禁止违反降密策略的程序命令的执行。最后,证明了自动机监控机制的可靠性。

**关键词** 信息流,自动机,保密性,无干扰
**中图法分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.7.043

## Declassification Policy Based on Automaton Monitoring

JIN Li[1] ZHU Hao[2,3]
(Jiangsu Key Laboratory of ASIC Design, Nantong University, Nantong 226019, China)[1]
(School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]
(School of Computer Science and Technology, Nantong University, Nantong 226019, China)[3]

**Abstract** Static enforcement mechanisms of declassification policies have the flaw of over restrictive, which exclude the programs judged secure by semantic conditions of declassification policies. In order to provide more permissive enforcement mechanisms, we established the dynamic monitoring mechanisms for the two-dimension declassification policy based on the automaton theory. Command events generated during the running of a program are abstracted as the inputs of automaton, and these inputs are used by the automaton to track the information flow during the program running. The command that violates the declassification policy will be forbidden. Additionally, we proved that the mechanisms based on automaton monitoring are sound.

**Keywords** Information flow, Automaton, Confidentiality, Non-interference

传统的加密技术和访问控制不能完全确保端到端的保密性[1],为此,研究者采用信息流控制策略研究端到端的保密性问题。无干扰策略[2]是最重要的信息流控制策略之一,许多信息流控制策略都是在它的基础上衍生而来。无干扰策略规定:系统的低安全等级的输出不能依赖于高安全等级的输入。其中的安全等级可以是保密性等级或完整性等级,本文关注的是保密性。无干扰策略的限制性非常强,使得系统的外部观察者无法从低安全等级的输出中推导出高安全等级的输入信息。但是在现实中,很多系统由于其功能的需要,不可避免地违反了无干扰策略,向用户释放了一定数量的敏感信息,例如用户登录的密码验证程序拒绝登录密码不正确的用户登录,却从反面透露给登录用户一定量的信息(排除一个错误的登录密码)。为此,有必要对无干扰策略的强限制性进行弱化。降密策略[3]是一种弱化的无干扰策略,它允许程序向外界释放必要的敏感信息,但是这种释放必须是受控的,以防止恶意用户通过释放通道非法窃取额外的敏感信息。降密策略具有广泛的应用,比如文献[4]提出使用降密策略确保 Mashup 应用程序的安全。

降密策略的语义条件主要集中在降密的内容、地点、主体以及时间等维度上[3],它们分别关注的是"什么信息可以降密"、"在哪儿降密"、"谁能降密"以及"什么时间能降密"的问题。不同维度的策略解决了降密问题的不同方面,具有一定的局限性。因此,降密策略语义条件的制定需要综合考虑不同的维度,但由于不同维度之间彼此正交,制定无缝集成多个维度的降密策略一直是该研究领域的挑战之一。Banerjee 等人[5]基于"渐进释放"策略提出了一种二维的降密策略,对信息降密的地点和内容进行了限制,但是该策略的语义条件忽略了降密表达式的初始状态,攻击者能利用这点进行"信息清洗"攻击。Askarov 等人[6]提出了"定位定界释放"(Localized Delimited Release)策略,该策略在"定界释放"策略对内容维度限制的基础上,进一步对降密地点做了限定:不允许敏感信息在执行降密语句之前释放,但是允许敏感信息在执行降密

到稿日期:2014-08-06 返修日期:2014-11-27 本文受江苏省博士后科研资助计划(1401022C)资助。
金 丽(1978—),女,讲师,主要研究方向为软件安全,E-mail:jin.li@ntu.edu.cn;朱 浩(1977—),男,博士,副教授,CCF 会员,主要研究方向为软件安全,E-mail:searain@nuaa.edu.cn(通信作者)。

语句之后释放，攻击者能利用这点进行“信息清洗”攻击。为了克服上述二维降密策略的缺点，在前期的工作中，我们已经提出集成内容和地点维度的降密策略[7]，策略的语义条件规定了在程序中合适的地点释放合适内容的敏感信息。

信息流控制策略的实施机制，可分为静态实施和动态实施两种基本方法[8,9]。静态实施方法是在程序的运行之前，通过建立策略的类型规则系统，对程序进行类型推导，推导结果为良类型的程序满足相应的信息流控制策略。动态实施方法是在程序的运行期间，由监控器对程序中的信息流进行合法性检查。静态实施方法关注程序的所有执行路径，但由于缺乏对程序所有执行路径的实时验证能力，往往只能做最保守的近似估计，从而导致对策略约束条件的限制性过强，通常将满足信息流控制策略的程序误判为不安全的；而动态方法采用监控机制，关注的是程序的当前执行路径，它能提供更加宽容的实施机制。但是，静态实施方法发生在程序的运行之前，所以它不会增加系统的运行负载，而动态实施发生在程序的执行过程中，会增加系统的运行负载。

近年来，由于动态应用程序的需要，信息流控制策略的动态实施引起了广泛的关注[8]。文献[10-12]分别在机器代码、字节码以及解释性语言层次实现了信息流的动态监控，但是它们仅仅监控程序中的直接流，没有对程序中的间接流进行监控。文献[13,14]利用动态监控实施了无干扰策略，但是没有考虑信息的降密问题，而在降密策略的实施机制上，研究工作集中在静态实施上[5-7]。

基于上述考虑，在前期工作中提出的结合地点和内容维度的二维降密策略的基础上，基于自动机理论，提出了该策略的动态监控机制，并进行了可靠性证明。本文第 1 节给出了语言模型；第 2 节描述了降密策略；第 3 节建立该降密策略自动机监控机制，并进行了可靠性证明；最后总结全文。

