基于依赖分析的云组合服务信息流控制机制

刘明聪[1,2] 王 娜[1,2] 周 宁[3]
(信息工程大学 郑州 450001)[1] (河南省信息安全重点实验室 郑州 450001)[2]
(江南计算技术研究所 江苏 无锡 214083)[3]

摘 要 云组合服务可以为用户提供更加丰富的功能,但在业务流程中敏感信息可能流经多个云服务,必须实施信息流控制来防止信息的泄露或非授权访问。针对云组合服务的信息流安全问题,提出了一种基于依赖分析的信息流控制机制,通过数据间的依赖关系分析云组合服务中的信息流动,并使用安全标签进行信息流控制。首先,构建了复杂组合结构的云组合服务加权有向图模型,基于安全属性定义了云服务的属性证书、数据的机密性标签以及完整性标签;接着,提出了服务内部输入依赖与服务间资源依赖的概念,并给出了基于历史信息的运行时输入依赖与资源依赖计算方法;其次,根据依赖分析给出了输出数据安全标签算法,定义了组合信息流策略并设计了分布式的信息流控制机制,实现了复杂组合结构下云组合服务中信息流的机密性和完整性保护;最后,分析评估了机制的有效性与性能。

关键词 云服务,服务组合,信息流,数据依赖,安全标签
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.04.030

Dependency Analysis Based Cloud Composition Service Information Flow Control Mechanism

LIU Ming-cong[1,2] WANG Na[1,2] ZHOU Ning[3]
(Information Engineering University, Zhengzhou 450001, China)[1]
(Henan Province Key Laboratory of Information Security, Zhengzhou 450001, China)[2]
(Jiangnan Institute of Computing Technology, Wuxi, Jiangsu 214083, China)[3]

Abstract Cloud composition service can provide users with richer capabilities, but sensitive information may flow through multiple cloud services in business process, so information flow control must be implemented to prevent information leakage or unauthorized access. Aiming at the security problem of information flow in cloud composite service, this paper proposed a data flow control mechanism based on dependency analysis. The information flow in cloud composite service was analyzed by the dependency between data and the information flow was controlled by using security label. Firstly, a cloud composition service weighted directed graph model with complex combination structure is constructed. Based on the security attributes, the attribute certificate of cloud service, the confidentiality label and integrity label of data are defined, then the input dependencies between services and resource dependencies between services are proposed, and the input dependence and resource dependency computing method based on historical information are given. After that, the output data security label algorithm is given according to the dependency analysis, the compositional information flow policy is defined and the distributed information flow control mechanism is designed, realizing the confidentiality and integrity protection of information flow in cloud composition service under complex compositional structure. At last, an example is given to anaylze the effectiveness and performance of the mechanism.

Keywords Cloud service, Service composition, Information flow, Data dependency, Security label

1 引言

云计算以其灵活、按需的服务模式得到了广泛应用,越来越多的企业和个人将其服务、应用部署到云端[1]。通过对云平台上大量的基础服务进行组合,能以较低成本创建复杂的、功能强大的组合服务,从而促进广泛的业务协作和资源共享[2]。但是,在组件服务的数据处理与服务间消息转发的过程中,敏感信息或污染数据可能会随着业务流程流入其他组件服务,造成严重的安全问题[3-4]。

为了保护云服务的数据安全,访问控制与信息流控制技术开始被应用在云服务中[5-6]。但是简单地对单独的云服务制定安全机制并不能完全解决问题,例如,在图1所示的组合服务中,云服务 s_1,s_2,s_3 被组合到同一个业务流程中,云服务 s_1 使用本地数据 d_1 计算得到输出数据 o_1,然后 s_1 将其发送

到稿日期:2018-03-02 返修日期:2018-05-24 本文受国家自然科学基金资助项目(61802436,61502531),国家863计划项目(2015AA016006),河南省自然科学基金项目(162300410334)资助。

刘明聪(1993-),男,硕士生,主要研究方向为云计算、信息流安全;王 娜(1980-),女,博士,副教授,主要研究方向为复杂系统环境下的信任管理,E-mail:twftina_w@126.com(通信作者);周 宁(1978-),女,硕士,工程师,主要研究方向为信息安全。

给 s_2，s_2 使用 o_1 与本地数据 d_2 计算得到输出数据 o_2，并将其发送给 s_3，s_3 从 o_2 中得到 d_1 与 d_2 的信息。但是，根据 d_1 的安全策略，s_3 不能读取数据 d_1，因此该组合服务破坏了数据 d_1 的机密性。由上述例子可以看出，若要解决云组合服务的信息流问题，必须从组合服务整体出发，制定全局的信息流控制机制[7]。

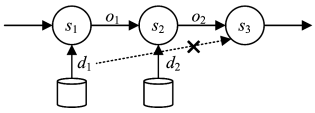

图 1 云组合服务中信息流的安全问题

Fig. 1 Security problem of information flow in cloud composite service

目前，已有学者对组合服务的信息流控制机制展开研究。Hutter 等[8]提出了一种在动态服务组合中基于类型的信息流控制方法，以保护数据在计算与传播过程中的安全。She 等[9]于 2009 年提出了一种面向服务链的信息流控制方法(Service Chain Information Flow Control，SCIFC)，该方法为服务链中的信息流建立了一种安全增强模型，并且提出了请求流和响应流中的控制协议，以防止服务链中敏感信息泄漏给不被信任的服务。2011 年，She 等[10]进一步提出了面向运行中服务链的信息流控制方法，该方法在分析服务链运行过程中，本地访问数据与动态产生数据间依赖关系的基础上，给出了服务链中敏感数据的释放策略与输入数据的接收策略。基于 She 等的工作，Yu 等[11]提出了一种新的面向服务链的基于标签的分布式信息流控制方法(Label-Based Information Flow Control，LBIFC)，以改进 SCIFC 中数据对象利益相关者的计算低效的问题。Xi 等[12]提出了一种面向云计算环境的基于信息流控制的分布式服务组合方法，该方法首先定义了面向组合服务的多级安全模型，在组合过程中先验证每个组件服务是否满足该模型，然后再进行服务组合，以确保所构建组合服务的信息流安全。Solanki 等[13]针对面向服务的系统提出了一种访问与信息流控制方法，该方法为每个数据对象附上数据依赖列表，在服务主体访问数据对象时不仅能进行传统的访问控制，而且能判断其是否满足数据依赖表中相关域的信息流策略，以保护多域环境下服务组合时的信息流安全。

