doi:10.3969/j.issn.1000-436x.2014.11.003

基于模型检测的服务链信息流安全可组合验证方法

习宁，马建峰，孙聪，卢笛，张涛
（西安电子科技大学 计算机学院，陕西 西安 710071）

摘 要：提出了一种可组合的服务链信息流安全验证方法。在保证单一组件信息流安全的基础上，给出相邻组件可组合的信息流安全条件和验证算法。实验和仿真结果表明，相比传统模型检测方法，所提的可组合验证算法能够有效减小验证开销，提高验证效率。

关键词：模型检测；服务链；信息流安全；可组合
中图分类号：TP393.2 **文献标识码**：A **文章编号**：1000-436X(2014)11-0023-09

Composable information flow verification for service chain based on model checking

XI Ning, MA Jian-feng, SUN Cong, LU Di, ZHANG Tao
(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Abstract: A composable information verification approach is proposed for the secure service chain composition. Based on the secure service component, the security constraints for the component's composibility is specified and the information flow verification algorithms is proposed. Through the experiments and simulation, it shows that the approach can decrease the verification cost effectively and improve the efficiency of the verification.

Key words: model checking; service chain; information flow security; composibility

1 引言

随着面向服务的体系结构(SOA, service oriented architecture)[1]技术的快速发展，即时通信、移动定位、社交网络、电子商务等网络服务逐渐成为人们日常生活中不可或缺的一部分。不同服务之间的组合和协同是下一代互联网提供服务的一种新模式[2]。在该模式下，同一种服务可以由不同提供商提供，如定位服务，包括 Google Map、Foursquare、高德地图、百度地图等，用户可以根据自己的需求，如 QoS、安全强度等，选取合适的服务[3,4]。然而组合模式在极大丰富网络服务种类的同时，也带来了新的安全威胁。

在如今城市信息化、网络化、智能化的环境下，存在着各种敏感度不同的数据信息，如公共环境信息、个人位置信息等。各服务组件通过输入数据获取这些信息，并在不同服务间传输和使用。如果组合的方式不合理，将会导致在服务操作过程中机密数据的泄露，这将对公共安全，用户隐私带来极大的威胁。传统的访问控制方法可用于判定服务执行过程中对象是否有权限读取或写入数据[5,6]，但仅适用于对单一服务的本地资源，由于不同服务之间安全策略的异构性，当数据传输给其他服务进行处理时，无法进行有效的控制和管理。因此，如何准确分析并保证组合服务中的信息流安全是影响服务组合安全的关键问题之一。

近年来，物联网、云计算技术的研究不断深入，大数据时代已经到来，在服务提供过程中，

收稿日期：2013-10-29；**修回日期**：2014-01-25
基金项目：国家自然科学基金资助项目(U1135002, 61303033)；国家科技部重大专项基金资助项目(2011ZX03005-002)；航空科学基金资助项目(2013ZC31003)；陕西省自然科学基础研究计划基金资助项目(2013JQ8036)
Foundation Items: The National Natural Science Foundation of China (U1135002, 61303033); The Aviation Science Foundation of China (2013ZC31003, 20141931001); The Natural Science Basis Research Plan in Shaanxi Province of China (2013JQ8036)

服务需要使用并产生大量的数据，如环境感知信息、服务计算数据等[7]。本文针对服务提供过程中数据的信息流安全问题，提出了一种可组合的服务链信息流验证方法，具体内容组织如下：首先介绍了信息流安全研究的相关工作，分析了在多服务并存环境下现有研究的不足；然后引入面向服务的网络系统模型，并提出服务链信息流安全模型，随后重点描述了信息流验证框架和方法，并通过实验表明该方法的优越性，最后进行了总结和展望。

2 相关工作

目前，信息流验证方法主要包括了类型系统、程序切片以及模型检测 3 种方法。类型系统在偏序模型的基础上，通过对程序中操作对象赋予安全标记，结合标准编程语言的语法制定了信息流安全策略，并在程序编译过程中检测信息流安全性[8]。程序切片则通过分析程序依赖图(PDG, program dependence graph)获得不同对象间的依赖关系[9]，在此基础上结合偏序模型验证程序中的信息流安全性。模型检测首先对系统进行形式化建模，然后对信息流无干扰性(noninterference)采用线性时序逻辑(LTL, linear temporal logic)语言进行描述，最后通过模型验证工具进行自动化验证[10]。

针对组合服务中的信息流安全需求，如何准确分析和验证组合服务中的信息流是异构多域网络环境下实现不同服务间安全协同的关键问题。文献[11]基于类型系统对包含顺序、条件、循环以及过程调用等基本结构的服务调用语言制定了相应信息流安全策略，在语言编译阶段对组合服务中不同组件间的信息流安全进行验证。然而该类方法随着组合服务中不同组件的变化都需要进行重新编译，从而给服务组合带来了额外开销。

文献[12]通过构建服务组件 PDG 图，分析组件的输入输出的依赖关系，在此基础上结合偏序模型，给出了相邻组件安全组合条件，进而提出了分布式信息流安全验证方法。但该方法主要针对静态的程序结构，无法准确分析程序实际执行过程中的对象依赖关系。

文献[13]在 BPEL(business process execution language)中引入基于格的安全模型，然后通过模型检测是否存在信息泄露以及死锁等属性，文献[14]则进一步对文献[13]中的安全模型进行了扩展，支持更多类型的安全策略。模型检测针对程序的实际过程进行验证，因此在精确性方面优于基于 PDG 的静态分析方法[15]。

模型检测的方法[13,14]虽然能够保证验证的精确性，但其需要通过对组合服务整体建模来验证其信息流安全性，不同候选组件组合后的验证数量使系统的验证开销大大增加，一方面，针对不同组合服务中存在的相同候选服务，系统需要进行重复验证。另一方面，若单一组件本身过于复杂，组合服务的建模和验证过程将会消耗更多的时间和资源。

