二维降密策略的内联引用监控方法

朱 浩[1,2] 陈建平[1] 金 丽[3]
(南通大学计算机科学与技术学院 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]
(南通大学电子信息学院 南通 226019)[3]

摘 要 降密策略的静态实施机制存在限制性过强的缺陷,基于虚拟机的动态监控机制不能完全适合 Web 和即时编译环境。为此,基于内联引用监控方法,实施了基于内容和地点维度的二维降密策略。提出了内联引用监控方法的程序变形规则,并证明了该方法的可靠性;根据该程序变形规则,将源程序进行变形重写,生成一个新的程序,它能脱离外部监控环境,实现自我监控。

关键词 降密策略,监控,内联引用,无干扰
中图法分类号 TP311 **文献标识码** A

In-lined Reference Monitor Method of Two-dimension Information Release Policy

ZHU Hao[1,2] CHEN Jian-ping[1] JIN Li[3]
(School of Computer Science and Technology, Nantong University, Nantong 226019, China)[1]
(School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]
(School of Electronics and Information, Nantong University, Nantong 226019, China)[3]

Abstract The static enforcement methods of declassification policies are over-restrictive. Dynamic approaches based on virtual machines are not suited to Web and just-in-time compiling environments completely. To this end, a two-dimension declassification policy based on the dimension of WHAT and WHERE was enforced by in-lined reference monitor method. The transformation rules of in-lined reference monitor method were presented, and the soundness of the rules was proved. According to transformation rules of the program, the source program is transformed and rewritten to a new program, which is independent of external monitoring environments and can be self-monitored.

Keywords Information release policy, Monitor, In-lined reference, Non-interference

1 引言

在安全性要求较高的场合,系统的输入和输出可分为不同的安全等级:高安全等级和低安全等级。高安全等级的信息对普通用户是保密的,而低安全等级的信息是公开的。为了防止系统对高安全等级的信息造成泄露,无干扰安全策略[1,2]规定:系统高安全等级输入的变化不能影响系统低安全等级的输出。该策略让用户无法从系统低安全等级的输出中推导出高安全等级的输入信息,从而确保高安全等级的零泄露,但是无干扰策略的限制性过强,很多实用系统由于功能的需求不可避免地释放一部分高安全等级的信息,比如信息购买协议和口令登录系统。为了确保高安全等级信息的安全释放,就有必要建立信息的释放策略,也称为降密策略[2]。

降密策略的研究可分为安全条件和实施机制两个方面。在安全条件方面,主要在降密的时间、地点、主体、内容等维度进行限制,确保在合适的时间和程序点由合适的主体释放合适内容的高安全等级信息[3]。在实施机制上,目前的研究主要集中在静态实施和动态实施两个方面[4]。在静态实施方面,主要通过指定编译规则,在程序编译阶段检查降密的安全条件是否满足,但是由于计算复杂度的限制,编译规则不可能考虑程序的所有执行路径,只能存在限制性过强的保守的规则,从而排除了一些事实上安全的程序[5]。在动态实施方面,在程序运行过程中,对每一条指令的执行进行监控,防止违反降密安全条件的行为发生[6]。目前降密策略动态监控方法的研究主要集中在基于虚拟机的监控机制上,比如类似于 JAVA 虚拟机。但是基于虚拟机的动态监控机制不能完全适合 Web 和即时编译环境(Just In Time, JIT)[7,8]。互联网用户使用不同浏览器,其中内置不同的脚本引擎(虚拟机),如 SquirrelFish 和 SpiderMonkey 等,在一种引擎上实现的监控功能不适合另一种。在 JIT 环境中,JIT 产生的本地代码的运行将脱离虚拟机的监控。为此,可以考虑引入内联引用监控方法[9]。该方法通过一个可信的重写进程将实施降密策略安全条件的代码嵌入到目标应用程序中,将原程序变形生成一个新的安全的应用程序,该程序的执行能脱离虚拟机监控环境,实现自我监控。文献[8]利用内联引用监控方法实施了无干扰策略的监控,但是没有考虑降密策略的情形。本文在前人

本文受江苏省博士后科研资助计划(1401022C),南通大学博士科研启动基金(14B22)资助。
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;陈建平(1960—),男,教授,主要研究方向为数值计算与隐私保护;金 丽(1978—),女,讲师,主要研究方向为软件安全。