## 1 语言模型

为了描述和推导程序中的信息流，本文采用安全类型的确定性顺序式语言，称为 While 语言。该语言中每个常量和变量都具有一个保密性级别，本文分低保密性 $L$ 和高保密性 $H$ 两级。对攻击者而言，低保密性 $L$ 的数据是完全公开的，而高保密性 $H$ 的数据是不公开的。令常量的保密性级别为 $L$，变量的保密性级别可通过从变量到保密性级别的映射 $\Gamma$ 表示。可将 $\Gamma$ 扩展为求表达式的保密性级别，其值是表达式中各个变量或常量的保密性级别的最小上界。While 语言由表达式 $e$ 和命令 $C$ 组成，用巴科斯范式描述如下：

$e::=n|x|e\oplus e$

$A::=\text{skip}|x:=e|x:=\text{declassify}(e)|\text{output}(e)$

$B::=\text{if } e \text{ then } C \text{ else } C \mid \text{while } e \text{ do } C$

$C::=A|B|C;C$

在表达式 $e$ 中，$n$ 表示常量，$x$ 表示变量，$\oplus$ 是二元运算符。命令 $C$ 分为原子命令 $A$、分支命令 $B$（包括选择语句和循环语句）以及复合命令 $C;C$。在原子命令中，命令 $x:=\text{declassify}(e)$ 是保密信息的降密语句，它的功能是将表达式 $e$ 的高保密性级别 $H$ 降到低保密性级别 $L$，并赋值给变量 $x$，其中 $e$ 称为降密表达式；命令 output($e$) 用来输出低保密性表达式 $e$ 的值。如果 $\Gamma(e)=H$，命令 output($e$) 是非法的。另外，While 语言中还引入了两个辅助命令 end 和 stop，它们仅出现在命令语义中，在退出当前层次的分支命令的作用域时，end 命令被触发执行；stop 命令表示命令的执行终止。

程序语言的语义分为表达式语义和命令语义。在表达式语义中，表达式配置具有形式$\langle m,e\rangle$，其中 $m$ 表示存储状态（它是表达式到其取值的映射），$e$ 是表达式。表达式求值规则具有形式$\langle m,e\rangle \downarrow m(e)$，其中 $m(e)$ 是表达式 $e$ 在存储状态 $m$ 下的取值。在命令的操作语义中，命令配置具有形式$\langle m,c\rangle$，表示命令 $c$ 在存储状态 $m$ 下执行，$\langle m,\text{stop}\rangle$ 表示命令执行完毕时的终止配置。命令配置之间的转移关系可描述为$\langle m,c\rangle \xrightarrow{\alpha}_{\gamma} \langle m',c'\rangle$，表示命令 $c$ 在存储状态 $m$ 下单步执行后到达配置$\langle m',c'\rangle$，其中 $\alpha$ 是 $c$ 触发的命令事件，$\gamma$ 是可观察事件，具有两种形式：$(x,n)$ 和 $obs(m(e))$，其中$(x,n)$表示变量 $x$ 的值将被赋值为 $n$，且赋值后变量 $x$ 的保密性级别为 $L$；$obs(m(e))$表示输出命令 output($e$)输出低保密性表达式 $e$ 的值。命令事件 $\alpha$ 和可观察事件 $\gamma$ 的语法用巴科斯范式描述如下：

$\alpha::=s|a(x,e)|d(x,e)|b(e,c)|f|o(e)$

$\gamma::=(x,n)|obs(m(e))$

在命令事件 $\alpha$ 中，$s$ 由 skip 命令触发；$a(x,e)$ 由赋值命令 $x:=e$ 触发；$d(x,e)$ 由降密命令 $x:=\text{declassify}(e)$ 触发；$b(e,c)$ 由分支命令触发，其中 $e$ 是分支表达式，$c$ 是未执行的分支（对于循环语句，如果 $e$ 为真，则 $c$ 用 skip 命令表示）；$f$ 由分支命令在即将退出其分支范围时触发；$o(e)$ 由命令 output($e$) 触发。所有的命令事件都是内部事件，对攻击者而言是不可观察的，但可以被监控器捕获。可观察事件 $\gamma$ 对攻击者而言是可以观察的，它由命令事件 $d(x,e)$、$a(x,e)$ 以及 $o(e)$ 引发，相应的可观察事件分别称为降密可观察事件、赋值可观察事件以及输出可观察事件。当命令配置转移中没有可观察事件时，$\gamma$ 为空。用$\xrightarrow{\vec{\alpha}}_{\vec{\gamma}}$或$\xrightarrow{*}_{\vec{\gamma}}$表示多步转移（包括零步），其中$\vec{\alpha}$和$\vec{\gamma}$分别表示命令事件序列和可观察事件序列（可能为空）；当不关心转移后的命令配置时，可略写为$\langle m,c\rangle \xrightarrow{*}_{\vec{\gamma}}$。具体的命令操作语义如图 1 所示，其中 $m[x\mapsto m(e)]$ 表示将存储状态 $m$ 中的 $x$ 值更新为 $m(e)$。

信息流可分为直接流和间接流[9]。While 语言中直接流是由赋值命令或降密命令引起的，比如赋值语句 $x:=y$，变量 $y$ 的信息直接流入了变量 $x$。间接流是由分支命令引起的，它又可分为显式的间接流和隐式的间接流，比如选择语句：