针对以上问题，本文提出了一种可组合的服务链信息流验证方法，首先通过模型检测的方法对单一组件进行验证，在此基础上通过分析组合服务中服务链中的信息流，给出组件间可组合的信息流安全条件和验证算法，从而避免了重复验证和组合后服务模型过于复杂的情况，能够有效减小验证开销，提高在多服务并存环境下服务安全组合的效率，保障用户快速、安全的使用组合服务。

3 服务链信息流安全模型

3.1 面向服务的网络系统模型

基于 SOA 架构构建的服务网络系统(SoNS, service-oriented network system)是一种大规模的异构分布式网络环境，如图 1 所示。在 SoNS 中，异构网络构成了不同的网络域，域中包含了各种不同的信息资源。域中的设备如计算机、移动终端、服务器等，获取各种资源并以服务的形式提供给网络中用户。每个网络域中还包含了安全中心来对不同域的安全策略进行管理，如信息的安全级别等。在不同网络域间存在第三方安全授权中心，负责协调和统一不同网络域中的安全级别，并辅助完成随后的信息流验证工作。

在面向服务的网络系统中，低层网络异构性由网络中间件屏蔽，节点功能统一由服务形式描述。

定义 1 节点服务 $\boldsymbol{s}$ 可表示为向量 $\boldsymbol{s}=<dom, In, Out, F, SCe>$，其中 $dom(\boldsymbol{s})\in D$ 表示服务节点所在网络域；In 表示服务的输入数据；Out 表示服务的输出数据；F 表示服务执行的操作序列 $<a_1, a_2, \cdots, a_n>$；SCe 表示服务的安全证书，记录服务的安全属性。

节点之间通过服务接口进行数据交互，实现服务间的相互协同，从而向用户提供功能更强的组合

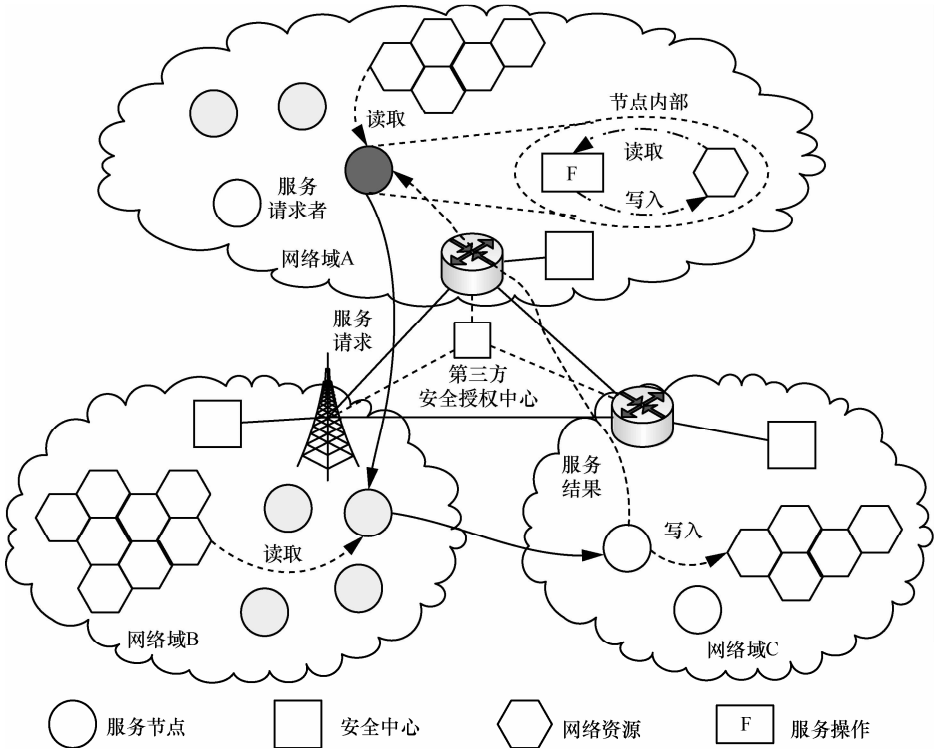


图 1　面向服务的网络系统模型

服务(CS, composite service)。服务链(service chain)是一种采用了顺序协同方式的组合服务。

定义 2　服务链 S_c 可表示为向量 $S_c =< CH, In_c, Out_c >$，其中 CH 表示服务的执行序列 $< s_0, s_1, \cdots, s_i, \cdots, s_n, s_{n+1} >$，$In_c$ 表示服务链的输入；Out_c 表示服务链的输出。

在服务链 S_c 中，每步服务 s_i 的输出将作为下一步服务 s_{i+1} 的输入，直到输出最终服务结果。每步服务 s_i 具有唯一的前驱 s_{i-1} 和后继 s_{i+1}。在此假设 s_0 表示提出服务请求的服务节点，s_{n+1} 表示接收到服务链 S_c 最终执行结果的节点。结合 SoNS 模型，s_0 和 s_{n+1} 对应同一网络节点，即服务请求者。相比具有复杂结构的组合服务，服务链流程控制简单，易于部署，在实际网络中得到了广泛使用。如文献[16]给出的公交车预计时间计算服务链实例。

3.2　多级安全模型

服务所涉及的数据种类不同（如环境、位置、医疗等），相应的数据安全敏感程度不同，描述数据安全关系的多级安全模型定义如下。

定义 3　多级安全模型表示为格 $(SL, \leqslant)$，$SL = \{sl_1, sl_2, \cdots\}$ 为数据的安全级别的集合，$\leqslant$ 表示 SL 上的偏序关系。

