第43卷 第12期 西 安 交 通 大 学 学 报 Vol.43 №12
2009年12月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Dec. 2009

面向变异分析的协议安全测试方法

章志燮[1,2,3]，周颢[1,2,3]，赵保华[1,2,3]
(1.中国科学技术大学计算机科学与技术系，230027，合肥；2.网络与交换技术国家重点实验室，100876，北京；
3.安徽省计算与通讯软件重点实验室，230027，合肥)

摘要：在基于构造类别代数的协议描述上引入变异分析方法，由此提出了一种基于错误模型的协议安全测试方法.通过设计针对构造类别代数的变异算子，限制了协议中的错误集合；应用变异算子生成变异体集合，并消除其中的等价变异体；基于变异体构造安全测试例.同比研究表明，采用基于错误模型的变异分析方法，可以有效解决协议安全测试中忽视协议数据流处理过程、错误集合无限和缺少结果判断机制等问题，限定协议可能存在的错误集合，有利于测试的量化和评估，能够更有针对性地进行测试例构造和测试结果判断，提高测试能力.

关键词：协议安全测试；构造类别代数；变异分析
中图分类号：TP393.06 **文献标志码**：A **文章编号**：0253-987X(2009)12-0011-05

A Protocol Security Testing Method Using Mutation Analysis

ZHANG Zhixie[1,2,3], ZHOU Hao[1,2,3], ZHAO Baohua[1,2,3]
(1. Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China;
2. State Key Laboratory of Networking and Switching Technology, Beijing 100876, China; 3. Anhui Province
Key Laboratory of Software in Computing and Communication, Hefei 230027, China)

Abstract: A new protocol security testing method is proposed based on a fault model. The method utilizes the mutation analysis based on the specification of constructed type algebra. Mutant operators are designed to restrict the possible fault sets in protocol; then mutants are generated and equivalent mutants are deleted; and test cases are finally constructed based on resulting mutants. Compared with existing methods, the proposed method can effectively solve several problems in current protocol security testing, such as neglecting the protocol dataflow, infinite fault sets, the lack of the mechanism to judge results, and so on. It is beneficial to the quantification and appraisal of testing to limit the possible fault sets of protocol. The method can pointedly be used to construct test cases and to judge test results, and improves testing capacity.

Keywords: protocol security testing; constructed type algebra; mutation analysis

协议是网络交互行为的主体，协议的正确性是保证网络正常运行的基础，然而协议在实现过程中不可避免地会引入错误.传统协议测试方法，如一致性测试等，主要关注协议功能，保证协议在正常数据交互时的正确性.在实际网络中，通常存在大量特定或异常协议报文，这些数据会引发协议出错，导致网络异常甚至崩溃，因此有必要进行协议安全测试.

目前，协议安全测试研究主要集中在如何构造异常网络数据报文，语法分析、错误注入[1-4]等技术未考虑协议数据流的处理过程，而是通过手动或随机生成异常数据报文注入待测协议，观察待测协议是否崩溃来判断协议实现的错误.协议实现中的错误集合是无穷的，现有方法与测试者的经验直接相关，因此无法保证测试的效率和覆盖率.

收稿日期：2009-03-17. 作者简介：章志燮(1982—)，男，博士生；赵保华(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(60872009，60602016)；国家高技术研究发展计划资助项目(2007AA01Z428，2009AA01Z148)；安徽省自然科学研究计划重大项目(ZD2008005-2，ZD200904，JK2009A013，JK2009A025).

协议形式化描述包括数据流和控制流描述，安全测试主要是针对数据流的. 现有形式化模型，如有限状态机等，多用于协议控制流描述，但对数据流描述的能力有限. 构造类别代数[5-6]是一种适用于协议数据流描述的形式化方法，文献[5]证明了其公理系统的一致性与完备性. 变异分析[7]是一种基于错误模型的测试方法，基本思想是：在正确的代码或规范中通过引入变异算子的微小语法变化，生成不同于原实体的语法正确、语义不同的变异体. 如果存在一个测试例，其在变异体和正常体上产生不同的执行结果，则称此变异体是可区分的，否则称为等价变异体. 变异分析实现简单，易于量化和评估，通过设计变异算子可以很好地限制协议的错误集合，基于错误模型可以更有针对性地生成测试例，判断测试结果.

本文在基于构造类别代数的协议描述上引入变异分析方法，生成变异体集合，基于变异体构造安全测试例，完成测试过程.

1 构造类别代数

构造类别代数[5]为一个四元组 $D=\langle S,C,E,Q\rangle$，其中 S 为类别名集合，C 和 E 分别是构造函数和延拓函数的有限集合，Q 是满足 C 和 E 的公理集合.

根据函数是否同用户直接交互，可分为可控制函数、可观察函数和内部函数3种：可控制函数是用户可以通过改变输入信息来影响协议实体的函数；可观察函数是产生输出信息的函数，用户通过接收并分析输出信息判断协议实现是否满足公理语义；内部函数不直接与用户交互，只能由其他函数间接调用. 在协议测试过程中，必须通过可控制函数对协议实现进行输入激励，间接或直接地调用可观察函数，以生成输出信息.