但是，上述研究对数据安全属性的考虑不足，仅考虑了对数据机密性的保护，不支持完整性保护；而且上述研究多局限于服务链这种简单的组合结构，其成果并不适用于存在选择结构、循环结构等复杂结构的组合服务。复杂组合结构下云组合服务中的信息流动是动态的，敏感信息可能随着不同的运行状态流向不同的位置。由于产生信息流的动态性，因此各数据间可能存在的联系也是动态的，现有的访问控制或信息流控制机制不能表达数据间的这种关联性与动态性，很可能导致信息泄漏与非授权访问等问题。如何准确地分析运行过程中各数据之间的信息流动情况，对云组合服务实施信息流控制是十分关键的。

本文针对复杂组合结构下云组合服务中的信息流安全问题，提出了一种基于依赖分析的信息流控制机制(Dependency Analysis Based Information Flow Control，DABIFC)。首先，构建了基于加权有向图的云组合服务模型，为实现支持复杂组合结构的信息流控制机制奠定了基础。随后，引入了云服务的属性证书与数据的安全标签，以支持基于属性的信息流控制，从而实现数据的机密性与完整性保护。为了准确地分析云组合服务运行时各组件服务内部与服务间各个数据之间的信息流动关系，本文提出输入依赖与资源依赖的概念，并分别设计了基于历史信息的运行时输入依赖计算方法和服务间资源依赖的计算方法；在此基础上，根据输入依赖给出云服务输出数据安全标签的继承和转换方法，以实现云服务输出数据安全标签的准确设置，保障服务内部信息流的安全；根据资源依赖，给出了组合信息流策略，以及分布式的信息流控制机制，以实现云组合服务间的信息流安全；最后，分析并评估了机制的有效性。

2 云组合服务模型

云服务是遵循提供者定义的限制和规则并通过接口访问的一种或多种功能的实现机制。云服务的抽象模型的定义如下。

定义 1 云服务 s 是一个四元组$\langle s.R, s.In, s.Out, s.F\rangle$，其中：

1)$s.R=\{s.r_1, s.r_2, \cdots\}$表示云服务 s 中所有参与计算的本地数据集合。

2)$s.In=s.In^L \cup s.In^N$ 表示云服务 s 的输入数据集合，且 $s.In^L=\{s.in_1^L, s.in_2^L, \cdots\}$ 表示本地输入数据的集合，$s.In^N=\{s.in_1^N, s.in_2^N, \cdots\}$表示来自其他服务的输入数据的集合。

3)$s.Out=s.Out^L \cup s.Out^N$表示云服务 s 的输出数据集合，且 $s.Out^L=\{s.out_1^L, s.out_2^L, \cdots\}$表示本地输出数据集合，$s.Out^N=\{s.out_1^N, s.out_2^N, \cdots\}$表示输出到其他服务的数据集合。

4)$s.F$ 表示云服务 s 的计算过程，此过程将输入 $s.In$ 转换为输出 $s.Out$，$s.Out=s.F(s.In)$。

云服务模型如图 2 所示。

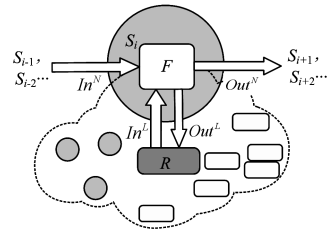

图 2 云服务模型

Fig. 2 Cloud service model

云服务中，数据的输入输出对 $s.R$ 进行写入或读取，如果某本地输入数据 $s.in_i^L$($s.in_i^L \in s.In^L$)或某本地输出数据 $s.out_i^L$($s.out_i^L \in s.Out^L$)写入或读取的对象是某本地数据 $s.r_j$，则可将数据表示为 $R(s.in_i^L)$或 $R(s.out_i^L)$，即 $s.r_j=R(s.in_i^L)$或 $s.r_j=R(s.out_i^L)$。

$s.F$ 由基本操作、程序结构和基本计算过程 3 个要素构成。基本操作有输入操作 $Input(in, v)$、输出操作 $Output(out, v)$、空操作 $skip$ 和赋值操作 $v:=u$；程序结构有顺序结构、选择结构(if…else…)和循环结构(while…do…)；在操作或结构语句中，基本的计算过程有数值/逻辑运算 $f(u_1, \cdots,$

u_n）和条件运算 $con(u_1,\cdots,u_n)$，当然计算的参数可能为空，表示为 $f()$或 $con()$。$s.F$ 可抽象描述为：

$F::=a;F$

$a::=skip\mid Input(in,v)\mid Output(out,v)\mid v:=f(u_1,\cdots,u_n)\mid a_1;a_2\mid$ if $con(u_1,\cdots,u_n)$ then a_1 else a_2 end $\mid$ while $con(u_1,\cdots,u_n)$ do a_1 end

云组合服务是多个独立的云服务有机组合到一起形成的应用系统，其定义了服务间的组合关系。

定义 2　一个云组合服务系统 C 可以描述为一个加权有向图 (S,G,Φ,I_Φ)。其中，S 是顶点集，是参与组合的所有云服务的集合；G 是边集，表示云服务间的组合关系，若服务 s_i，s_j 之间存在组合关系，则图中存在一条从顶点 s_i 到 s_j 的有向边，记为$\langle s_i,s_j\rangle$；Φ 是图中边上权值的集合，ϕ 是某组合关系的执行条件，对于任意的 $\phi(\phi\in\Phi)$，当条件取值为真时表示执行此组合关系，否则不执行；$I:\Phi\to G$ 是权值与边的映射函数，描述条件与组合关系之间的对应关系。

定义 2 可以用来形式化地刻画一个复杂的云组合服务结构。在这个云组合服务中可存在多种组合结构，包括顺序结构、分支结构、选择结构与循环结构等，而信息就通过这些复杂的组合关系在各个云服务间流动。

定义 3　在云组合服务的加权有向图中，如果顶点 s_i 到顶点 s_j 之间存在一条路径，并且在一次执行中路径上的权值取值均为真时，则称 s_i 到 s_j 是可达的。

用 $Pre(s_i)$表示所有到 s_i 可达的节点，$Suc(s_i)$表示所有从 s_i 可达的节点。s_i^- 表示到服务 s_i 直接可达的前趋服务集合，$s_i^-=\{s_j\mid s_j\in S,\langle s_j,s_i\rangle\in G$ 且 s_j 到 s_i 是可达的$\}$；s_i^+ 表示从服务 s_i 直接可达的后继服务集合，$s_i^+=\{s_j\mid s_j\in S,\langle s_i,s_j\rangle\in G$ 且 s_i 到 s_j 是可达的$\}$。