为了清晰说明，文中假设 $SL = \{L, H\}$ 分别表示高、低 2 种安全级别且满足 $L \leqslant H$。

此外，信息流安全属性分为机密性属性和完整性属性 2 类，机密性属性要求机密数据不能够流向公开对象中；完整性属性要求重要数据不能受到低安全信息源的影响。机密性和完整性是一对相对偶的信息流安全策略[17]，本文从机密性属性进行考虑。

3.3　组件信息流安全

服务链在执行过程中，单步的服务组件 s_i 根据服务请求对输入的数据进行处理并将结果输出到本地以及后继服务 s_{i+1}，如图 2 所示。

s_i 的输入数据 $In_i = In_i^M \cup In_i^D \cup In_i^L$，其中，$in_{i,x}^M \in In_i^M$ 表示 s_i 从前驱服务 s_{i-1} 中接收到的输入数据，即前一步服务的输出结果；$in_{i,x}^D \in In_i^D$ 表示 s_i 从网络域内收集到的数据，如环境及公共数据库中的信息；$in_{i,x}^L \in In_i^L$ 表示 s_i 从本地存储获取到的输入数据，即节点自身存储的信息。

s_i 的输出数据 $Out_i = Out_i^M \cup Out_i^D \cup Out_i^L$，其中，$Out_{i,x}^M \in Out_i^M$ 表示 s_i 执行后输出给后继服务 s_{i+1} 的数据，是下一步服务的输入；$Out_{i,x}^D \in Out_i^D$ 表示 s_i 在执行中更新网络域中数据资源，如公共数据库中

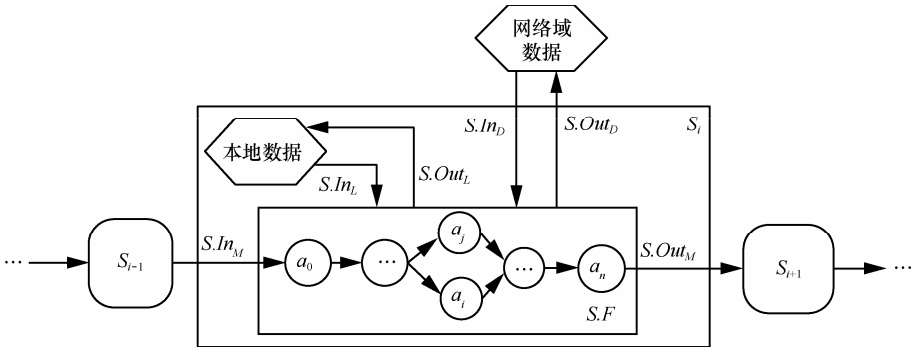


图2 服务组件 s_i 的信息流

的信息；$Out_{i,x}^L \in Out_i^L$ 表示 s_i 输出到节点本地存储中的数据。

无干扰性是保证软件信息流安全的根本属性，在组件信息流模型的基础上定义如下。

定义 4 服务 s_i 满足无干扰性当且仅当 $\forall u \in HIn_i$，$\forall v \in LOut_i$，u 的变化不会影响 v，记作 $u \nrightarrow v$。

其中，HIn_i，LIn_i 分别表示高、低安全级输入集合，即 $HIn_i = \{in_{i,x} | in_{i,x} \in In_i \wedge \mathrm{Sec}\ (in_{i,x}) = H\}$；$LIn_i = \{in_{i,x} | in_{i,x} \in In_i \wedge \mathrm{Sec}(in_{i,x}) = L\}$. $HOut_i$，$LOut_i$ 分别表示高、低安全级输出的集合，即 $HOut_i = \{out_{i,x} | out_{i,x} \in Out_i \wedge \mathrm{Sec}(out_{i,x}) = H\}$，$LOut_i = \{out_{i,x} | out_{i,x} \in Out_i \wedge \mathrm{Sec}(out_{i,x}) = L\}$，且 $In_i = HIn_i \cup LIn_i$，$Out_i = HOut_i \cup LOut_i$。函数 $\mathrm{Sec}::= In_i \cup Out_i \rightarrow sl$ 表示输入输出到安全级的映射。

3.4 服务链信息流安全

在服务链中，信息将在不同的服务组件中进行处理，单一服务组件信息流安全无法保证整个服务链信息流安全。例如，对于组件 s_i 来说，$\forall in \in In_i^M$ 可能由 s_{i-1} 或之前的服务组件计算得出；而 $\forall out \in Out_i^M$ 也可能交给 s_{i+1} 以及后续的服务进行处理，在这种情况下，定义服务链信息流无干扰性如下。

定义 5 服务链 S_c 满足无干扰性当且仅当 $\forall u \in HIn_c$，$\forall v \in LOut_c$，u 的变化不会影响 v，记作 $u \nrightarrow v$。其中，HIn_c 和 $LOut_c$ 分别表示服务链的高安全输入和低安全输出集合。

结合服务链定义，$In_c = \bigcup_{0 \leqslant i \leqslant n+1} In_i^D \cup In_i^L$，$Out_c = \bigcup_{0 \leqslant i \leqslant n+1} Out_i^D \cup Out_i^L$，且 $In_0^M = \phi$，$Out_0^D \cap Out_0^L = \phi$，$In_{n+1}^D \cap In_{n+1}^L = \phi$，$Out_{n+1}^M = \phi$，且 $\forall 0 \leqslant i \leqslant n$，$Out_i^M = In_{i+1}^M$。

引理 1 在服务链 S_c 中对 $\forall u \in In_i \cup Out_i$，$\forall v \in In_j \cup Out_j$，$0 \leqslant i < j$ 若 $u \rightsquigarrow v$，则 $\exists w_1 \in Out_{j-1}^M$，$w_2 \in In_j^M$，$u \rightsquigarrow w_1$，$w_1 \rightsquigarrow w_2$ 且 $w_2 \rightsquigarrow v$。