$x:=1;\ y:=1;\text{if } e \text{ then } x:=0 \text{ else } y:=0;$

当 $e$ 的值为逻辑真时，then 分支中的赋值语句 $x:=0$ 将会执行，根据 $x$ 的值可以推出 $e$ 的真假值，此时从分支表达式 $e$ 流向 then 分支中的 $x$ 变量的信息流称为显式的间接流，而 else 分支中的赋值语句 $y:=0$ 未执行，但是根据 $y$ 的值也可以推出 $e$ 的真假值，此时从分支表达式 $e$ 流向未执行的 else 分支中的 $y$ 变量的信息流称为隐式的间接流。同理，当表达式 $e$ 的值为逻辑假时，then 分支不会执行，执行 else 分支，此时从分支表达式 $e$ 流向 then 分支的信息流称为隐式的间接流，从分支表达式 $e$ 流向 else 分支的信息流称为显式的间接流。循环结构也存在类似的间接流。在 While 语言中，降密命令是高保密性信息流入低保密性变量的唯一合法通道（除此以外的任何高保密性信息流入低保密性变量的信息流都是非法的）。因此，需要制定和实施降密策略，以防止攻击者利用这个唯一的合法通道获取额外的高保密性信息，或者利用其他非法通道获取高保密性信息。

(S-ASSIGN-HIGH)：

$$\frac{\langle m,e\rangle \downarrow m(e) \quad \Gamma(x)=H}{\langle m,x:=e\rangle \xrightarrow{a(x,e)} \langle m[x\vdash\!\!\rightarrow]m(e)],stop\rangle}$$

(S-ASSIGN-LOW)：

$$\frac{\langle m,e\rangle \downarrow m(e) \quad \Gamma(x)=L}{\langle m,x:=e\rangle \xrightarrow[(x,m(e))]{a(x,e)} \langle m[x\vdash\!\!\rightarrow]m(e)],stop\rangle}$$

(S-SKIP)：

$$\langle m,skip\rangle \xrightarrow{s} \langle m,stop\rangle$$

(S-DECL)：

$$\frac{\langle m,e\rangle \downarrow m(e)}{\langle m,x:=declassify(e)\rangle \xrightarrow[(x,m(e))]{d(x,e)} \langle m[x\vdash\!\!\rightarrow]m(e)],stop\rangle}$$

(S-IF$_1$)：

$$\frac{\langle m,e\rangle \downarrow m(e) \quad m(e)\neq 0}{\langle m,\text{if e then } c_1 \text{ else } c_2\rangle \xrightarrow{b(e,c_2)} \langle m,c_1;end\rangle}$$

(S-IF$_2$)：

$$\frac{\langle m,e\rangle \downarrow m(e) \quad m(e)=0}{\langle m,\text{if e then } c_1 \text{ else } c_2\rangle \xrightarrow{b(e,c_1)} \langle m,c_2;end\rangle}$$

(S-WHILE$_1$)：

$$\frac{\langle m,e\rangle \downarrow m(e) \quad m(e)\neq 0}{\langle m,\text{while e do c}\rangle \xrightarrow{b(e,skip)} \langle m,c;end;\text{while e do } c\rangle}$$

(S-WHILE$_2$)：

$$\frac{\langle m,e\rangle \downarrow m(e) \quad m(e)=0}{\langle m,\text{while e do c}\rangle \xrightarrow{b(e,c)} \langle m,skip;end\rangle}$$

(S-SEQ$_1$)：$\dfrac{\langle m,c_1\rangle \xrightarrow{\alpha}_{\gamma} \langle m',stop\rangle}{\langle m,c_1;c_2\rangle \xrightarrow{\alpha}_{\gamma} \langle m',c_2\rangle}$

(S-SEQ$_2$)：$\dfrac{\langle m,c_1\rangle \xrightarrow{\alpha}_{\gamma} \langle m',c_1'\rangle}{\langle m,c_1;c_2\rangle \xrightarrow{\alpha}_{\gamma} \langle m',c_1';c_2\rangle}$

(S-END)：$\langle m,end\rangle \xrightarrow{f} \langle m,stop\rangle$

(S-OUTPUT)：$\dfrac{\langle m,e\rangle \downarrow m(e) \quad \Gamma(e)=L}{\langle m,output(e)\rangle \xrightarrow[obs(m(e))]{o(e)} \langle m,stop\rangle}$

图 1　命令的操作语义

## 2　降密策略

在前期的工作中[7]已经提出了结合内容和地点维度的降密策略，本节将简要回顾该策略的语义条件，并进行适当修改，以适应监控实施。该策略的语义条件建立在攻击者知识[15]的基础上。直观上，攻击者知识表示攻击者根据程序运行产生的可观察事件序列而推导出的可能产生这些事件的初始存储状态的集合。该集合中的元素越少，攻击者对高保密信息的确定性就越大。令 $m_L$ 表示存储状态 $m$ 仅对低保密性变量进行映射，$m_{1L}=m_{2L}\Leftrightarrow\forall x.\,\Gamma(x)=L\Rightarrow m_1(x)=m_2(x)$。

**定义 1**(攻击者知识)　设程序 $c$ 从初始存储状态 $m_0$ 开始运行，产生可观察事件序列 $\vec{\gamma}$，则攻击者知识为：

$$k(m_{0L},c,\vec{\gamma})=\{m'\mid m_L'=m_{0L}\wedge(\langle m',c\rangle \xrightarrow[\gamma]{*} \langle m'',stop\rangle \vee \langle m',c\rangle \xrightarrow[\gamma]{*} \vec{\gamma}\cdot\gamma')\}$$

