探讨JAVA的安全性改造与验证

肖军模[1]，刘 军[1]，于 泠[1,2]

(1. 解放军理工大学通信工程学院 南京 210007; 2. 南京师范大学数学与计算学院 南京 210097)

【摘要】讨论了OO语言的安全改造问题；给出了以扩展军用安全模型(EMSM)为基础的多级信息流安全控制原理，依据EMSM模型的要求把非安全的JAVA改造成安全语言sJAVA；讨论了sJAVA语言中主要的可执行语句的安全执行条件；以实例说明了sJAVA程序的安全性验证方法。

关 键 词 信息流安全控制; 军用安全模型; 程序安全性验证; 安全程序语言
中图分类号 TP309 文献标识码 A

Research of JAVA Security Reforms and Validation

XIAO Jun-mo[1]，LIU Jun[1]，YU Ling[1,2]

(1. Information Security Research Center of Communication Engineering Institute,
PLA University of Electronic Science and Technology of China Nanjing 210007;
2. College of Mathematics and Computer Science, Nanjing Normal University Nanjing 210097)

Abstract In this paper, the security reforming problems of OO language are discussed, the principles multi-information flow security control principles are presented based on extended military security model. JAVA is transformed into a security program language (sJAVA) depending upon the EMSM. the security execution conditions of main executable sentences in sJAVA are discussed. A validation by an example of sJAVA program is given.

Key words information flow control; military security model; program security validation; security program language

文献[1-3]提出了基于格模型的信息流安全策略，给出了通用程序设计语言中信息流安全性证明方法。本文将JAVA语言改造成安全JAVA(Security JAVA，sJAVA)，探讨了面向对象程序语言的信息流安全性验证方法。

1 信息系统中信息流安全控制的基本模型

军用安全模型是以格作为基础的信息流控制模型[4]，适用于涉密组织(如军队、政府、企业等)的信息系统模型。文献[5]根据实际系统的需要对其进行了扩展。在扩展军用安全模型(EMSM)中，每个格元素或称安全类(Security Class，SC)的形式是(ℓ, h, c)，其中，c为信息子集；$\ell \in L$为敏感级；$h \in H$为层次级。根据格的定义$(\ell_1, h_1, c_1) \geqslant (\ell_2, h_2, c_2)$，当且仅当$(\ell_1, h_1) \geqslant (\ell_2, h_2)$，$c_1 \supseteq c_2$。如果两个安全类$(\ell_1, h_1, c_1)$和$(\ell_2, h_2, c_2)$有共知信息(即$c_1 \cap c_2 \neq \varnothing$)，则允许在两个安全类之间建立一个关联，即：$(\ell_1, h_1, c_1) \xleftrightarrow{P} (\ell_2, h_2, c_2)$。其中，符号$\xleftrightarrow{P}$表示允许两个安全类之间交流其共知的信息子集$P=c_1 \cap c_2$，这是对旧军用安全模型的扩展。利用EMSM的特点可以实现信息系统中信息流的安全控制策略：允许信息由低安全类流向较高的安全类或在同一类内流动，允许两个无关的但存在共知信息的两个类之间交流信息；除此以外的信息交流是非法的。

如果为程序语言中的每个变量、方法或对象都增加一个安全标记(表示安全类)，编译器就能根据EMSM的规则检测信息流的安全性，禁止不符合保密性要求的信息流发生或对其报错，最终生成安全目标程序。如对赋值语句“$V=A$”，编译器生成代码：if $SC(A) \leqslant SC(V)$ then $ADDR(V) \leftarrow VAL(A)$ else error，表示如果A的安全类不高于V(表示为$SC(A) \leqslant SC(V)$)，则可以执行赋值；否则拒绝或报错。

收稿日期：2006－01－25
基金项目：国家重点自然科学基金资助项目(69931040)； 江苏省自然科学基金资助项目(BK2004015)
作者简介：肖军模(1947－)，男，教授，博士生导师，主要从事信息安全与软件工程方面的研究.