证明 假设当 $u \rightsquigarrow v$ 时，对 $\forall w_1 \in Out_{j-1}^M$，$w_2 \in In_j^M$，有 $u \nrightarrow w_1$ 或 $w_1 \nrightarrow w_2$ 或 $w_2 \nrightarrow v$。

1) $u \nrightarrow w_1$

根据定义3，Out_{j-1}^M 是服务 $s_i\ (0 \leqslant i < j)$ 将中间结果输出给 s_j 的唯一途径，若 $u \nrightarrow w_1$，则 $u \nrightarrow w_2$，又对 $\forall w_3 \in In_j^D$，$w_4 \in In_j^L$，有 $u \nrightarrow w_3$ 且 $u \nrightarrow w_4$。

故 $u \nrightarrow v$，与 $u \rightsquigarrow v$ 矛盾。

2) $u \rightsquigarrow w_1$ 且 $w_1 \nrightarrow w_2$

根据定义3，有 $Out_i^M = In_{i+1}^M$，则对 $\forall w_1 \in Out_{j-1}^M$，必 $\exists w_2 \in In_j^M$ 满足 $w_1 \rightsquigarrow w_2$，故与 $w_1 \nrightarrow w_2$ 矛盾。

3) $u \rightsquigarrow w_1$，$w_1 \rightsquigarrow w_2$ 且 $w_2 \nrightarrow v$

根据定义3，In_j^M 是服务 s_j 接收 $s_i\ (0 \leqslant i < j)$ 输出数据的唯一途径，若 $w_2 \nrightarrow v$，对 $\forall w_3 \in In_j^D$，$w_4 \in In_j^L$，有 $u \nrightarrow w_3$ 且 $u \nrightarrow w_4$。

可得 $u \nrightarrow v$，与条件 $u \rightsquigarrow v$ 矛盾。

引理 2 服务链 S_c 中前 m 组件满足无干扰性，且 $\forall w_1 \in Out_i^M$，$w_2 \in In_{j+1}^M$，$0 \leqslant i \leqslant m$，当 $w_1 \rightsquigarrow w_2$ 时满足 $\mathrm{Sec}(w_1) \leqslant \mathrm{Sec}(w_2)$，则对 $\forall u \in \bigcup_{0 \leqslant i \leqslant m} HIn_i$，$\forall v \in \bigcup_{i \leqslant j \leqslant m} LOut_j$，有 $u \nrightarrow v$。

证明 采用数学归纳法证明如下。

1) 当 $m=1$ 时，考虑服务组件 s_0 和 s_1。

①服务组件 s_0 满足无干扰性，由定义5可得出对 $\forall u \in HIn_0$，$\forall v \in LOut_0$ 有 $u \nrightarrow v$。

②服务组件 s_1 满足无干扰性，由定义5可得出对 $\forall u \in HIn_1$，$\forall v \in LOut_1$ 有 $u \nrightarrow v$。

以下证明对 $\forall u \in HIn_0$， $\forall v \in LOut_1$ 有 $u \nrightarrow v$，采用反证法证明如下：

假设 $\exists u \in HIn_0$， $\exists v \in LOut_1$ 有 $u \rightsquigarrow v$。根据引理 1，可得 $\exists w_1 \in Out_0^M$，$\exists w_2 \in In_1^M$，$u \rightsquigarrow w_1$，$w_1 \rightsquigarrow w_2$ 且 $w_2 \rightsquigarrow v$。

由于服务组件 s_0 是无干扰的，$u \in HIn_0$ 且 $u \rightsquigarrow w_1$，则 $w_1 \in HOut_0^M$。又 $\mathrm{Sec}(w_1) \leqslant \mathrm{Sec}(w_2)$，则 $w_2 \in HIn_1^M$，由于 $v \in LOut_1$，则 $w_2 \rightsquigarrow v$ 与服务组件 s_1 是无干扰的矛盾。

因此在引理 2 条件下对 $\forall u \in HIn_0$，$\forall v \in LOut_1$ 有 $u \nrightarrow v$。

2) 假设当 $m = n$ 时，引理 2 成立。

以下证明当 $m = n+1$ 时的情况。

① 因单个组件 s_{n+1} 满足无干扰性，由定义 5 可得，对 $\forall u \in HIn_{n+1}$，$\forall v \in LOut_{n+1}$ 有 $u \nrightarrow v$。

② 以下证明对 $\forall u \in HIn_i$，$\forall v \in LOut_{n+1}$，$0 \leqslant i < n+1$ 有 $u \nrightarrow v$，采用反证法。

假设 $\exists u \in HIn_i$，$\exists v \in LOut_{n+1}$，有 $u \rightsquigarrow v$。根据引理 1 可得：$\exists w_1 \in Out_n^M$，$\exists w_2 \in In_{n+1}^M$，$u \rightsquigarrow w_1$，$w_1 \rightsquigarrow w_2$ 且 $w_2 \rightsquigarrow v$。

由 $m = n$ 时假设可得，$\forall u \in \bigcup_{0 \leqslant i \leqslant n} HIn_i$ 且 $u \rightsquigarrow w_1$，则 $w_1 \in HOut_n^M$，又 $\mathrm{Sec}(w_1) \leqslant \mathrm{Sec}(w_2)$，则 $w_2 \in HIn_{n+1}^M$。

由于 $\forall v \in LOut_{n+1}$，则 $w_2 \rightsquigarrow v$ 与服务组件 s_{n+1} 是无干扰的矛盾。

因此在引理 2 条件下对 $\forall u \in HIn_i$，$\forall v \in LOut_{n+1}$，$0 \leqslant i < n+1$ 有 $u \nrightarrow v$。