工作的基础上，结合我们前期工作中提出的二维降密策略[10,11]，提出利用内联引用监控方法实施基于内容和地点维度的降密策略，并从理论上证明了内联引用监控方法在实施二维降密策略时的可靠性。

本文第 2 节描述了程序设计语言模型；第 3 节回顾了二维降密策略的语义条件；第 4 节描述该二维降密策略的内联引用监控方法，并进行了可靠性证明；最后总结全文。

2 程序语言模型

为了在程序设计语言层面推导变量级别的信息流动，本节采用支持安全类型的程序设计语言原型 MWhile，其语法如图 1 所示。

$$e ::= n \mid v \mid e_1 \oplus e_2$$

$$A ::= \text{skip} \mid v := e \mid v := \text{declassify}(e)$$

$$B ::= \text{if } e \text{ then } C_1 \text{ else } C_2 \mid \text{while } e \text{ do } C$$

$$C ::= A \mid B \mid C_1 ; C_2$$

图 1 MWhile 语言语法

在该原型语言中，每个常量和变量都标有相应的安全等级，本文将其分为低安全等级 Low 和高安全等级 $High$。为了获取变量的安全等级，引入映射 Γ，将变量映射到其安全等级。declassify(e)是降密语句，它将表达式 e 的安全等级从 $High$ 降低到 Low，降密语句中 e 称为降密表达式。令 $L=\{Low, High\}$，$Low \sqsubseteq High$，$(L, \sqsubseteq)$构成格，令 $\sqcup$ 为 L 上的最小上界运算。一个表达式的安全等级是该表达式内各个变量和常量的安全等级的最小上界。

3 降密策略

本节简要回顾我们前期工作中提出的基于内容和地点维度的二维降密策略[10,11]。

假设程序 C 从初始内存状态 m_0 执行，在执行过程中产生公开的事件序列 $\vec{\gamma}_n = \gamma_1 \cdots \gamma_n (n \geqslant 1)$，其中 $\gamma_{r_1} \cdots \gamma_{r_k}$ 是 $\vec{\gamma}_n$ 中的降密事件序列，$1 \leqslant r_k \leqslant n$ 且 $1 \leqslant k \leqslant n$，其中每个降密事件 γ_{r_i} 对应的降密表达式为 e_{r_i}，令 m^{r_i} 表示 γ_{r_i} 发生时的内存状态，程序 C 满足“WHAT & WHERE”二维降密策略当且仅当同时满足下列两个条件：

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

4 内联引用监控

程序中需要监控的信息流分为直接流和间接流[4]。直接流由赋值语句触发；间接流由分支语句触发，它可分为显式间接流和隐式间接流。

文献[8]利用内联引用监控方法对无干扰策略实施了监控，但是没有考虑信息降密策略，本文对其工作进行扩展，利用内联引用监控方法实施第 3 节提出的“WHAT&WHERE”策略。

4.1 监控规则

为了实现对程序的内联引用监控，引入了一些辅助函数和变量。引入函数 S 将程序中的常量、变量以及表达式映射到它的安全等级，令 $dom(s)$和 $rng(s)$分别表示 S 的定义域和值域，函数 S 将常量 n 映射为 Low，变量 x 映射为它的影子变量 $S(x)$，其中存储的是当前变量的安全等级。对于表达式，将其安全等级映射为它的各个子表达式的安全等级的最小上界，形式化描述如下所示：

$$S \vdash n \blacktriangleright Low \quad S \vdash x \blacktriangleright S(x)$$

$$\frac{S \vdash e_1 \blacktriangleright \widetilde{e}_1 \quad S \vdash e_2 \blacktriangleright \widetilde{e}_2}{S \vdash e_1 \oplus e_2 \blacktriangleright \widetilde{e}_1 \sqcup \widetilde{e}_2}$$

另外，引入栈结构 G，存储影响当前程序点的所有分支表达式的安全等级最小上界；引入栈结构 I，存储未执行分支中被赋值变量的集合；栈 G 和 I 的栈顶指针是同步的；$head(G)$表示 G 的栈顶元素，$tail(G)$表示除 G 栈顶元素外剩下的部分。栈 G 和 I 以及函数 S 组成了程序的变形环境。程序语句变形规则的一般形式为 $G, I, S \vdash C \blacktriangleright \widetilde{C}$，表示在变形环境 C，I, S 下，程序语句 C 变形为$\widetilde{C}$。