其中，$\vec{\gamma}\cdot\gamma'$ 表示 $\vec{\gamma}$ 与 $\gamma'$ 的连接，即 $\gamma'$ 是 $\vec{\gamma}$ 后续的可观察事件。另外，$k(m_{0L},c)$ 表示攻击者初始知识，直观上表示能导致程序运行终止且与初始存储状态 $m_0$ 有相同的低保密性的存储状态的集合，形式上：

$$k(m_{0L},c)=\{m'\mid m_L'=m_{0L}\wedge\exists m'',\vec{\gamma}\cdot\langle m',c\rangle \xrightarrow[\gamma]{*} \langle m'',stop\rangle\}$$

**定义 2**(WHAT&WHERE 策略)　设程序 $c$ 从初始存储状态 $m_0$ 开始运行，产生长度为 $n$ 的可观察事件序列 $\vec{\gamma}_n=\gamma_1\cdots\gamma_n$，$n\geqslant 1$，其中 $\gamma_{r_1}\cdots\gamma_{r_k}$ 是 $\vec{\gamma}_n$ 中的降密可观察事件序列，$1\leqslant r_k\leqslant n$ 且 $1\leqslant k\leqslant n$，该序列中每个降密可观察事件 $\gamma_{r_i}$ 对应的降密表达式为 $e_{r_i}$，其中 $1\leqslant i\leqslant k$，令 $m^{r_i}$ 表示当降密事件 $\gamma_{r_i}$ 发生时的当前存储状态，程序 $c$ 的运行满足“WHAT&WHERE”策略，当且仅当同时满足下列两个安全条件：

(I) $\forall i,1\leqslant i\leqslant n,(\forall j,1\leqslant j\leqslant k,i\neq r_j)\Rightarrow k(m_{0L},c,\vec{\gamma}_{i-1})=k(m_{0L},c,\vec{\gamma}_i)$，其中 $k(m_{0L},c,\vec{\gamma}_0)=k(m_{0L},c)$；

(II) $\forall j,1\leqslant j\leqslant k,m_0(e_{r_j})=m^{r_j}(e_{r_j})$。

条件(I)表示只有降密可观察事件发生时才可能改变攻击者知识，从而对保密信息降密的地点进行了限定；条件(II)表示在降密可观察事件发生时，其对应的降密表达式的值相对于初始存储状态下该表达式的值并没有发生改变，从而对保密信息降密的内容进行了限定。

## 3　自动机监控

为了实施“WHAT&WHERE”策略，本文对 Guernic 等人[14]的无干扰监控工作进行了扩展，建立了一个流敏感的自动机，动态地监控程序执行的每一步，授权符合“WHAT&WHERE”策略的命令的执行，修改或禁止不符合该策略的命令的执行。

### 3.1　自动机的定义

程序 $P$ 中的所有变量和高保密性的输入变量分别用集合 $Vars(P)$ 和 $S(P)$ 表示，其中 $S(P)\subseteq Vars(P)$；令监控自动机为 $A(P)$，其定义为一个五元组 $(Q,\Phi,\Psi,\delta,q_0)$，其中 $Q$ 是状态集合，$\Phi$ 是输入集合，$\Psi$ 是输出集合，$\delta$ 是状态转移函数，$q_0$ 是初始状态。

具体地，自动机的状态集合中的元素是一个二元组 $(V,w)$，其中元素 $V$ 是变量集合，用来跟踪 $P$ 的每个执行步中被 $S(P)$ 中的变量值所影响的变量；元素 $w$ 是由正则表达式 $(L+H)^*$ 表示的语言中的一个字(或称字符串)，它模拟了一个栈，用来跟踪显式间接流。自动机的初始状态 $q_0=(S(P),\varepsilon)$，其中 $\varepsilon$ 表示空串。

自动机输入集合 $\Phi$ 的元素是程序 $P$ 在执行过程中命令事件的抽象。对于原子命令，输入元素 $nop$、$a(x,e)$、$d(x,e)$ 以及 $o(e)$ 分别对应命令 skip、$x:=e$、$x:=$declassify$(e)$ 以及 output$(e)$。对于分支命令，自动机的输入元素依次为 $b(e)$、$f$ 以及 $not\ C$，其中输入元素 $b(e)$ 在分支表达式 $e$ 被求值时首先发送给自动机，当分支命令中某分支执行完毕，即将退出分支命令作用域时，将自动触发 end 命令，与此命令对应的输入元素 $f$ 将被发送给自动机；同时，为了跟踪隐式间接流，分支命令中未执行的分支 $C$ 对应的输入元素 not $C$ 被发送给自动机。比如在条件语句“if $x>8$ then $C_1$ else $C_2$”中“$x>8$”被求值时，输入元素“$b(x>8)$”被发送给自动机，设“$x>8$”的值为真，则执行 then 分支中的命令 $C_1$，当命令 $C_1$ 被执行完毕时，输入元素 $f$ 以及 not $C_2$ 被发送给自动机。

自动机输出集合 $\Psi$ 的元素包括 OK、NO、ACK 以及 output$(\theta)$，其中 OK 用来授权原子命令的执行，NO 用来禁止原子命令的执行，ACK 用来确认接收到非原子命令对应的输入元素，output$(\theta)$ 用来输出缺省的提示信息。