综上所述，引理 2 得证。

定理 1　当服务链 S_c 服务组件 s_i，$0 \leqslant i < n+1$ 满足以下 2 个条件时，S_c 是信息流无干扰。

1) 服务组件 s_i 是信息流无干扰的。

2) 服务链中任意相邻 2 个服务 s_i 和 s_{i+1}，$0 \leqslant i \leqslant n$ 中，对 $\forall w_1 \in Out_i^M$，$w_2 \in In_{i+1}^M$，$w_1 \rightsquigarrow w_2$ 满足 $\mathrm{Sec}(w_1) \leqslant \mathrm{Sec}(w_2)$，也称 s_i 和 s_{i+1} 是可组合的。

证明　令 $m = n+1$，结合引理 2 和定义 5 直接得证。

4　基于模型检测的动态服务链信息流安全可组合验证方法

4.1　服务链信息流安全验证框架

在 SoNS 模型中，针对服务链 S_c 中的同一种服务 s_i，有不同的服务实现 $s_{i,j}$ 供用户选择，形成候选服务集合 $S_i = \{s_{i,j} \mid 0 \leqslant i \leqslant n+1, j \in N\}$。为保证服务链在执行过程中满足信息流安全性，服务请求者需要对候选服务组件内信息流及组合后组件间的可组合性分别进行验证。服务链信息流安全验证框架如图 3 所示。

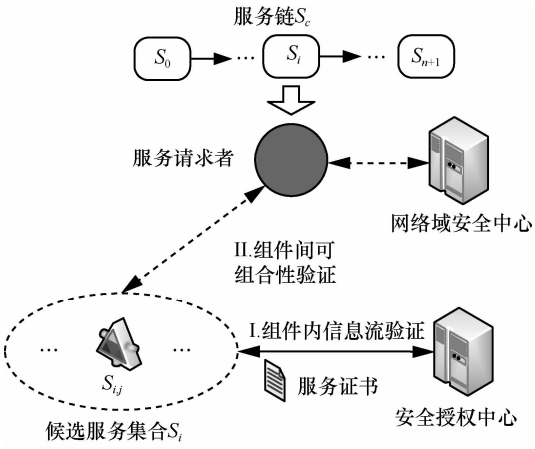


图 3　服务链信息流安全验证框架

安全验证框架包含了服务请求者、网络域安全中心、安全授权中心以及候选服务集合 4 种实体。结合定理 1，服务链信息流安全验证过程包括组件内信息流以及组件间可组合性验证 2 个部分。组件内验证由可信第三方安全授权中心完成；组件间由服务请求者结合服务链上下文来进行验证。

4.2　组件内信息流验证

针对服务执行过程中不同对象间的动态依赖关系，组件内信息流验证采用自合成模型检测[18]方法验证组件内信息流安全性，具体验证过程包括模型描述、模型自合成、无干扰性描述和模型验证 4 个阶段。

1) 模型描述

结合 SoNS 模型，采用标记迁移系统（LTS, labelled transition system）对 s_i 的功能实现 F_i 进行建模。F_i 表示服务 s_i 的执行功能，包含了程序语言中的基本操作和结构类型，即 *skip*、服务输入和输出、赋值操作；顺序、条件和循环结构，具体语法定义如图 4 所示。

$$
\begin{aligned}
&F_i ::= a; F_i' \\
&a ::= skip \mid input(in_{i,x}^y, \mathrm{var}) \mid output(out_{i,x}^y, \mathrm{var}) \\
&\quad \mid \mathrm{var} := e \mid a; a' \mid if(e)\ then\ a\ else\ a' \mid while(e)\ a \\
&\quad (y = M \mid D \mid L) \\
&e ::= \mathrm{var} \mid e\,\mathrm{Re} \\
&R ::= + \mid - \mid \times \mid \div \mid < \mid > \mid \leqslant \mid \geqslant
\end{aligned}
$$

图 4　服务功能 F_i 语法

在服务功能语法定义的基础上，F_i 执行状态 μ 由输入映射 I，程序变量映射 V 和输出映射 O 组成，记作 $\mu ::=(I, V, O)$。其中，$I ::=In_i \to Val$，$V ::=Var_i \to Val$，$O ::=Out_i \to Val$；Val 为数据值集。F_i 执行初态 μ_0 和终态 μ_e 分别由输入值集和输出值集唯一表示。

s_i 的功能实现 F_i 可抽象为标记迁移系统 $\mathcal{M}=(\mu,\to)$，其中，状态集合 μ 为 F_i 执行状态抽象的集合；迁移集合 $\to$ 为 μ 上的二元关系。由服务实现到标记迁移系统的转换规则 $\Phi(a,n_i,n_j,\to)$ 定义如图 5 所示，n_i 和 n_j 分别表示操作在 F_i 中的进入和退出点，$\to$ 表示变迁关系。

$$\Phi(skip,n_i,n_j,\to)=\{<n_i>\to<n_j>\mid I'=I\wedge O'=O\wedge V'=V\}$$

$$\Phi(input(\mathrm{in}^{y}_{i,x},\mathrm{var}),n_i,n_j,\to)=\{<n_i>\to<n_j>\mid I'=I\wedge O'=O\wedge V'(\mathrm{var})=I(\mathrm{in}^{y}_{i,x})\wedge(\forall\,\mathrm{var}'\neq\mathrm{var},\ V'(\mathrm{var}')=V(\mathrm{var}'))\}$$

$$\Phi(output(\mathrm{out}^{y}_{i,x},\mathrm{var}),n_i,n_j,\to)=\{<n_i>\to<n_j>\mid I'=I\wedge O'(\mathrm{out}^{y}_{i,x})=V(\mathrm{var})\wedge V'=V\}\quad(\mathrm{y}=\mathrm{M}\mid\mathrm{D}\mid\mathrm{L})$$

$$\Phi(\mathrm{var}:=\mathrm{e},n_i,n_j,\to)=\{<n_i>\to<n_j>\mid I'=I\wedge O'=O\wedge V'(\mathrm{var})=I(\mathrm{e})\wedge(\forall\,\mathrm{var}'\neq\mathrm{var},\ V'(\mathrm{var}')=V(\mathrm{var}'))\}$$