3　属性证书与安全标签

为了解决云组合服务中细粒度的信息流控制问题，支持数据的机密性与完整性保护，本文采用了一种基于属性的信息流控制机制。该机制的基本思路是，为每个云服务和数据都授予一个或多个安全属性，以表征云服务或数据的某种信息流特征或安全性类型，如安全属性“*Confidential*”表示对象含有机密信息或者有机密信息的处理能力；每个云服务根据服务与数据的安全属性和自身基于属性的信息流策略，独立地判断是否允许某次信息流动，从而实现分布式的信息流控制。

本文认为属于同一个提供者的云服务和数据位于同一个安全域中，安全域是独立实施安全策略的单元。在每个安全域中，引入一个安全中心（Security Authority，SA）（见图 3），来负责本域内云服务、数据安全属性和策略的管理。

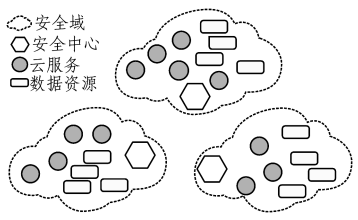

图 3　安全域与安全中心

Fig. 3　Security domain and security center

云服务和数据安全属性的表述形式是不同的，云服务使用属性证书来声明自己的安全属性，属性证书可在各服务间传递交换，而数据安全属性使用安全标签的形式来表征，并且安全标签随着数据的流动在服务间传递。

3.1　云服务的属性证书

云服务通过属性证书来声明自身的安全属性。云服务属性证书的定义如下。

定义 4　云服务 s 的属性证书 ac 至少含有 5 个部分的内容：*Owner*，*Issuer*，*AttributeSet*，*Effective time*，*Signature*，其中，*Owner* 表示证书所有者即云服务的身份；*Issuer* 表示证书的签发机构，即某个 *SA*；*AttributeSet* 表示云服务具有的属性集合；*Effective time* 表示证书的有效期；*Signature* 表示颁发机构的签名。

一个云服务可能拥有多个属性证书，即除本域 *SA* 外，其他域 *SA* 也可为该云服务签发属性证书，以授予该云服务访问本域数据资源的权限。云服务 s_i 拥有的服务 s_j 所属域 *SA* 签发的属性证书可表示为 $dom(s_j).AC(s_i)$（其中，$dom(s)$表示云服务 s 所属的安全域，且这里 s_j 可以是 s_i）。属性证书可以用 XML 语言来描述，如图 4 所示。

```
〈AttributeCertificate〉
  〈Owner id=“s1”/〉
  〈Issue id=“dom1”/〉
  〈AttributeSet〉
    〈Attribute attr1=“a1”/〉
    〈Attribute attr2=“a2”/〉
  〈AttributeSet/〉
  〈 Effective_time end=“2018.01.01” /〉
  〈Signature〉
    〈 SignatureAlgorithm algorithm=“RSA with sha1”/〉
    〈SignatureValue value=“signaturevalue”/〉
  〈Signature/〉
〈AttributeCertificate/〉
```

图 4　云服务属性证书示例

Fig. 4　Cloud service attribute certificate example

3.2　数据的安全标签

本文主要从机密性和完整性两个角度来描述数据的安全属性。数据的机密性属性体现出数据的防泄露需求，要求控制信息流向高安全等级实体；完整性属性体现出数据的真实性需求，要求信息来自可信的实体[14]，因此数据的安全标签由两个分别体现机密性与完整性的标签集合构成。

定义 5　数据 X 的安全标签 $L(X)$可表示为二元组$(L_S(X),L_I(X))$。其中，$L_S(X)$表示 X 的机密性属性的集合，$L_I(X)$表示 X 的完整性属性的集合。

为了实施信息流控制，在云服务模型中每个输入输出数据都应当具有安全标签，输入数据的安全标签是由上一个服务或者本域 *SA* 决定的，而输出数据的安全标签由当前的服务确定。下文将对云服务的数据依赖关系进行分析，给出输出数据安全标签的计算方法。

4　数据依赖分析

在云组合服务中，云服务 s_i 的输出数据 $s_i.out_j$ 可能由本

地输入 $s_i.in_k^L$ 或上个服务的输入 $s_i.in_l^N$ 计算得到，而 $s_i.in_k^L$ 和 $s_i.in_l^N$ 可能具有不同的安全标签，为了能够确定输出数据 $s_i.out_k$ 的安全标签，应当知道其与各个输入数据之间的关系。本文使用依赖关系刻画这种输出数据与其信息来源之间的关系。如果输出数据 v 中的信息来源于输入数据 u，那么就称数据 v 依赖于 u。显然，数据 v 是否依赖于 u 的问题等价于是否存在从 u 到 v 的信息流的问题，即如果存在从 u 到 v 的信息流则 v 依赖于 u，反之亦然。

4.1 服务内输入依赖分析

在 $s.F$ 的计算过程中，操作(如赋值操作)产生了直接的信息流，信息由参与计算的数据直接流向计算结果，直接信息流可分为显式信息流和隐式信息流。

定义 6 在 $s.F$ 中，输入、输出或赋值操作产生的信息流称作显式信息流；而在条件或循环结构中，条件语句中的某数据与结构体中某数据间存在的信息流称作隐式信息流。

定义 7 若存在从数据 u 到数据 v 的显式信息流，则称数据 v 显式依赖数据 u，记为 $u\in Dep_E(v)$，其中 $Dep_E(v)$ 表示数据 v 的显式依赖数据集合；若存在从数据 u 到数据 v 的隐式信息流，则称数据 v 隐式依赖数据 u，记为 $u\in Dep_I(v)$，其中 $Dep_I(v)$ 表示数据 v 的隐式依赖数据集合。

例如，“if $u=1$ then $v:=k$ else skip end”中，当 $u=1$ 时，存在从 k 到 v 的显式信息流与从 u 到 v 的隐式信息流。即 v 显式依赖于 k，记作 $k\in Dep_E(v)$，v 隐式依赖于 u，记作 $u\in Dep_I(v)$。

信息流具有传递性，由多个直接信息流依次传递引起的信息流动称为间接信息流。如“$v:=k;u:=v$”中，从 k 到 v 的直接信息流与从 v 到 u 的直接信息流构成了从 k 到 u 的间接信息流。

依赖关系也具有传递性，而且显式依赖与隐式依赖的传递性质不同。例如，只有数据 v 显式依赖于 u 且 u 显式依赖于 w 时，v 才显式依赖于 w。依赖关系的传递性可以表示为：