图 2 描述了程序语句的内联引用变形规则。

$$(\text{TR-SKIP}): G, I, S \vdash \text{skip} \blacktriangleright \qquad (\text{TR-ASSIGN}): \frac{S \vdash e \blacktriangleright \widetilde{e}}{G, I, S \vdash x := e \blacktriangleright S(x) := \widetilde{e} \sqcup \text{head}(G); x := e}$$

$$(\text{TR-DECLASSIFY}): G, I, S \vdash x := \text{declassify}(e) \blacktriangleright \text{if head}(G) = \text{Low} \wedge m_0(e) = m(e) \text{ then } S(x) := \text{Low}; x := \text{declassify}(e) \text{ else skip}$$

$$(\text{TR-SEQ}): \frac{\forall i \in 1 \cdots 2, G, I, S \vdash C_i \blacktriangleright \widetilde{C}_i}{G, I, S \vdash C_1 ; C_2 \blacktriangleright \widetilde{C}_1 ; \widetilde{C}_2}$$

$$(\text{TR-IF}): \frac{S \vdash e \blacktriangleright \widetilde{e} \quad x \notin \text{dom}(S) \cup \text{rng}(S) \wedge x \notin G \quad \forall i \in 1 \cdots 2.(x :: G), (\text{modif}(C_{i \bmod 2+1}) :: I) S \vdash C_i \blacktriangleright \widetilde{C}_i ; \widetilde{C}_i' = \widetilde{C}_i ; \kappa(\text{modif}(C_{i \bmod 2+1}), S, x)}{G, I, S \vdash \text{if } e \text{ then } C_1 \text{ else } C_2 \blacktriangleright x := \widetilde{e} \sqcup \text{head}(G); \text{if } e \text{ then } \widetilde{C}_1' \text{ else } \widetilde{C}_2'}$$

$$(\text{TR-WHILE}): \frac{S \vdash e \blacktriangleright \widetilde{e} \quad x \notin \text{dom}(S) \cup \text{rng}(S) \wedge x \notin G \quad (x :: G), S, (\emptyset :: I) \vdash \widetilde{C}}{G, I, S \vdash \text{while } e \text{ do } C \blacktriangleright x := \widetilde{e} \sqcup \text{head}(G); \text{while } e \text{ do } (\widetilde{C}; x := \widetilde{e} \sqcup x); \kappa(\text{modif}(C), S, x)}$$

图 2 程序语句的变形规则

规则(TR-SKIP)表示语句 skip 始终是安全的；规则(TR-ASSIGN)引入赋值语句 $S(x) := \widetilde{e} \sqcup head(G)$，更新变量 x 的当前安全等级，防止非法的直接流和非法的显式间接流；规则(TR-DECLASSIFY)将降密语句变形为一个选择分支语句，降密语句的执行须满足以下条件：G 栈顶元素的值等于 Low 且降密表达式的当前值等于它的初值。如果条件不成立，该规则将忽略该降密语句的执行，防止信息的非法释放；规则(TR-SEQ)表示语句序列的变形结果为语句 C_1 和 C_2 各自变形结果的序列组合。在规则(TR-IF)中，为了防止非法的隐式间接流，引入辅助函数 κ 来更新未执行的分支中被赋值的变量的安全等级 $\kappa(head(I), S, head(G))$，函数 κ 的递归定义如下：

$$\kappa(xs,S,y)=\begin{cases}S(x):=y\sqcup S(x);\kappa(xs\backslash x,S,y), & \text{if } x\in xs\\ skip, & \text{if } xs=\emptyset\end{cases}$$

其中，“$xs\backslash x$”表示从集合 xs 中删除元素 x；规则(TR-IF)还引入变量 x，用来保存影响当前程序点的所有分支表达式安全等级的最小上界，通过函数 κ 更新 $modif$ 函数的值中变量的安全等级，$modif$ 函数的定义如图 3 所示，其中 mod 表示取余运算。规则(TR-WHILE)引入的一些函数和变量与规则(TR-IF)类似。

modif (x:=e)={x}

modif(x:=declassify(e))={x}

modif(skip)=∅

modif(if e then C_1 else C_2=modif(C_1)∪modif(C_2)

modif(while e do C)= modif(C)

modif(C_1;C_2)=modif(C_1)∪modif(C_2)