$$\Phi(\mathrm{a};\mathrm{a}',n_i,n_j,\to)=\Phi(\mathrm{a},n_i,n_k,\to)\cup\Phi(\mathrm{a}',n_k,n_j,\to)$$

$$\Phi(if(\mathrm{e})\ then\ \mathrm{a}\ else\ \mathrm{a}',n_i,n_j,\to)=\{<n_i>\to<n_k>\mid I'=I\wedge O'=O\wedge V'=V\wedge\mathrm{e}\}\cup\{<n_i>\to<n_j>\mid I'=I\wedge O'=O\wedge V'=V\wedge\neg\mathrm{e}\}\cup\Phi(\mathrm{a},n_k,n_j,\to)\cup\Phi(\mathrm{a}',n_i,n_j,\to)$$

$$\Phi(while(\mathrm{e})\ \mathrm{a},n_i,n_j,\to)=\{<n_i>\to<n_k>\mid I'=I\wedge O'=O\wedge V'=V\wedge\mathrm{e}\}\cup\{<n_i>\to<n_j>\mid I'=I\wedge O'=O\wedge V'=V\wedge\neg\mathrm{e}\}\cup\Phi(\mathrm{a},n_k,n_i,\to)$$

图 5 迁移系统建模规则

2) 模型自合成

为了能够有效验证定理 1 的安全条件，针对具有初态 μ_0 的组件模型 $\mathcal{M}$，采用自合成方法[18]进行模型变换如下。

① 复制输入模型 $\mathcal{M}=(\mu,\to)$，生成 $\mathcal{M}'=(\mu',\to)$。

② 对 $\mathcal{M}$、$\mathcal{M}'$ 初始状态下的低安全级输入 $li_{i,x}\in LIn_i$ 之间进行交叉赋值，$I_{\mu_0'}(li_{i,x}):=I_{\mu_0}(li_{i,x})$，用以保证任一初态下 $\mathcal{M}$ 和 $\mathcal{M}'$ 执行前的状态满足低安全级等价。

3) 无干扰性描述

自合成后模型 $\mathcal{M}$ 和 $\mathcal{M}'$ 的执行后终态分别为 $\mu_e=F_i(\mu_0)$，$\mu_e'=F_i(\mu_0')$；结合文献[18]，信息流无干扰性的断言表示如下

$$\bigwedge_{0\leqslant x\leqslant|LOut_i|} assert(O_{\mu_e}(lo_{i,x})==O_{\mu_e'}(lo_{i,x}))$$

4) 模型验证

将安全属性以断言形式加入输入模型后，由模型检测工具自动验证并输出验证结果。若断言满足则可说明输入的服务组件模型满足无干扰性定义，是信息流安全的；若断言判定失败，则说明服务组件模型不满足无干扰性定义，模型检测工具将给出具体的反例。组件验证通过后，结果以证书形式由安全授权中心发布给服务，用于组合过程中的信息流安全验证。

组件内信息流验证采用离线方式在服务组件部署时进行验证，当不同组件开始组合时只需进行组件间可组合性验证即可，从而减小验证时延，提高组合效率。

4.3 组件间可组合性验证

在 SoNS 模型中，服务链 S_c 由不同的候选服务组件 $s_{i,j}\in S_i$ 动态组合而成。根据所选定组件的不同，组合后的组件中对象的依赖关系也不尽相同。为了保证服务链中信息流无干扰性，系统首先请求相邻组件 s_i 和 s_{i+1} 的证书，并验证其有效性，然后根据定理 1 对其可组合性进行验证。具体验证算法流程如下。

算法 1 Composability_Verification( )

输入：相邻服务 s_i, s_i+1

输出：True 或 False

1) \\Require_Cert(s_i, s_i+1)表示验证系统向 s_i 和 s_i+1 请求并验证证书

2) if Require_Cert(s_i, s_i+1)==False then

3) Return False

4) end if

5) for each $Out_{i,x}^{M}\in In_{i+1,x}^{M}\in In_{i+1}^{M}$ do

6) if $\mathrm{Sec}(Out_{i,x})\leqslant Sex(in_{i+1,x})$ then

7) Return False

8) end if

9) end for

10) Return True

不难看出，组件间验证算法复杂度为 $O(n)$，其中，n 表示服务 s_i 输出 Out_i^M 的个数。

4.4 服务链信息流安全验证

在组件内和组件间验证算法的基础上，本文采用图论思想对整体的安全服务链验证算法进行了设计，分为 2 个阶段：有向图构造阶段和路径查找阶段。在有向图构造阶段，首先将所有候选服务作为有向图节点，然后对相邻服务间调用组件间验证算法，若验证通过，则在该对节点间构造有向边，若不合法，则不构造，从而过滤掉不合法的组合。

最后采用路径查找算法寻找从 s_0 至 s_{n+1} 所有路径 P_{ex} 即为安全的服务可执行路径。不难看出，构造阶段的复杂度为 $O(n^3)$，其中 n 表示候选服务的平均数。具体算法流程如下。

算法 2　Service_Chain_Verification ( )

输入：候选服务序列 $S_0,S_1,\cdots,S_{n+1}$

输出：其他的服务可执行路径 P_{ex}

1) $\backslash\backslash G\langle V,\vec{E}\rangle$ 表示候选服务构成的有向组合服务图

2) \\push_edge_into(G, v_1,v_2)表示在图 G 中添加从 v_1 到 v_2 的有向路径

3) \\FindPath(G, v_1,v_2)表示在有向图 G 中寻找从 v_1 节点到 v_2 节点的所有路径

4) for $0\leqslant i\leqslant n$　do

5)　　for each $s_{i,j}\in S_i$ do

6)　　for each $s_{i+1,k}\in S_{i+1}$ do

7)　　if Compossability_Verfication ($s_{i,j}$, $s_{i+1,k}$==True) then

8)　　push_edge_into($G.\vec{E}$, $s_{i,j}$, $s_{i+1,k}$)