$k\in Dep_E(u)\wedge u\in Dep_E(v)\Rightarrow k\in Dep_E(v)$；

$k\in Dep_E(u)\wedge u\in Dep_I(v)\Rightarrow k\in Dep_I(v)$；

$k\in Dep_I(u)\wedge u\in Dep_E(v)\Rightarrow k\in Dep_I(v)$；

$k\in Dep_I(u)\wedge u\in Dep_I(v)\Rightarrow k\in Dep_I(v)$。

由上述分析可知，完整的服务计算过程会产生从输入数据到输出数据的信息流，也会产生输出数据对输入数据的依赖关系，本文把这种依赖关系定义为输入依赖。

定义 8 对于云服务 s，$s.out_i\in s.Out$ 是云服务的某个输出数据，如果输出数据 $s.out_i$ 显式依赖或隐式依赖于某个输入数据 $s.in_j\in s.In$，则称 $s.out_i$ 输入依赖于 $s.in_j$，记作 $s.out_i\in Dep_{IN}(s.in_i)$，$Dep_{IN}(s.in_i)$ 表示 $s.in_i$ 依赖的输入数据集合。

分析云服务内部的信息流是生成依赖关系的基础。显式依赖关系可以直接由赋值操作或输入输出操作得到，隐式依赖则比较复杂。为每个云服务 s_i 都维护一个栈 $St(s_i)$，可实现服务运行时的隐性依赖分析。当程序开始执行选择或循环结构时，就将条件中的全部变量作为一个集合元素压入栈中，而当结束一个选择或循环结构时，就将栈顶的元素弹出。$St(s_i)$ 中的元素表示当前正在计算变量的隐式依赖关系。

但是，对于如下代码段：

$x=$true；$y=$false；$v=x$；$w=y$；

if u then $v:=y$ else skip end；

if $v=y$ then $w:=x$ else skip end；

按照执行路径分析，当 $u=$true 时，v 隐式依赖于 u，w 隐式依赖于 v，因此 w 隐式依赖于 u；当 $u=$false 时，v 和 w 都不依赖于 u。事实上，当 $u=$true 时，$v=$false 且 $w=$true，而 $u=$false 时，$v=$true 且 $w=$false，显然 v 与 w 的值始终由 u 决定。这意味着，当 $u=$false 时，v 和 w 都依赖于 u。上述例子说明，在某些选择结构中，信息可能会通过服务未运行的部分进行传播[15]。因此，为了更加准确地分析出隐式依赖关系，必须采取措施检查未运行的程序路径上可能的信息流动。

基于此，本文引入历史信息，提出了一种基于历史信息的运行时隐式依赖分析方法。对所有已经运行的路径保存数据的隐式依赖，在下次运行分析时，将历史信息与分析结果合并得到最终的隐式依赖。即数据 v 的隐式依赖历史信息 $Dep_I^*(v)=\bigcup_k Dep_I^k(v)$，$Dep_I^k(v)$ 是第 k 次运行云服务 s_i 所得的数据 v 的隐式依赖。应当注意，如果云服务运行历史不能覆盖其所有的路径，那么这里得到的隐式依赖仍然是不完全的。为了覆盖所有的路径，可以用测试集合进行离线的训练以尽可能地使结果贴近真实值。

结合以上所有考虑，下面给出在云服务运行时输出数据输入依赖的动态生成规则。

$R1:Input(v,s.in_i)\Rightarrow Dep_E(v)=\{s.in_i\}\wedge Dep_I(v)=\{\}$

$R2:Output(s.out_i,v)\Rightarrow Dep_{IN}(s.out_i)=Dep_E(v)\cup Dep_I(v)$

$R3:$if $con(u_1,\cdots,u_n)\Rightarrow St.push(\{u_1,\cdots,u_n\})$

$R4:$while $con(u_1,\cdots,u_n)\Rightarrow St.push(\{u_1,\cdots,u_n\})$

$R5:$end$\Rightarrow St.pop$

$R6:v:=op(u_1,\cdots,u_n)\wedge St=\perp\Rightarrow Dep_E(v)=\bigcup_{i=1}^{n}Dep_E(u_i)\wedge Dep_I(v)=\bigcup_{i=1}^{n}Dep_I(u_i)-Dep_E(v)$

$R7:v:=op(u_1,\cdots,u_n)\wedge St\neq\perp\Rightarrow Dep_E(v)=\bigcup_{i=1}^{n}Dep_E(u_i)\wedge Dep_I(v)=\bigcup_{i=1}^{n}Dep_I(u_i)\cup Dep(St)-Dep_E(v)$，其中 for all $w_k\in\bigcup_j St[j]$，$Dep(St)=\bigcup_k(Dep_E(w_k)\cup Dep_I(w_k))$

$R8:\forall v,Dep_I(v)=Dep_I(v)\cup Dep_I^*(v)$

其中，$R1$ 与 $R2$ 描述了输入操作、输出操作的规则，即由外部输入直接赋值的数据显式依赖于输入数据，输出数据依赖于对其直接赋值的数据的显式依赖与隐式依赖。$R3$ 与 $R4$ 描述了选择与循环结构的规则，利用栈来记录控制语句(选择或循环)产生的隐式依赖，将条件计算中使用的变量集合入栈。$R5$ 表示选择或循环结构结束时的操作，起控制作用的变量不再与后面的数据构成依赖关系，因此将其出栈。$R6$ 与 $R7$ 则是描述赋值操作中显式依赖与隐式依赖的计算，数据 v 显式依赖于参与数值/逻辑运算的有数据，根据传递性其输入依赖等于这些数据的输入依赖的并集。赋值操作中，当栈为空时

数据的隐式依赖是参与运算的数据的隐式依赖的并集；当栈不为空时，隐式依赖为栈中所有元素包含的数据的依赖关系和所有参与运算的所数据的隐式依赖的并集。由于显式依赖强于隐式依赖，为了不重复记录，还应当删除与显式依赖重复的部分。$R8$ 描述了如何利用历史依赖信息来修正本次运行的分析结果。

4.2 服务间的资源依赖分析

由于云服务之间的组合关系，数据间的依赖关系不仅存在于云服务内部，也存在于云服务间。由于云服务的输出中可能带有本地数据的信息，因此存在从本地数据到其后继服务的输出数据的信息流动。

定义 9　若云服务 s_i 的输出数据 $s_i.out_k \in s_i.Out$ 中，含有其前趋服务 $s_j(s_j \in Pre(s_i))$ 的本地数据 $s_j.r_l$ 中的信息，则称其与之存在资源依赖，记作 $s_j.r_l \in Dep_R(s_i.out_k)$。