2 从JAVA到sJAVA

JAVA虽然利用封装与隐藏机制，在一定程度上保证了信息安全，但还不能保证信息流的保密性。对JAVA语言进行安全性改造的方法为：对JAVA中的包(package)、类、成员(变量或方法)等语言成分增加安全类描述，然后根据EMSM的原则控制程序中的信息安全流动。对JAVA中主要语言成分的安全性改造为[6]：

(1) 对package语句的修改。PackageName Security Flow (ℓ, h, {pid-list})为新增语法成分。其中，pid-list是可以访问本包的包名表，包括本包自己在内；ℓ与h分别表示本包的密级与层级(以下类同)。只有不低于安全类(ℓ, h, {pid-list})的包才能引用PackageName包中的标记public的类。

(2) 对类声明语句的修改。[原各修饰符]class ClassName [extends SuperClassName] Security Flow (ℓ, h, {cid-list})。其中，cid-list允许和本类有关系的其他类名，包括本类自己在内。

(3) 对变量成员声明的修改。[成员的其他属性] VarName Security Flow (ℓ, h, {vid-list})。其中，vid-list表示需要向本成员变量流入信息的变量名或方法名，也包括本变量在内。

(4) 对方法声明语句的修改。改造后的方法为：ReturnType Mname ([fparlist]) Security Flow (ℓ, h, {oid-list})。其中，//{oid-list}为允许引用本方法的类名或方法名的集合；fparlist为形参表，其组成形式是Type x_1 Security Flow (ℓ, h, {vid-list}),…,Type x_m Security Flow (ℓ, h, {vid-list}), Type y_1 Security Flow (ℓ, h, {vid-list}),…,Type y_n Security Flow (ℓ, h, {vid-list}); x_i为传值参数；y_j为传地址参数(可以是变量、数组、方法、对象等名字)；{vid-list}为允许流入x_i或y_j的参数。

本文给出了按EMSM要求定义sJAVA的信息流说明，为每一客体确定了一个固定的安全类。只要流向某个客体的源在该客体规定的流说明中，那么该流的发生就是安全的。在调用方法时，实际参数的安全类仅需满足对形式参数定义的关系。

3 sJAVA的安全执行条件

本文仅讨论sJAVA中与对象有关语句的安全执行条件，这些语句类型在文献[4]中未涉及到。

1) 对外部package中类的引用语句：如果程序包P_B需要从包P_A引入类C_A，仅当SF(P_A)≤SF(P_B)成立，且C_A具有public属性。

2) 对象实例创建语句的安全性条件。ObjectVar Var=New ClassName ([rparlist])。其中，rparlist为与类声明中构造函数形参表fparlist所对应的实参表；Var为ObjectVar的对象实例，其安全类与Objectar的安全类相同。当在类A中创建类B的实例时，要求满足SF(B)≤SF(A)。

3) 对象变量引用的安全性条件。对象变量的引用是ObjectVar・Var。设在对象O_x中执行这种引用，那么其安全条件是SF(O_x)≥SF(ObjectVar・Var)。如果引用需要通过多个“・”运算的串联实现，设其形式为O_1・O_2・…・O_n・ObjectVar・Var，则该引用语句的安全执行条件是SF(O_1)≥SF(O_2)…≥SF(O_n)≥SF(ObjectVar・Var)，该条件表明每个位于“・”运算符左面的对象的安全级不能低于其右边对象(或成员)的安全级。sJAVA中父子对象之间的信息流控制机制也可以类似处理。

4) 方法引用语句Mname ([rparlist])的安全性。假定方法引用语句形如Mname (a_1, a_2,…, a_m, b_1, b_2,…, b_n)。其中，实参a_1,a_2,…,a_m为传值型参数，它们分别对应于形参x_1, x_2,…,x_m；b_1,b_2,…,b_n为传地址类参数，分别对应于形参y_1, y_1, y_2,…, y_n。方法引用语句“Mname(a_1, a_2,…,a_m，b_1, b_2,…,b_n)”的安全条件为：(1) 方法Mname是安全的；(2) 如果SF(x_i)≤SF(y_j)，满足SF(a_i)≤SF(b_j), 1≤i≤m，1≤j≤n；如果SF(y_i)≤SC(y_j)，满足SF(b_i)≤SF(b_j), 1≤i≤n, 1≤j≤n。

假定方法内y_j有函数关系y_j=f(x_i, x_k, y_p)，那么只要满足SF(a_i)≤SF(b_j)，SF(a_k)≤SF(b_j)和SF(b_p)≤SF(b_j)。其中，a_i、a_k、b_p、b_j为分别对应于形参x_i、x_k、y_p和y_j的实参。根据赋值语句安全性条件[4]，y_j=f(x_i, x_k, y_p)的安全执行条件可表示为SF(y_j)≥SF(a_i)⊕SF(a_k)⊕SF(b_p)，⊕是格中最小上界运算符。若用S表示该引用语句，则有SF(S)=SF(b_1)⊙SF(b_2)⊙…⊙SF(b_n)。