9)　　end if

10)　　end for

11)　　end for

12) end for

13) P_{ex}=*FindPath*(G, s_0, S_{n+1})

14) Return P_{ex}

5　仿真与实验

在安全性证明的基础上，本文采用 spin 模型检测工具[19]和 NS-3 网络仿真工具[20]对信息流可组合安全验证框架的性能进行测试和分析。结合第 3 部分组合服务实例，实验分别对单一采用模型检测的验证方法以及组合验证方法的性能进行测试。

1) 单一模型检测方法

本文对不同服务步骤中常用的定位和测速算法输入 spin 工具进行检测。测试算法包括：①定位服务的 3 种算法：卫星定位（GPS, global positioning system，全球定位系统），到达时间差（TDOA, time difference of arrival），Wi-Fi 无线定位；②终端移动速度计算服务的 3 种算法：基于电平通过率（LCR，level crossing rate），零通过率（ZCR，zero crossing rate），协方差（COV，covariance）；③预计时间服务 ETA（estimation of time algorithm）。验证时延测试结果如图 6 所示。

由图 6 可以看出，验证时延与算法的复杂度相关，随着代码量的增加，验证状态增多，验证时延增大。当不同服务组合成服务链后，单步服务可能由不同服务提供商执行，根据组合后不同的执行路径，验证状态进一步增多。结合具体实验环境，组合后服务验证代价如图 7 所示。

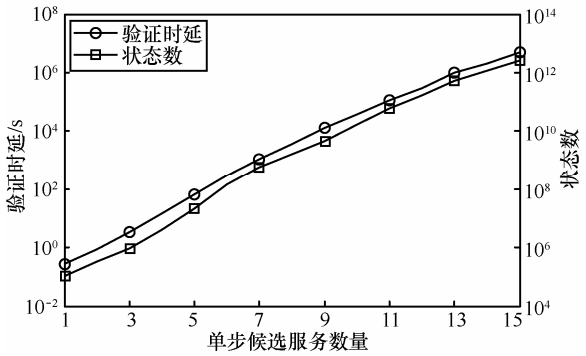


图 7　组合服务验证开销

由图 7 看出，随着状态数的增加，状态间的转移关系更加复杂，加快了验证时延的增长速率。当

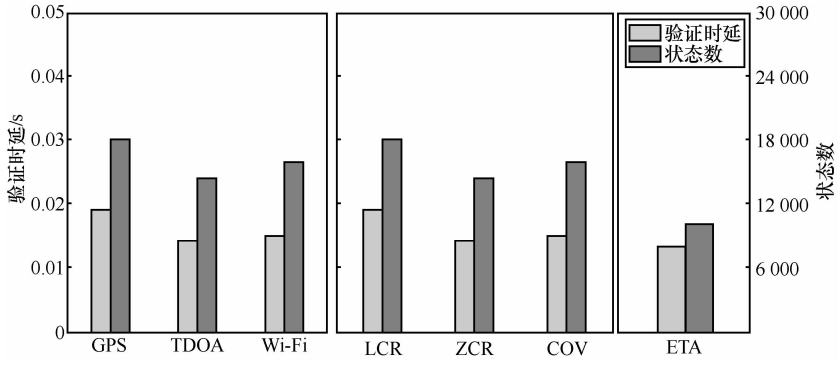


图 6　不同服务算法验证开销

候选服务数量超过 15 时，验证工具状态溢出，无法继续验证。由此可见，在大量候选服务并存的 SoNS 模型中，由用户采用模型检测的方法来验证每条可行的服务执行路径，验证开销太大，大大影响用户后续使用服务的体验。

2) 基于模型检测的可组合验证方法

在可组合验证框架中，验证开销包括组件内和组件间 2 部分。对于组件内验证过程，采用 spin 模型检测工具对单个服务组件逐一测试，统计组件内验证时延；对于组件间验证过程，采用 NS-3 网络仿真工具，统计组件间时延。测试结果如图 8 所示。

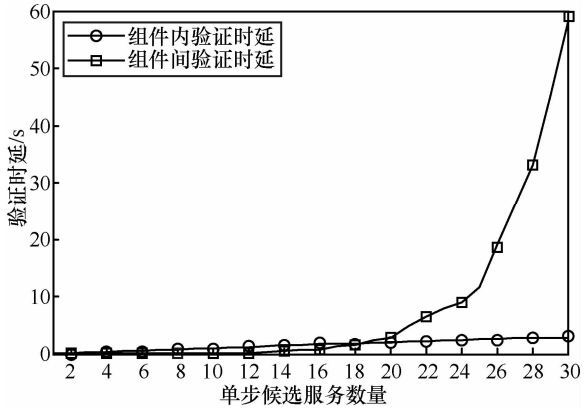


图 8　可组合方法验证开销

在图 8 中，当候选服务数量较小时，验证开销主要集中在组件内部；随着候选数量的增大，组件内验证开销平稳增加，而组件间由于复杂化的组合关系，验证开销呈指数级增大，当到达 20 以后，开销则主要集中在组件间验证过程。