自动机的状态转移函数具有一般形式 $(q,\phi)\xrightarrow{\psi}q'$，它表示

自动机在状态 $q$ 下接收到输入 $\phi$，状态转移为 $q'$，并且输出 $\psi$。转移函数使用了两个句法分析函数：函数 $FV(e)$ 用来获取表达式 $e$ 中自由变量的集合，例如 $FV(x+y)$ 返回集合 $\{x,y\}$；函数 $modified(C)$ 用来获取由命令 $C$ 的执行所引起修改的变量的集合，该函数的定义如图 2 所示。

在转移函数中，$m_0$ 表示初始的存储状态，$m$ 表示当前存储状态。具体的状态转移函数（以下简称为函数）如图 3 所示。

$modified(x:=e)=\{x\}$

$modified(x:=declassify(e))=\{x\}$

$modified(output\ (e))=\emptyset$

$modified(skip)=\emptyset$

$modified(if\ e\ then\ C_1\ else\ C_2)=modified(C_1)\cup modified(C_2)$

$modified(while\ e\ do\ C)=modified(C)$

$modified(C_1;C_2)=modified(C_1)\cup modified(C_2)$

图 2　函数 modified(C)的定义

| | | |
|---|---|---|
| $((V,w),b(e))\xrightarrow{ACK}(V,wL)$ | iff $FV(e)\cap V=\emptyset$ | (T-BRANCH-low) |
| $((V,w),b(e))\xrightarrow{ACK}(V,wH)$ | iff $FV(e)\cap V\neq\emptyset$ | (T-BRANCH-high) |
| $((V,wa),f)\xrightarrow{ACK}(V,w)$ | | (T-END) |
| $((V,w),not\ C)\xrightarrow{ACK}(V\cup modified(C),w)$ | iff $w\notin\{L\}^*$ | (T-NOT-high) |
| $((V,w),not\ C)\xrightarrow{ACK}(V,w)$ | iff $w\{L\}^*$ | (T-NOT-low) |
| $((V,w),nop)\xrightarrow{ACK}(V,w)$ | | (T-SKIP) |
| $((V,w),a(x,e))\xrightarrow{OK}(V\cup\{x\},w)$ | iff $w\notin\{L\}^*$ or $FV(e)\cap V\neq\emptyset$ | (T-ASSIGN-sec) |
| $((V,w),a(x,e))\xrightarrow{OK}(V\backslash\{x\},w)$ | iff $w\in\{L\}^*$ and $FV(e)\cap V=\emptyset$ | (T-ASSIGN-pub) |
| $((V,w),d(x,e))\xrightarrow{OK}(V\backslash\{x\},w)$ | iff $w\in\{L\}^*$ and $m(e)=m_0\neq(e)$ | (T-DECLASSIFY-OK) |
| $((V,w),d(x,e))\xrightarrow{NO}(V,w)$ | iff $w\notin\{L\}^*$ or $m(e)m_0(e)$ | (T-DECLASSIFY-NO) |
| $((V,w),o(e))\xrightarrow{OK}(V,w)$ | iff $w\notin\{L\}^*$ and $FV(e)\cap V=\emptyset$ | (T-OUTPUT-OK) |
| $((V,w),o(e))\xrightarrow{output(\theta_2)}(V,w)$ | iff $w\in\{L\}^*$ and $FV(e)\cap V\neq\emptyset$ | (T-OUTPUT-DEFAULT) |
| $((V,w),o(e))\xrightarrow{NO}(V,w)$ | iff $w\notin\{L\}^*$ | (T-OUTPUT-NO) |

图 3　监控自动机的状态转移函数

当自动机在状态$(V,w)$下接收到输入 $b(e)$时，它将通过计算交集 $FV(e)\cap V$ 的值判断分支表达式 $e$ 是否受 $S(P)$中的元素影响。函数(T-BRANCH-low)表示当 $FV(e)\cap V$ 为空集(∅)时，分支表达式 $e$ 的值未受 $S(P)$中的元素影响，则在字 $w$ 后面连接字母 $L$，表示当前层次的分支表达式 $e$ 为低保密性。函数(T-BRANCH-high)表示当交集 $FV(e)\cap V$ 非空时，分支表达式 $e$ 的值受集合 $S(P)$中的元素影响，则在字 $w$ 的后面连接字母 $H$，表示当前层次的分支表达式 $e$ 为高保密性。

函数(T-END)表示当自动机在状态$(V,wa)$下接收到输入 $f$，它将字 $wa$ 的最后面一个字母删除，从而恢复到进入该分支命令前的字值 $w$。

当自动机在状态$(V,w)$下接收到输入 not $C$，它通过计算字 $w$ 是否包含 $H$ 字母来判断 $C$ 的执行是否受 $S(P)$中元素的影响。函数(T-NOT-high)表示当 $w$ 的值包含 $H$ 字母时，命令 $C$ 必定存在于某个分支表达式为高保密性的分支命令的作用域内，为了防止高保密性的分支表达式的信息隐式地流入命令 $C$ 中的低保密性变量中(即非法的隐式间接流)，函数(T-NOT-high)将命令 $C$ 中赋值语句左边被赋值的变量加入到集合 $V$ 中(即 $V\cup modified(C)$)，相当于提升了这些变量的保密性级别，从而防止了非法的隐式间接流。函数(T-NOT-low)表示当 $w$ 的值不包含 $H$ 字母时，不可能存在流向命令 $C$ 的非法的隐式间接流，因此该函数不作任何处理，自动机的状态保持不变。