资源依赖具有传递性。例如，云服务 s_i 与 s_j 中存在的资源依赖关系为 $s_i.in_m \in Dep_{IN}(s_i.out_k) \wedge s_j.r_n \in Dep_R(s_i.in_m)$，根据传递性可得到 $s_j.r_n \in Dep_R(s_i.out_k)$。云服务的输出数据的资源依赖算法如算法 1 所示。

算法 1　云服务 s 中的资源依赖的计算

Input：$Dep_{IN}(s.out_i)$，$Dep_R(s.in_j)$，其中，$s.out_i \in s.Out$，$s.in_j \in s.In$，$\forall i,j, 1 \leqslant i \leqslant |s.Out|, 1 \leqslant j \leqslant |s.In|$

Output：$Dep_R(s.out_i)$，$\forall i, 1 \leqslant i \leqslant |s.Out|$

1. For each $s.out_i$ in $s.Out$：

1.1 For each $s.in_j$ in $Dep_{IN}(s.out_i)$：

1.1.1 $Dep_R(s.out_i) = Dep_R(s.out_i) \cup Dep_R(s.in_j)$；

1.1.2 If $s.in_j \in s.In^L$ then

$Dep_R(s.out_i) = Dep_R(s.out_i) \cup \{R(s.in_j)\}$；

需要注意的是，云服务 s 的本地数据 $s.R$ 也具有资源依赖关系，并且可通过本地的输入输出传递这种依赖关系。当从本地输入数据 $s.in_i^L(s.in_i^L \in s.In^L)$ 时，输入数据继承本地数据 $R(s.in_i^L)$ 的资源依赖，$Dep_R(s.in_i^L) = \{R(s.in_i^L)\} \cup Dep_R(R(s.in_i^L))$；当输出本地数据 $s.out_i^L(s.out_i^L \in s.Out^L)$ 时，更新本地数据 $R(s.out_i^L)$ 的资源依赖，$Dep_R(R(s.out_i^L)) = Dep_R(s.out_i^L)$。

本节根据云组合服务中的信息流，分析了数据之间的依赖关系，下文将描述如何根据这些依赖关系来实现信息流的控制，以保护信息流动的安全性。云组合服务的信息流控制可分为两部分：1）云服务内的信息流控制，用以保证云服务内部的数据处理过程的安全性；2）组合信息流控制，用以在转发数据时验证其他服务是否符合本服务的信息流安全策略。

5 安全标签的继承与转换

云服务的数据处理过程是指从输入数据到计算出输出数据的过程，在该过程中要求保持数据的安全属性不被破坏，也就是说，云服务内信息流控制要求保证输出数据安全标签的正确性，包括继承与转换。

云服务输出数据的安全标签可以通过其输入依赖来确定。对于云服务 s 的输出数据 $s.out_i$，可以计算出其输入依赖 $Dep_{IN}(s.out_i)$。对于所有的 $s.in_j \in Dep_{IN}(s.out_i)$，为了保证信息在流动过程中的机密性与完整性不被破坏，当且仅当 $L_S(s.in_j) \subseteq L_S(s.out_i) \wedge L_I(s.out_i) \subseteq L_I(s.in_j)$ 时，允许存在从 $s.in_j$ 到 $s.out_i$ 的信息流，因此 $\forall s.in_j \in Dep_{IN}(s.out_i)$，$L_S(s.in_j) \subseteq L_S(s.out_i) \wedge L_I(s.out_i) \subseteq L_I(s.in_j)$。为了保证数据的可用性，输出数据的标签在满足以上条件的同时应当取到极值。取所有输入依赖的机密性标签的并集作为输出的机密性标签，所有输入依赖的完整性标签的交集为输出的完整性标签。这种由输入依赖关系决定安全标签的过程被称为标签的继承。特别地，本地输入数据 $s.in_i^L(s.in_i^L \in s.In^L)$ 将继承本地数据 $R(s.in_i^L)$ 的安全标签 $L(s.in_i^L) = L(R(s.in_i^L))$。

在实际业务过程中，数据经过加工后其原有的安全属性可能会发生改变，如原有的机密数据通过加密或匿名处理后会变成可以公开的数据，来源未知的数据得到权威组织承认后会变得可信等。在这种情况下，标签继承将不能真实反映数据实际的安全属性。另外，标签的继承性会导致数据的机密性越来越高，完整性越来越低，从而逐渐降低数据的可用性。据此，本文提出云服务转换因子的概念，以体现数据经过云服务处理后安全标签的调整情况，该过程被称为标签的转换。

定义 10　设云服务 s 的转换因子为 $TF(s)$，规定了对输出数据安全标记的增删操作。$TF(s)$ 可表示为四元组 $\langle tf_S^+(s), tf_S^-(s), tf_I^+(s), tf_I^-(s)\rangle$，$tf_S^+(s)$ 表示输出数据机密性标签中需增加标签的集合，$tf_S^-(s)$ 表示输出数据机密性标签中需去除标签的集合，$tf_I^+(s)$ 表示输出数据完整性标签中需增加标签的集合，$tf_I^-(s)$ 表示输出数据完整性标签中需去除标签的集合。

显然，$tf_S^+(s)$，$tf_S^-(s)$，$f_I^+(s)$，$tf_I^-(s)$ 均可以为空，表示无需对标签进行调整。转换因子是由云服务根据数据的计算过程而定的，本文仅考虑最特殊的情况，即所有输出数据的安全标签都使用同一个转换因子来进行转换。

输出数据在继承了输入依赖的安全标签之后，经过转换因子得到最终的安全标签，输出数据的安全标签计算过程如算法 2 所示。

算法 2　云服务 s 中输出数据的安全标签的计算

Input：$TF(s)$，$Dep_{IN}(s.out_i)$，$L(s.in_j)$，其中，$s.out_i \in s.Out$，$s.in_j \in s.In$，$\forall i,j, 1 \leqslant i \leqslant |s.Out|, 1 \leqslant j \leqslant |s.In|$