3) 方法对比

结合以上实验，在不同候选服务数量下，2 种方法的验证时延对比如图 9 所示。

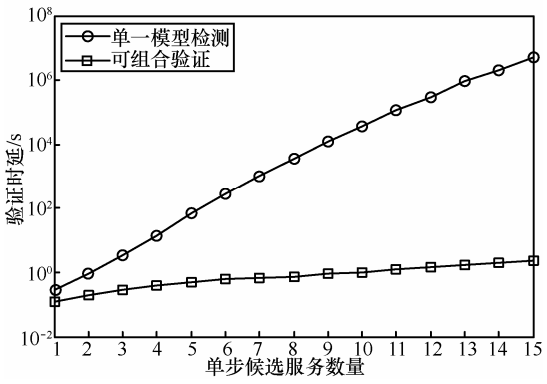


图 9　2 种方法的验证时延

单一的基于模型检测方法可应用在服务流程确定的静态组合服务中；在 SoNS 模型中，不同候选服务组件将组合成不同的服务流程，直接采用模型检测方法验证，需要将每条可能的执行路径进行建模并输入验证，不同路径中相同服务组件的重复验证给系统带来额外开销，而可组合验证方法采用模型检测验证在单一组件安全的基础上，避免了重复验证，提高了验证效率，能够在多服务并存环境下，更加快速安全地为用户提供组合服务。

6　结束语

针对传统模型检测方法组合服务建模复杂，信息流安全验证开销大的缺陷，本文提出了一种可组合的信息流验证方法。该方法首先对单一组件内的信息流采用模型检测的方法进行建模验证，然后通过分析相邻组件间的数据的依赖关系，给出安全可组合条件并设计了相应的验证算法。通过实验和仿真可以看出，可组合验证方法能够简化模型检测建模过程，避免重复验证组件的开销，提高验证效率。采用顺序组合的服务链是一种简化的组合服务，对包含条件、循环等结构的组合服务中的可组合性的验证将会做进一步研究。

参考文献：

[1] ENDREI M, ANG J, ARSANJANI A, *et al*. Patterns: Service Oriented Architecture and Web Services[S]. IBM International Technical Support Organization, 2004.

[2] JOANNA N. The Personal Web: smart internet for me[A]. Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research[C]. 2010.

[3] YU T, ZHANG Y, LIN K J. Efficient algorithms for Web services selection with end-to-end QoS constraints[J]. ACM Transactions on the Web (TWEB), 2007,1(1): 103-126.

[4] 龙军, 袁鑫攀, 桂卫华. 基于环境感知的可信 QoS 评价与服务选取策略[J]. 电子学报, 2012,40(6): 1133-1140.
LONG J,YUAN X P,GUO W H. A policy for the trusted QoS evaluation and service selection with environment aware[J]. Acta Electronica Sinica, 2012,40(6):1133-1140.

[5] BERTINO E, SQUICCIARINI A C, MEVI D. A fine-grained access control model for Web services[A]. SCC[C]. 2004.33-40.

[6] BHATTI R, BERTINO E, GHAFOOR A. Trust-based context-aware

access control model for Web-services[A]. ICWS[C]. 2004.184-191.

[7] ZHENG Z B, ZHU J M, LYU M R. Service-generated big data and big data-as-a-service: an overview[A]. 2013 IEEE International Congress on Big Data[C]. 2013.403-410.

[8] VOLPANO D, IRVINE C, SMITH G. A sound type system for secure flow analysis[J]. Journal of Computer Security, 1996,4(2):167-187.

[9] FERRANTE J, OTTENSTEIN K J, WARREN J D. The program dependence graph and its use in optimization[J]. ACM Transactions on Programming Languages and Systems, 1987, 9(3): 319-349.

[10] DIMITROVA R, FINKBEINER B, KOVÁCS M, *et al*. Model Checking Information Flow in Reactive Systems. Verification, Model Checking, and Abstract Interpretation[M]. Springer Berlin Heidelberg, 2012. 169-185.

[11] HUTTER D, VOLKAMER M. Information flow control to secure dynamic Web service composition[A]. Security in Pervasive Computing[C]. 2006. 196-210.

[12] XI N, MA J F, SUN C, *et al*. Distributed information flow verification framework for the composition of service chain in wireless sensor network[EB/OL].http://dx.doi.org/10.1155/2013/693639.

[13] NAKAJIMA S. Model-checking of safety and security aspects in Web service flows[A]. ICWE 2004[C].2004.488-501.

[14] SABINA R L S. Model checking adaptive multilevel service compositions[A]. FACS 2010[C]. 2012. 106-124.

[15] SUN C, ZHAI E, CHEN Z, *et al*. A multi-compositional enforcement on information flow security[J]. Information and Communications Security Lecture Notes in Computer Science, 2011, 7043: 345-359.

[16] GROBA C, CLARKE S. Opportunistic composition of sequentially-connected services in mobile computing environments[A]. ICWS[C]. 2011.17-24.

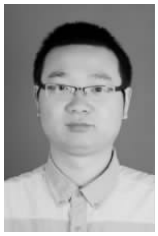
[17] BIBA K J. Integrity Considerations for Secure Computer Systems[R]. Electronic Systems Division, Bedford, MA, 1977.

[18] BARTHE G, D'ARGENIO P R, RESK T. Secure information flow by self-composition[A]. Proc of the 17th Computer Security Foundations Workshop[C]. 2004.100-114.

[19] Open Source Project. Spin verification tool[EB/OL]. http://www.spinroot. com/spin.

[20] Open Source Project. NS-3 project[EB/OL]. http://www.nsnam.org/.

作者简介：



习宁（1986-），男，陕西渭南人，博士，西安电子科技大学讲师，主要研究方向为服务计算、信息流安全。

马建峰（1963-），男，陕西西安人，博士，西安电子科技大学教授、博士生导师，主要研究方向为密码学、无线和移动安全、系统可生存性等。

孙聪（1982-），男，陕西兴平人，博士，西安电子科技大学副教授，主要研究方向为信息流安全、程序分析与验证。

卢笛（1983-），男，陕西安康人，西安电子科技大学博士生，主要研究方向为虚拟化技术、存储系统、操作系统安全、虚拟化资源调度。

张涛（1986-），男，陕西西安人，西安电子科技大学博士生，主要研究方向为服务计算、可信计算、信息安全、社交网络。