5) 线程(Thread)间的同步与通信(notify()及wait())。假定W与N是两个互相通信的线程，notify()及wait()的并发执行导致从N的参数x到W的参数y的隐式流x→y，那么保证流x→y安全的条件是SC(x)≤SC(y)。

由于一个程序由有限语句序列组成，如果序列中每一条语句的执行都是安全的，且程序的执行可正常终止，则程序的执行是安全的。要求程序正常终止是由于程序内部信息能沿着联结于非正常终止的隐蔽信道泄漏出来[4]。

4 sJAVA程序安全性验证实例

本文通过sJAVA的一个实例程序Example，说明各语句的安全性执行条件保证程序的安全。为了简单起见，假定所有安全类说明中ℓ和h都相同。Example类中包括一个方法copy(m, n)，完成把m拷贝给n的操作，m和n的取值都限为0或1。

Import java.long.* ;

Class Example Security Flow ({example, test }) // 除自身外还允许test类访问本类

{Public int a = −1 Security Flow ({a}), b = −1 Security Flow ({a, b});

Public void copy (int m Security Flow({m})，int n Security Flow({m, n})) Security Flow({example, test }) // copy m to n，and value of m and n both are 0 or 1

{ private int k Security Flow ({m});

k=1; // S_1：SF(1)=low≤SF(k)，常数的安全级为low

n= −1; // S_2：SF(−1)=low≤SF(n)

while k= =1 do // S_w：SF(k)≤SF(n)⊙SF(k)={m}，n和k是循环中赋值目标

{ n=n+1; // S_4：SF(n)≤SF(n)

if (n= =0) // S_5：SF(n)≤SF(k)⊙SF(k)= SF(k)，if语句的执行是安全的

k=m // S_6：SF(k)≤SF(m)

else　k=0//S_7：low≤SF(k)

}

if (m!=n)system • out • println(“安全级出错”)

else {a=m; b=m}

}

public static void example (int m Security Flow ({m})，int n Security Flow ({m}))

{//类构造函数的安全类与本类相同

system • out • println (“m_0=”，m，“n_0=”，n); //显示参数m和n

}

}

class test () Security Flow ({test, example})

{ public static void main()

{int m Security Flow({m}), n Security Flow({m, n});

system. • out • readln (m , n);

example ex = New example (m, n) ; // ex即为example的安全类，且有SF(example)≤SF(test)

ex • copy (m, n); // 引用方法copy( )，满足SF(ex)≥SF(copy())，SF(m)≤SF(n)

system • out • println (“m_1=”，ex • a，“n_1=”，ex • b)；//引用成员变量a和b并显示

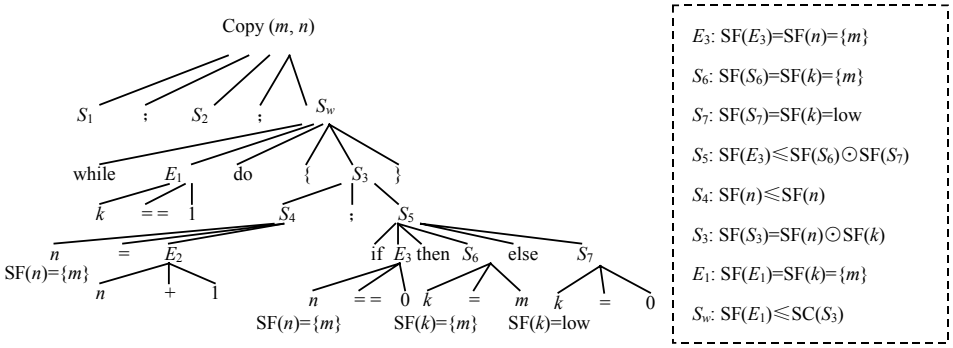

图1　方法copy(n, m)的安全性语义树

在该例中，方法copy( )中各语句右边给出的安全条件都能得到满足，因此方法本身的执行是安全的，图1给出了copy( )的语法和安全性语义树。在类test中，满足SF(m)≤SF(n)，因此可保证方法copy(m, n)的执行安全；由于满足SF(copy())≤SF(ex)，那么引用语句ex • copy (m, n)也是安全的。图1还说明了信息流的安全控制机制与编译过程的融合，在自下而上的分析过程中，每当某个语法树结点形成的时候，同时计算该结点的安全类，或者检查相应语句(或其成分)执行的安全条件是否能够得到满足；如不能满足，则需要报告程序员修改，图中虚线框说明了各结点的安全类或应该满足的安全关系。