Output：$L(s.out_i)$，$\forall i, 1 \leqslant i \leqslant |s.Out|$

1. For each $s.out_i$ in $s.Out$：

1.1 For each $s.in_j$ in $Dep_{IN}(s.out_i)$：

1.1.1 $L_S(s.out_i) = L_S(s.out_i) \cup L_S(s.in_j)$；

1.1.2 $L_I(s.out_i) = L_I(s.out_i) \cap L_I(s.in_j)$；

1.2 $L_S(s.out_i) = L_S(s.out_i) \cup tf_S^+(s) - tf_S^-(s)$；

1.3 $L_I(s.out_i) = L_I(s.out_i) \cup f_I^+(s) - f_I^-(s)$；

6 组合信息流控制

由于云组合服务中服务众多、结构多样，信息流动复杂，在云服务的数据转发过程中，被转发的数据可能会存在对其他服务的资源依赖。这种依赖可能会导致数据在输入输出时

访问其他服务数据资源，当某个数据被发送到 s_j 时，如果它资源依赖于 s_i 的某个本地数据，那么就相当于发生了 s_j 对 s_i 的数据资源的读访问；当某个数据被 s_i 写入本地时，如果它资源依赖于 s_j 的本地数据，那么就相当于发生了 s_j 对 s_i 的数据资源的写访问。这两种访问产生的信息流并不是云服务可以单独进行安全验证的。为了保护所有服务的数据资源的安全性，需要制定相应的组合信息流策略。

6.1 组合信息流策略

组合信息流策略可分为读策略与写策略。在云服务组合中，每个云服务的读策略用于验证其他服务对其数据资源的读访问，写策略用于验证其他服务对其数据资源的写访问。这些策略是通过检查数据的安全标签以及云服务的属性证书实现的。另外，由于云服务的转换因子涉及改变数据的安全属性，因此在对服务进行验证时，必须把转换因子考虑在内。

定义 11 云服务 s_i 的读策略集 $RIFC(s_i)$ 规定了是否允许服务 $s_k(s_k \in Suc(s_i) \cup \{s_i\})$ 向直接后继服务 $s_j(s_j \in s_k^+)$ 输出满足下列条件的数据 $s_k.out_l^N$：1）$s_k.out_l^N \in s_j.Out^N$；2）$s_i.r_m \in Dep_R(s_k.out_l^N)$，其中，$s_i.r_m \in s_i.R$，即 $s_k.out_l^N$ 资源依赖于 s_i 的某个本地数据 $s_i.r_m$。

在上述定义中，验证读策略需要一系列条件，包括输出数据 $s_k.out_l^N$ 的安全标签 $L(s_k.out_l^N)$、被依赖的 s_i 的本地数据 $s_i.r_m$ 的安全标签 $L(s_i.r_m)$、s_j 的转换因子 $TF(s_j)$ 以及由 s_i 所在域颁发的属性证书 $dom(s_i).AC(s_j)$。本文用断言 $check(L(s_k.out_l^N), L(s_i.r_m), dom(s_i).AC(s_j), TF(s_j), RIFC(s_i))$ 来表示这个验证过程。

定义 12 云服务 s_i 的写策略集 $WIFC(s_i)$ 规定了是否允许 s_i 向本地写入满足下述条件的数据 $s_i.out_k^L$：1）$s_i.out_k^L \in s_i.Out^L$；2）$s_j.r_m \in Dep_R(s_i.out_k^L)$，其中，$s_j.r_m \in s_j.R$，即 $s_i.out_k^L$ 资源依赖于某个前趋服务 $s_j(s_j \in Pre(s_i))$ 的本地数据 $s_j.r_m$。

在上述定义中，验证写策略同样需要一系列条件，包括 s_i 输出数据 $s_i.out_k^L$ 的安全标签 $L(s_i.out_k^L)$、输出数据 $s_i.out_k^L$ 对应的本地数据的安全标签 $L(R(s_i.out_k^L))$、s_j 由 s_i 所在域颁发的属性证书 $dom(s_i).AC(s_j)$。本文用断言 $check(L(s_i.out_k^L), L(R(s_i.out_k^L)), dom(s_i).AC(s_j), WIFC(s_i))$ 来表示这个验证过程。

信息流策略的验证有 3 种可能结果：true 表示通过验证，false 表示未通过验证，unknown 则表示无法验证。只有验证结果为 true 时才允许信息流发生。

根据以上定义和分析，下文将给出云组合服务中每个服务需要实施的信息流控制机制。

6.2 信息流控制机制

由于云组合服务的动态性以及业务过程的复杂性，云服务输出数据的资源依赖是不可预知的，因此云服务无法根据信息流策略对每个后继服务进行验证。针对该问题，目前有两种解决方法：携带策略和反向检查[9]。携带策略指，在发送数据时将自身的安全策略一并发送，当接下来的操作需要验证时则由其后继服务完成。这种方法存在明显的缺陷：1）大大增加了网络开销；2）安全策略具有泄露的风险。反向检查指，当被发送的数据或存储的数据具有对前趋服务的资源依赖时，将对其产生依赖的资源所属的云服务请求策略验证。反向检查会使得服务间的信息交换变得复杂，但其具有更高的安全性。本文采用反向检查的方法进行云组合服务中的策略验证，并且利用安全中心来提高验证效率。

图 5 描述了安全中心的结构与功能。一个域的安全中心由属性机构、策略机构与策略决策点构成，不仅需要负责本域的属性管理，为云服务签发属性证书，而且需要为本域的云服务提供策略决策的功能。当需要反向检查，请求其他域的服务进行决策时，由策略机构向其他域的安全中心进行策略协商，帮助完成信息流策略的决策过程。

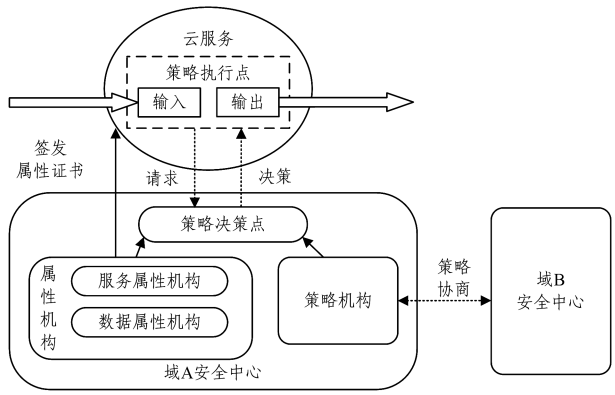

图 5 安全中心的结构与工作原理

Fig. 5 Structure and working principle of security authority

云组合服务中组件服务的信息流控制在以下 3 种情况下发生：1）从本地输入数据时，若该输入数据资源依赖于某些前趋服务，则使用反向检查来验证相关服务的读策略；2）向本地输出数据时，若输出数据资源依赖于某些前趋服务，则由云服务对这些服务进行写策略的验证；3）向后继服务输出数据时，若输出数据资源依赖于某些前趋服务，则使用反向检查来验证相关服务的读策略。其详细的流程描述如算法 3 所示。