函数(T-SKIP)描述了自动机在接收到输入 $nop$ 时将授权 skip 命令的执行，自动机的状态保持不变。

函数(T-ASSIGN-sec)表示当赋值命令 $x:=e$ 右边的表达式 $e$ 包含高保密性变量(即 $FV(e)\cap V\neq\emptyset$)或者该赋值命令出现在分支表达式为高保密性的分支命令中(即 $w\notin\{L\}^*$)时，自动机将授权该赋值命令的执行，但需要将赋值命令左边的被赋值变量 $x$ 加入到集合 $V$ 中，即 $V\cup\{x\}$，相当于提升了 $x$ 的保密性级别，从而防止了非法的直接流和间接流。函数(T-ASSIGN-pub)表示当赋值命令 $x:=e$ 右边的表达式 $e$ 不包含高保密性变量(即 $FV(e)\cap V=\emptyset$)并且该赋值命令未出现在分支表达式为高保密性的分支命令中(即 $w\in\{L\}^*$)时，自动机授权该赋值命令的执行，但需要将赋值语句左边的被赋值变量 $x$ 从集合 $V$ 中删除(即 $V\backslash\{x\}$)。

函数(T-DECLASSIFY-OK)表示当 $w$ 的值不包含 $H$ 字母，而且降密表达式 $e$ 在初始存储状态 $m_0$ 与当前存储状态 $m$ 下的值相等时，降密语句 $x:=declassify(e)$ 才被授权执行；另外，由于流敏感的原因，自动机还需要将变量 $x$ 从集合 $V$ 中删除(即 $V\backslash\{x\}$)。函数(T-DECLASSIFY-NO)表示当 $w$ 的值包含 $H$ 字母，或降密表达式 $e$ 在初始存储状态与当前存储状态下的值不相等时，自动机将禁止降密语句的执行，自动机的状态保持不变。

函数(T-OUTPUT-OK)表示自动机授权 output($e$)的执行需要满足两个条件：第一，命令 output($e$)不能出现在分支表达式为高保密性的分支命令中，即需要满足 $w\in\{L\}^*$；第二，输出的表达式 $e$ 中不能包含高保密性变量，即需要满足 $FV(e)\cap V=\emptyset$。函数(T-OUTPUT-DEFAULT)表示如果仅满足上述第一个条件而不满足第二个条件，那么自动机将把命令 output($e$)替换为 output($\theta_2$)，输出缺省值 $\theta_2$，提示用户非法输出。函数(T-OUTPUT-NO)表示当上述第一个条件不满足时，自动机将禁止命令 output($e$)的执行。

### 3.2　自动机监控实例

根据 3.1 节介绍的自动机及其状态转移函数，本节将以两个实例具体分析自动机的监控过程。

实例 1(平均工资攻击)　假设某公司中有 $n$ 个员工，其

工资分别用高保密性变量 $h_i$ 表示，$1\leqslant i\leqslant n$。假设需要把 $n$ 个员工的平均工资降到低保密性变量 $avg$ 中，不能泄漏除平均工资以外关于工资的信息。假设 $n=4$，求平均工资的语句 $c_1$：

$avg:=\text{declassify}((h_1+h_2+h_3+h_4)/4)$

一种针对求平均工资的攻击 $c_1$-*attack*：

$h_2:=h_1;h_3:=h_1;h_4:=h_1;avg:=\text{declassify}((h_1+h_2+h_3+h_4)/4)$

显然 $h_1$ 完全泄漏给了 $avg$，用类似的途径可获取其他工资信息。表 1 描述了该程序的自动机监控执行过程。表 1 的第 1 列描述了 $c_1$-*attack* 的源程序，第 2 列描述了源程序在没有自动机监控下将要执行的原子命令序列，第 3 列和第 4 列分别描述了自动机的输入和输出，第 5 列描述了自动机根据 3.1 节图 3 所示的函数发生转移后的新状态，最后一列描述了自动机监控下所执行的命令序列。假设初始存储 $m_0$ 满足 $m_0(h_1):=2,m_0(h_2):=3$，对于 $\forall i\in\{3,4\}$ 有 $m_0(h_i):=0$。程序高保密性输入变量的集合 $S(P)$ 为 $\{h_1,h_2,h_3,h_4\}$。在表的第 4 行，根据函数(T-ASSIGN-sec)，变量 $e$ 被加入到集合 $V$ 中。在表的第 5 行，$m_0(e)\neq m(e)$，根据图 3 中的函数(T-DECLASSIFY-NO)，自动机监控将禁止该命令的执行，防止高保密信息 $h_1$ 的完全泄漏。

表 1　平均工资攻击的自动机监控执行

| Program $c_1$-attack | Proposed action: $m_0(h_1):=2,m_0(h_2):=3$ $\forall i\in\{3,4\},m_0(h_i):=0$ | Automaton input $\Phi$ | Automaton output $\Phi$ | Automaton state (V,w) | Executed |
|---|---|---|---|---|---|
| $h_2:=h_1;$ | $h_2:=h_1;$ | $a(h_2,h_1)$ | OK | $(\{h_1,h_2,h_3,h_4\},\varepsilon)$ | $h_2:=h_1;$ |
| $h_3:=h_1;$ | $h_3:=h_1;$ | $a(h_3,h_1)$ | OK | $(\{h_1,h_2,h_3,h_4\},\varepsilon)$ | $h_3:=h_1;$ |
| $h_4:=h_1;$ | $h_4:=h1;$ | $a(h_4,h_1)$ | OK | $(\{h_1,h_2,h_3,h_4\},\varepsilon)$ | $h_4:=h_1;$ |
| $e:=(h_1+h_2+h_3+h_4)/4;$ | $e:=(h_1+h_2+h_3+h_4)/4;$ | $a(e,(h_1+h_2+h_3+h_4)/4)$ | OK | $(\{h_1,h_2,h_3,h_4,e\},\varepsilon)$ | $e:=(h_1+h_2+h_3+h_4)/4;$ |
| avg:= declassify(e); | avg:= declassify(e); | d(avg,e) | NO | $(\{h_1,h_2,h_3,h_4,e\},\varepsilon)$ | (禁止执行) |