5 结 束 语

本文介绍了把JAVA语言改造成sJAVA的方法，给出了保证sJAVA程序安全执行的条件，以实例程序说明了sJAVA程序安全性验证的方法。

(下转第537页)

的ARM)进行优化。

4 结 束 语

本文通过前向安全来实现基于身份的非交互式密钥更新，方案简单，只需用户自身来更新密钥，适合于对安全性要求不高的环境。还可通过入侵弹性实现基于身份的非交互式密钥更新，该方案提供更强的安全性，但需要引入与网络无连接的、计算资源和存储资源受到限制的私有设备来帮助更新密钥，需要进一步的研究。

参 考 文 献

[1] SHAMIR A. Identity-based cryptosystems and signature schemes[C]//Proc. of Crypto'84, LNCS 196. Berlin: Springer-Verlag, 1985: 47-53.

[2] BONEH D, FRANKLIN M. Identity-based encryption from the Weil pairing[C]//Proc. of Crypto'01, LNCS 2139. Berlin: Springer-Verlag, 2001: 213-229.

[3] CANNETI R, HALEVI S, KATZ J. A forward secure public key encryption scheme[C]//Proc. of Eurocrypt'03, LNCS 2656.Berlin: Springer-Verlag, 2003: 255-271.

[4] DODIS Y, KATZ J, XU S. Key-insulated public key cryptosystems[C]//Proc. of Eurocrypt'02, LNCS 2332. Berlin: Springer-Verlag, 2002: 65-82.

[5] DODIS Y, FRANKLIN M, KATZ J. Intrusion-resilient public-key encryption[C]//Proc. of CT-RSA'03, LNCS 2612, Berlin: Springer-Verlag, 2003:19-32.

[6] FUJISAKI E, OKAMOTO T. Secure integration of asymmetric and symmetric encryption schemes[C]//Proc. of Crypto'99, LNCS 1666. Berlin: Springer-Verlag, 1999: 537-554.

[7] 毛文波. 现代密码学: 理论与实践[M]. 北京: 电子工业出版社, 2004.

编 辑 黄 莘

(上接第530页)

5 结 论

本文将粗糙集理论引入数据库的推理泄漏控制，可以提取出数据库中非敏感和敏感数据之间蕴含的确定性推理规则。根据这些规则和属性的重要程度，在保证数据库推理泄漏控制的前提下，实现发布给用户数据量修改的最小化。同现有方法相比，该方法计算量小，可扩展性强，在保证大规模数据库可用的同时，增加了数据库的安全性。

参 考 文 献

[1] MARKS D. Inference in MLS database system[J]. IEEE Trans Knowledge and Data Eng, 1996, 8(1): 46-55.

[2] DAWSON S, VIMERCATI S D C. Minimal data upgrading to prevent inference and association attacks[C]// In: Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. Pennsylvania: ACM Press, 1999: 114-125.

[3] SHAFER G. Detecting inference attacks using association rules[J/OL]. http://www. glennshafer. com/courses/ downloads/ raman.pdf, 2004-04-18.

[4] CHANG L, MOSKOWITZ S. A study of inference problem in distributed database systems[C]//In: Proceedings of IFIP Data Security and Applications. Cambridge, UK: Cambridge Uni. Press, 2002: 229-243.

[5] 刘 清. Rough集及Rough推理[M]. 北京: 科学出版社, 2001.

[6] ULLMAN J D. Principle of database and knowledge-base system, Vols. I and II[M]. Rockville, MD: Computer Science Press, 1988,1989.

编 辑 漆 蓉

(上接第533页)

参 考 文 献

[1] DENNING D E. Secure information flow in computer systems[D]. W. Lafayette, Ind.: Purdue Univ., 1975.

[2] DENNING D E. A lattice model of secure information flow[J]. COMM ACM, 1976, 19(5): 236-243.

[3] DENNING D E, DENNING P J. Certification of program for secure information flow[J]. COMM ACM, 1977, 20(7): 504-513.

[4] 肖军模, 刘 军, 周海刚. 网络信息安全[M]. 北京: 机械工业出版社, 2006

[5] 肖军模. 对军用安全模型的扩展[J]. 电子科技大学学报, 2005, 34(2): 186-189.

[6] 江义华. JAVA完美经典[M]. 北京: 中国铁道出版社, 2004.

编 辑 黄 莘