图 3　modif 函数

4.2　监控实例

针对求平均工资的信息清洗攻击[10]进行内嵌式变形，变形前的源程序如下：

$h_2:=h_1;h_3:=h_1;h_4:=h_1;$

$sum:=h_1+h_2+h_3+h_4;$

$avg:=declassify(\mathrm{sum}/4);$

根据图 2 的变形规则，对上述源程序进行变形：

$S(h_2):=\widetilde{h_1}\sqcup head(G);h_2:=h_1;$

$S(h_3):=\widetilde{h_1}\sqcup head(G);h_3:=h_1;$

$S(h_4):=\widetilde{h_1}\sqcup head(G);h_4:=h_1;$

$sum:=h_1+h_2+h_3+h_4;$

if $head(G)=Low\wedge m_0(sum/4)=m(sum/4)$

then $S(avg):=Low;avg:=declassify(sum/4);$

else skip;

假设初始内存状态 m_0 中 $h_1\sim h_4$ 的值分别为 3，6，8，11，则当该程序执行到分支语句 if 时，由于 $m_0(sum/4)=7\neq m(sum/4)=3$，因此 if 语句的分支表达式不成立，then 分支不会被执行，即降密语句不会被执行，防止了信息非法释放。

4.3　可靠性分析

定理 1　根据图 2 的监控规则，MWhile 语言程序的内联引用监控执行满足“WHAT&WHERE”二维降密策略。

证明：设程序的内联引用监控从初始内存状态 m_0 开始执行，执行过程中产生公开事件序列$\vec{\gamma}_n=\gamma_1\cdots\gamma_n$，下面对该事件序列进行结构化归纳证明。

(1)基础：当 $n=0$ 时，降密策略的两个安全条件 I 和 II 恒成立。

(2)归纳：假设定理对序列长度为 $n-1$ 的公开事件序列$\vec{\gamma}_{n-1}$成立($n\geqslant 1$)，下一步需要证明定理对序列长度为 n 的公开事件序列$\vec{\gamma}_n$依然成立。由于事件 γ_n 的触发只有两种可能性：由降密语句引发或由普通赋值语句触发，因此需要对 γ_n 分情况进行讨论：

1)当 γ_n 是由降密语句触发时，根据内联引用监控规则(TR-DECLASSIFY)，$head(G)=Low$ 和 $m_0(e)=m(e)$ 成立，因此降密策略的安全条件(II)满足；此外，由于$\vec{\gamma}_n$是由降密语句触发，那么降密策略安全条件(I)的蕴涵式的前提为假，则安全条件(I)恒成立。

2)当$\vec{\gamma}_n$是由普通赋值语句触发时，根据内联引用监控规则(TR-ASSIGN)和(TR-SKIP)，不存在非法的直接流；根据规则(TR-IF)和(TR-WHILE)，也不存在非法的间接流，因此攻击者知识没有得到更新，因此降密策略的安全条件(I)成立；另外，由于$\vec{\gamma}_n$不是由降密语句触发的事件，降密策略的安全条件(II)恒成立。

综上，定理的结论成立。证毕。

结束语　信息降密策略是一种弱化的无干扰安全策略，在不同的维度对敏感信息的释放进行控制，防止信息的非法释放。本文基于我们前期工作中提出的二维降密策略，利用内联引用监控方法实施该策略，从而使得降密策略的实施方法脱离外部监控环境，实现自我监控。在后续的工作中，将研究更多维度降密策略及其实施方法，以进一步提高程序的安全性。

参 考 文 献

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

[2] Focardi R, Gorrieri R, Martinelli F. Non Interference for the Analysis of Cryptographic Protocols[C]// International Colloquium on Automata Languages and Programming. Springer-Verlag, 2000:354-372

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

[4] Sabelfeld A, Russo A. From Dynamic to Static and Back: Riding the Roller Coaster of Information-Flow Control Research [C]// Proceedings of 7th International Andrei Ershov Memorial Conference. 2009:352-365

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

[6] Zhu H, Zhuang Y, Chen X. Information Declassification for Multi-Threaded Programs[J]. Applied Mathematics & Information Sciences, 2014, 8(4):1911-1916

[7] Chudnov A, Naumann D A. Information flow monitor in-lining [C]// Proceedings of IEEE Symposium on Computer Security Foundations. 2010:200-214

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

[9] Sridhar M, Hamlen K W. Flexible in-lined reference monitor certification: Challenges and future directions[C]// Proceedings of ACM Workshop on Programming Languages Meets Program Verification. 2011:55-60

[10] 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

[11] 金丽，朱浩. 多线程环境中的准降密策略[J]. 计算机科学，2015，42(12)：243-246