算法 3 云组合服务中 s_i 中的信息流控制过程

1. s_i 收到来自直接前趋服务的服务请求，请求内容包括输入 $s_i.In^N$ 和每个输入数据 $s_i.in_j^N(s_i.in_j^N \in s_i.In^N)$ 的安全标签 $L(s_i.in_j^N)$ 和资源依赖 $Dep_R(s_i.in_j^N)$。
2. 执行带赖分析的业务计算过程 $s_i.F$ 得到 $s_i.Out$，以及每个输出数据 $s_i.out_k$ 的输入依赖 $Dep_{IN}(s_i.out_k)$；

   如果在计算中从本地输入数据 $s_i.in_j \in s_i.In^L$：

   对所有的 $r \in Dep_R(s_i.in_j)$，当 $r \in s_m.R(s_m \in Pre(s_i) \cup \{s_i\})$ 时，若 $check(L(s_i.in_j), L(r), dom(s_m).AC(s_i), TF(s_i), RIFC(s_m)) = true$，则验证通过，否则报告信息流违规。
3. 根据输入依赖 $Dep_{IN}(s_i.out_k)$ 计算所有输出数据 $s_i.out_k$ 的安全标签 $L(s_i.out_k)$ 与资源依赖 $Dep_R(s_i.out_k)$。
4. 对于 $s_i.Out$ 中的每个输出 $s_i.out_k$：

   4.1 如果输出数据 $s_i.out_k \in s_i.Out^L$：

   对于所有的 $r \in Dep_R(s_i.out_k)$，当 $r \in s_m.R(s_m \in Pre(s_i) \cup \{s_i\})$ 时，若 $check(L(s_i.out_k), L(R(s_i.out_k)), dom(s_i).AC(s_m), WIFC(s_i)) = true$，则验证通过，否则报告信息流违规。

   4.2 如果输出数据 $s_i.out_k \in s_i.Out^N$：

4.2.1 得到要发送的后继服务 $s_l(s_l \in s_i^+)$ 的属性证书以及转换因子 $TF(s_l)$；

4.2.2 对于所有的 $r \in Dep_R(s_i.out_k)$，当 $r \in s_m.R(s_m \in Pre(s_i) \cup \{s_i\})$ 时，若 $check(L(s_i.out_k), L(r), dom(s_m).AC(s_l), TF(s_l), RIFC(s_m)) = true$，则验证通过，否则报告信息流违规。

5. s_i 输出 $s_i.Out$，将 $s_i.Out^L$ 写入本地并向所有的后继服务 $s_l(s_l \in s_i^+)$ 发送服务请求，消息中包含输出数据 $s_i.out_k^N \in s_i.Out^N$，及其标签 $L(s_i.out_k^N)$ 与资源依赖 $Dep_R(s_i.out^N{}_k)$。

7　分析与评估

7.1　实例分析

本节通过引言中提到的例子来分析如何通过本文提出的机制来保护云组合服务中的信息流安全。假设 s_1，s_2 的读策略分别为 $RIFC(s_1)$，$RIFC(s_2)$，s_1 与 s_2 的转换因子均为空，d_1 的安全标签 $L(d_1)$ 中 $L_S(d_1)=\{l_1\}$，$L_I(d_1)=\{l_2\}$，d_2 的安全标签 $L(d_2)$ 中 $L_S(d_2)=\{l_3\}$，$L_I(d_2)=\{l_4\}$。

组合服务从 s_1 开始，如果 d_1 满足 $RIFC(s_1)$，云服务 s_1 使用本地数据 d_1 得到输出 o_1 且有 $Dep_{IN}(o_1)=\{d_1\}$，计算可得 $Dep_R(o_1)=\{d_1\}$，$L_S(o_1)=\{l_1\}$，$L_I(o_1)=\{l_2\}$。s_1 输出 o_1 时，通过 d_1，o_1 的安全标签与 s_2 的属性证书，检查 s_2，o_1 与 d_1 是否满足 $RIFC(s_1)$，若满足则将 o_1 发送给 s_2。

接着组合服务运行到 s_2，如果 d_2 满足 $RIFC(s_2)$，s_2 使用 o_1 与本地数据 d_2 得到输出 o_2 且有 $Dep_{IN}(o_2)=\{o_1,d_2\}$，计算可得 $Dep_R(o_2)=\{d_1,d_2\}$，$L_S(o_2)=\{l_1,l_3\}$，$L_I(o_1)=\varnothing$。当 s_2 输出 o_2 时，通过 o_2 和 d_2 的安全标签与服务 s_3 的属性证书，检查 s_3，o_2 与 d_2 是否满足 $RIFC(s_2)$；另外，s_2 向 s_1 发送请求，通过 o_2 和 d_1 的安全标签与服务 s_3 的属性证书，验证 s_3，o_2 与 d_1 是否满足 $RIFC(s_1)$。如果同时满足 s_1 与 s_2 的读策略，则将 o_2 输出至 s_3，否则不允许将 o_2 输出至 s_3。遵循云组合服务原有的策略：允许 s_1 和 s_2 读取 d_1 且允许 s_2 和 s_3 读取 d_2，但不允许 s_3 读取 d_1。在没有相应的信息流机制时，s_3 从 o_2 中得到了 d_1 的信息，这违反了 d_1 的访问策略。而在采用本文提出的机制之后，o_2 输出至 s_3 将不满足 s_1 的读策略，输出将被阻止，从而避免了敏感信息的泄露。

7.2　实验评估

本节最后对信息流机制进行实验，以评估在加入信息流机制后对组合服务性能的影响。本文实现了如图 6 所示的组合服务实例该实例包含顺序结构、选择结构（OR）、分支结构（AND）、循环结构（LOOP）等基本结构，s_0 是组合服务的发起者，s_8 是组合服务结果的接受者，实验中仅假设一个安全中心为所有服务的信息流决策提供判决。