实例 2(电子钱包攻击)　假设变量 $h$ 表示客户电子钱包中金钱的数量，变量 $l$ 表示当前购物会话中需要付款的总数，变量 $k$ 表示购买某项商品需要的花费数量，其中 $\Gamma(h)=H$，$\Gamma(l)=\Gamma(k)=L$。下面的程序 $c_2$ 是实现电子购物的关键代码，它首先判断电子钱包中钱的数量是否足够购买某项商品，如果钱的数量充足，则从客户电子钱包中转移一定数量的钱到变量 $l$ 中：

$e:=\text{declassify}(h\geqslant k);$

if $e$ then $\{h:=h-k;l:=l+k;\}$ else skip;

其中 $\Gamma(e)=L$。经过分析不难发现，上述程序满足定义 2 的两个安全条件。设 $h$ 是 $n$ 比特的整数，考虑该程序的变体 $c_2$-*attack*：

$l:=0;$

while $(n\geqslant 0)$ do {

$k:=2^{n-1};$

$e:=\text{declassify}(h\geqslant k);$

if $e$ then $\{h:=h-k;l:=l+k;\}$ else skip;

$n:=n-1;\}$

其中 $\Gamma(e)=\Gamma(k)=\Gamma(n)=L$。该程序是对电子钱包的一种信息清洗攻击，利用降密语句将高保密性变量 $h$ 的信息逐位泄漏到变量 $l$ 中。分析可知定义 2 判定上述程序是不安全的。

表 2　电子钱包攻击的自动机监控执行

| Program $c_2$-attack | Proposed action: $m_0(n):=3;\ m_0(h):=6$ | Automaton input $\Phi$ | Automaton output $\Phi$ | Automaton state (V,w) | Executed |
|---|---|---|---|---|---|
| l:=0; | l:=0; | a(l,0) | OK | ({h},ε) | l:=0; |
| while (n>0) do { | | b(n>0) | ACK | ({h},L) | |
| k:=$2^{n-1}$; | k:=$2^{n-1}$; | a(k,$2^{n-1}$) | OK | ({h},L) | k:=$2^{n-1}$; |
| e:= declassify(h≥k); | e:= declassify(h≥k); | d(e,h≥k) | OK | ({h},L) | e:= declassify(h≥k); |
| if e then | | b(e) | ACK | ({h},LL) | |
| h:=h−k; | h:=h−k; | a(h,h−k) | OK | ({h},LL) | h:=h−k; |
| l:=l+k; | l:=l+k; | a(l,l+k) | OK | ({h},LL) | l:=l+k; |
| else | | | | | |
| skip; | | not skip | ACK | ({h},LL) | |
| | | f | ACK | ({h},L) | |
| n:=n−1;} | n:=n−1; | a(n,n−1) | OK | ({h},L) | n:=n−1; |
| | | not skip | ACK | ({h},L) | |
| | | f | ACK | ({h},ε) | |
| | (第 2 轮循环) | b(n>0) | ACK | ({h},L) | |
| | k:=$2^{n-1}$; | a(k,$2^{n-1}$) | OK | ({h},L) | k:=$2^{n-1}$; |
| | e:= declassify(h≥k); | d(e,h≥k) | OK | ({h},L) | e:= declassify(h≥k); |
| | h:=h−k; | b(e) | ACK | ({h},LL) | |
| | l:=l+k; | a(h,h−k) | OK | ({h},LL) | h:=h−k; |
| | | a(l,l+k) | OK | ({h},LL) | l:=l+k; |
| | | not skip | ACK | ({h},LL) | |
| | n:=n−1; | f | ACK | ({h},L) | |
| | | a(n,n−1) | OK | ({h},L) | n:=n−1; |
| | | not skip | ACK | ({h},L) | |
| | | f | ACK | ({h},ε) | |
| | (第 3 轮循环) | b(n>0) | ACK | ({h},L) | |
| | k:=$2^{n-1}$; | a(k,$2^{n-1}$) | OK | ({h},L) | k:=2n−1; |
| | e:= declassify(h≥k); | d(e,h≥k) | NO | ({h},L) | (禁止执行) |

表 2 描述了电子钱包攻击的自动机监控执行过程。假设程序执行的初始存储状态 $m_0$ 满足 $m_0(n)=3,m_0(h)=6$。当

程序中的循环语句循环到第 3 轮时，降密表达式 $h \geqslant k$ 的值与初始存储状态下该表达式的值不再相同，根据图 3 中的函数(T-DECLASSIFY-NO)，自动机将禁止该降密语句的执行。

### 3.3 监控的可靠性

**定理 1**(可靠性定理) 程序 C 的基于上述自动机的监控执行满足定义 2 的降密策略。

证明：对自动机输入序列 $\vec{s}_n$ 的长度 $n$ 进行归纳证明。

(1)基础：当 $n=0$ 时，定义 2 的两个安全条件恒成立。

(2)归纳：假设结论对自动机的输入序列成立，需要证明对于输入元素 $\vec{s}_n$，结论仍成立。下面对输入元素 $s_n$ 分情况进行讨论。

1)当 $s_n=d(x,e)$时，分两种子情况：

①当 $w \notin \{L\}^*$ 或 $m(e) \neq m^0(e)$时，根据函数(T-DECLASSIFY-NO)，降密命令将被禁止执行，攻击者不能获取任何高保密性信息，而且自动机状态和存储状态都没有发生改变，根据归纳假设，定义 2 的两个安全条件仍然成立。

②当 $w \in \{L\}^*$ 且 $m(e)=m^0(e)$时，定义 2 的安全条件(II)显然成立。另外，安全条件(I)的蕴涵式的左边为逻辑假，因此不要求攻击者的知识不发生变化，从而安全条件(I)也成立。

2)当 $s_n=a(x,e)$时，分两种子情况：

①当 $w \notin \{L\}^*$ 或 $FV(e) \cap V \neq \emptyset$ 时，根据函数(T-ASSIGN-sec)，赋值命令左边的被赋值变量 $x$ 加入到集合 $V$ 中，相当于提升了 $x$ 的保密性级别，攻击者不能观察到任何高保密性信息，攻击者的知识没有发生变化，从而定义 2 的安全条件(I)成立。另外，由于本情况下不存在降密表达式，因此定义 2 的安全条件(II)恒成立。

②当 $w \in \{L\}^*$ 且 $FV(e) \cap V = \emptyset$ 时，显然不存在从表达式 $e$ 到变量 $x$ 的非法信息流，攻击者不能观察到任何高保密性信息，攻击者的知识没有发生变化，从而定义 2 的安全条件(I)成立。同理，由于本情况下不存在降密表达式，因此定义 2 的安全条件(II)恒成立。

3)当 $s_n' \in \{b(e), f, \text{not } C, nop, o(e)\}$时，根据图 3 相应的函数，不存在非法信息流，因此定义 2 的安全条件(I)满足。又由于这些情况下不存在降密表达式，因此定义 2 的安全条件(II)恒成立。

**结束语** 本文提出了一个二维的“WHAT&WHERE”降密策略的自动机监控机制。该二维降密策略分别从内容和地点维度对信息降密进行了限制，内容维度限定降密的内容不能超过程序功能允许降密的信息，地点维度限定降密仅仅能发生在特定的降密语句中。为了动态实施该策略，建立了该策略的自动机模型。自动机首先捕获程序执行的命令事件，对其进行分析，授权符合”WHAT&WHERE”降密策略的命令执行，修改或禁止违反该降密策略的命令的执行。最后，证明了自动机监控执行的程序一定满足“WHAT&WHERE”降密策略。下一步的研究工作将集中在更复杂的语言模型中研究多维的降密策略及其实施机制。

## 参 考 文 献


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

[2] Goguen J A, Meseguer J. Security policies and security models [C]// IEEE Symposium on Security and Privacy. 1982: 11-20

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

[4] Magazinius J, Askarov A, Sabelfeld A. A lattice-based approach to mashup security[C]// 5th ACM Symposium on Information, Computer and Communications Security. 2010: 15-23

[5] Banerjee A, Naumann D A, Rosenberg S. Expressive declassification policies and modular static enforcement [C]// IEEE Symposium on ecurity and Privacy. Oakland, CA: IEEE Computer Society Press, 2008: 339-353

[6] Askarov A, Sabelfeld A. Localized delimited release: combining the what and where dimensions of information release [C]// 2007 Workshop on Programming Languages and Analysis for Security. San Diego, California: ACM Computer Society Press, 2007: 53-60

[7] 朱浩，庄毅，薛羽，等. 基于内容和地点维度的机密信息降级策略[J]. 计算机科学，2012，39(8)：153-157，185
Zhu H, Zhuang Y, Xue Y, et al. Declassification Policy Based on Content and Location Dimensions[J]. Computer Science, 2012, 39(8): 153-157, 185

[8] Russo A, Sabelfeld A. Dynamic vs. static flow-sensitive security analysis[C]// 23rd IEEE Computer Security Foundations Symposium. 2010: 186-199

[9] David B, Vincent J, Felix k, et al. Enforceable Security Policies Revisited[J]. ACM Transactions on Information and System Security, 2013, 16(1): 3-26

[10] Zhu Y, Jung J, Song D, et al. *Privacy Scope: A precise information flow tracking system for finding application leaks*: EECS-2009-145[R]// Berkeley: Electrical Engineering and Computer Sciences, University of California. 2009

[11] Nair S K, Simpson P N D, Crispo B, et al. A virtual machine based information flow control system for policy enforcement [J]. Electronic Notes in Theoretical Computer Science, 2008, 197(1): 3-16

[12] Dhawan M, Ganapathy V. Analyzing information flow in Java Script-based browser extensions [C]// Computer Security Applications Conference. Honolulu, HI: IEEE Computer Society Press, 2009: 382-391

[13] Magazinius J, Russo A, Sabelfeld A. On-the-fly inlining of dynamic security monitors [J]. Computers & Security, 2012, 31 (7): 827-843

[14] Le G G, Banerjee A, Schmidt D A. Automata-based confidentiality monitoring[M]// Okada M, Satoh I, eds. Advances in Computer Science-ASIAN 2006. Secure Software and Related Issues: 11th Asian Computing Science Conference. Springer Berlin Heidelberg, 2007: 75-89

[15] Askarov A, Myers A. A semantic framework for declassification and endorsement[C]// Gordon A D, ed. Proceedings of the 19th European Symposium on Programming Languages and Systems. *Springer Berlin Heidelberg*, 2010: 64-84