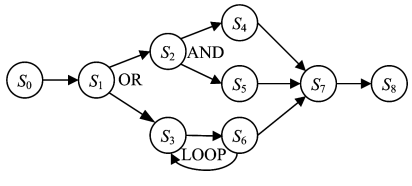

图 6　云组合服务实例

Fig. 6　Exmaple of cloud composition service

本文基于 Java 语言实现了相关的信息流组件，包括带标签的类与带依赖分析的函数，实验的系统环境为：Intel Core CPU 3.4 GHz，8 GB 内存，64 位 CentOS7 操作系统。编程环境为：Eclipse 4.5.0＋Java 1.8，服务器采用 Tomcat 8.0。实验分为两组，一组使用相关的信息流组件来实现功能（记作 DABIFC），另一组使用普通类与函数实现服务功能（记作 NIFC），并采用多种不同的数据输入来多次运行组合服务（只记录最终组合服务成功的输入），记录组合服务运行的时间，取每次输入运行 10 次后的平均值。我们发现依赖关系的复杂程度决定了信息流的控制复杂程度，因此本文精心设计了具有不同复杂程度的数据资源以供比较。用数值 $r=\frac{|LR|}{|LD|}$ 来表示这种复杂程度，其中，$|LR|$ 表示组合服务中所有资源的集合，$|LD|$ 表示所有资源在服务运行前的资源依赖数目之和。相关实验结果如图 7、图 8 所示。

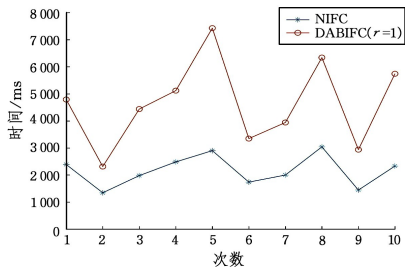

图 7　DABIFC 与 NIFC 的时间性能对比

Fig. 7　Performance comparison of time between DABIFC and NIFC

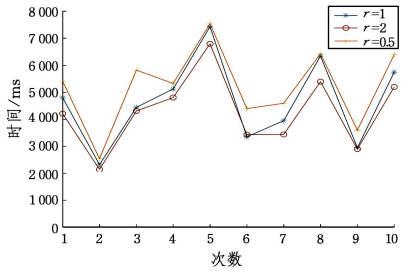

图 8　不同复杂程度的 DABIFC 时间性能对比

Fig. 8　Performance comparison of time among DABIFCs with different complexity

从实验结果可以看出，使用 DABIFC 将对性能产生很大影响，在完成同样服务的情况下，将消耗两倍左右的时间。而图 8 表示，如果参与计算的数据间依赖关系具有不同的复杂程度，那么性能也会出现差别，依赖关系越复杂（数据资源相对资源依赖的比值越低），性能损耗就越大，这与我们的预期是一致的。可以预测，如果系统内存在多安全中心进行策略协商，那么性能损耗将会更大；另外，信息流机制会阻止不合法的信息流使得服务中断，这会影响组合服务的性能。

结束语　本文提出了一种基于依赖分析的云组合服务信息流控制方法（DABIFC），该方法通过在云组合服务运行时分析各云服务输出数据的输入依赖与资源依赖，给出了输出数据安全标签的计算方法与组合信息流策略，并设计了信息流控制机制，保护了复杂组合结构下云组合服务中数据的机密性与完整性。

在云组合服务运行过程中，报告信息流违规将导致服务中断，影响系统的运行效率。在后续的工作中将进一步研究如何在云服务的组合阶段验证信息流的安全性，以减少系统运行时的错误开销。

参 考 文 献

[1] MENG S M. Trusted Service Composition and Its Key Technologies in Cloud Environment[D]. Nanjing: Nanjing University, 2016. (in Chinese)
孟顺梅. 云计算环境下可信服务组合及其关键技术研究[D]. 南京：南京大学，2016.

[2] JULA A, SUNDARARAJAN E, OTHMAN Z. Cloud computing service composition: A systematic literature review[J]. Expert Systems with Applications, 2014, 41(8): 3809-3824.

[3] XI N. A Study on Composable Information Flow Security Model and Approach[D]. Xi'an: Xidian University, 2014. (in Chinese)
习宁. 可组合信息流安全验证模型及方法研究[D]. 西安：西安电子科技大学，2014.

[4] YU B. Research on Key Security Techniques of Web Service Composition[D]. Changsha: National University of Defense Technology, 2013. (in Chinese)
喻波. Web 服务组合的关键安全技术研究[D]. 长沙：国防科学技术大学，2013.

[5] WANG Y D, YANG J H, XU C, et al. Survey on access control technologies for cloud computing[J]. Journal of Software, 2015, 26(5): 1129-1150. (in Chinese)
王于丁，杨家海，徐聪，等. 云计算访问控制技术研究综述[J]. 软件学报，2015，26(5)：1129-1150.

[6] BACON J, EYERS D, PASQUIER J M, et al. Information Flow Control for Secure Cloud Computing[J]. IEEE Transactions on Network & Service Management, 2014, 11(1): 76-89.

[7] SHE W, YEN I L, THURAISINGHAM B, et al. Security-Aware Service Composition with Fine-Grained Information Flow Control[J]. IEEE Transactions on Services Computing, 2013, 6(3): 330-343.

[8] HUTTER D, VOLKAMER M. Information Flow Control to Secure Dynamic Web Service Composition[J]. Lecture Notes in Computer Science, 2006, 3934: 196-210.

[9] SHE W, YEN I L, THURAISINGHAM B, et al. The SCIFC Model for Information Flow Control in Web Service Composition[C]// IEEE International Conference on Web Services. Los Angeles: IEEE, 2009: 1-8.

[10] SHE W, YEN I L, THURAISINGHAM B, et al. Rule-based run-time information flow control in service cloud[C]// 2011 IEEE International Conference on Web Services (ICWS). Washington, DC: IEEE, 2011: 524-531.

[11] YU B, YANG L, CHEN S, et al. An information flow control approach in composite services[C]// In IET International Conference on Information and Communications Technologies. Beijing: IET, 2013: 263-269.

[12] XI N, SUN C, MA J, et al. Secure service composition with information flow control in service clouds[J]. Future Generation Computer Systems, 2015, 49(C): 142-148.

[13] SOLANKI N, HOFFMAN T, YEN I L, et al. An Access and Information Flow Control Paradigm for Secure Information Sharing in Service-Based Systems[C]// 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC). Taichung: IEEE, 2015: 60-67.

[14] PASQUIER T, BACON J, SINGH J, et al. Data-Centric Access Control for Cloud Computing[C]// Symposium on Access Control Models and Technologies. Shanghai: ACM, 2016: 81-88.

[15] WANG L, LI F, LI L, et al. Principle and Practice of Taint Analysis[J]. Journal of Software, 2017, 28(4): 860-882. (in Chinese)
王蕾，李丰，李炼，等. 污点分析技术的原理和实践应用[J]. 软件学报，2017，28(4)：860-882.