公理为一个四元组 $a=\langle p_{\mathrm{pre}},p_{\mathrm{post}},p_{\mathrm{pri}},e_{\mathrm{equ}}\rangle$，$a\in Q$，其中：$p_{\mathrm{pre}}$ 为前条件，是输入参数、内部变量、常数和函数调用构造的谓词；p_{post} 为后条件，是结果变量与输入参数、内部变量之间的某种关系；p_{pri} 为公理优先级，是非负整数，数值越大则优先级越低，如对于公理 a 和 b，若有 $p^a_{\mathrm{pri}}<p^b_{\mathrm{pri}}$，表明 a 的优先级高于 b；e_{equ} 为等式公理. 在函数调用时，按照公理优先级从高到低依次匹配公理的 p_{pre}，如果当前输入参数和环境变量满足 p_{pre}，则匹配等式 e_{equ} 的右边并返回执行公理 p_{post}，不再匹配低优先级的公理. 因此，低优先级公理的 p_{pre} 隐含了所有高优先级公理前条件均不匹配的因素.

2 变异分析

2.1 变异算子设计

基于构造类别代数，设计变异算子如下.

(1)类别变异算子. 构造类别代数中定义的类别，类似于程序设计语言中的数据类型，需指定数据长度和取值范围. 该变异算子可通过类别替换改变规范中的变量及函数的定义. 例如，将 unsigned int 型的参数替换为 int 型变量，会导致错误的数据取值范围，而在描述报文的构造函数定义中，将 short 型的参数替换为 int 型，可定义结构异常的协议报文.

(2)函数变异算子. 此类算子主要针对函数定义的外部接口，包括增加或删除参数，改变参数顺序等.

构造类别代数中的函数包括构造函数 C 和延拓函数 E：C 定义协议报文结构，其变异影响数据报文格式的处理；E 的参数规定了其对外部环境的需求，在变异过程中主要影响报文字段的处理.

(3)公理变异算子. 公理的 p_{pre} 可采用析取范式(DNF)描述，表示为形如 $(p_1\wedge p_2\wedge\cdots)\vee(\cdots\wedge p_{n-1}\wedge p_n)$ 的析取表达式，其中 $p_i(i\in 1,\cdots,n)$ 为具有布尔结果的条件表达式，可以是一个布尔变量，或由关系运算符 $\{=,\neq,<,\leqslant,>,\geqslant\}$ 连接的一对代数表达式，或返回值为布尔型的函数调用，可将 p_i 称为公理的子条件. 将公理 p_{pre} 中形如 $(p_1\wedge p_2\wedge\cdots)$ 的单个析取表达式称为公理的判定条件，代表数据流某个分支的条件判断或数据处理. 公理的 p_{post} 为一系列赋值语句的合取表达式. 在数据流分析中，公理的不同优先级规定了数据流处理的先后顺序，而相同优先级的不同公理则代表数据流的不同分支. 针对公理的变异算子设计如下.

(1)简单变异算子. 此类算子主要针对公理的前、后条件中使用到的简单元素，包括变量、常量、关系运算符等. 变量和常量都有其类别所限定的取值范围，可以用该类别的特定值、边界值或特殊的表达式来替代. 关系运算符的变异主要是关系运算符 $\{=,\neq,<,\leqslant,>,\geqslant\}$ 之间的替代，如：整型变量 x 可替换为特殊值0、1、−1或特殊表达式 $x+1$ 等；布尔型变量 y 可以替换为 true、false 或 y 自身取反.

(2)函数调用变异算子. 在公理中若涉及到对其他函数的调用，可将此调用过程视为整体，再根据其返回值类型进行替换，或使用具有相同返回值类型

的其他函数调用进行替换.

(3)子条件变异算子. 此类算子是将公理前条件中的子条件视为整体，采用布尔值或其他子条件进行替换. 该算子的变异会影响数据流分支的判断.

(4)判定条件变异算子. 此类算子主要针对公理的前条件，并将判定条件视为整体. 该算子的变异同样会影响数据流分支的判断.

(5)公理优先级变异算子. 此类算子将公理视为一个整体，通过改变公理的优先级来影响公理匹配的顺序.

公理集合是描述协议数据流的主体，应用变异算子得到的变异体表现为变异公理集合. 变异算子应用于协议描述中的不同位置时，所得到的变异公理集合有所不同. 上述各变异算子的应用位置及影响公理集合的范围如下.

(1)类别变异算子适用于全局描述、函数定义和公理. 全局描述的变异会影响所有使用该类别对象的函数及其公理集合；函数定义的变异会影响与该函数相关的所有公理；公理变异会影响公理中使用的数据类型.

(2)针对函数的变异算子适用于函数定义，影响该函数的公理集合.

(3)针对公理的变异算子，其子条件和判定条件变异算子、公理优先级变异算子和简单变异中的运算符变异算子只能针对单个公理；简单变异中的变量和常量变异算子及函数调用变异算子适用于单个公理和同优先级的多个公理，可以表示数据流从一个分支偏移到另一个分支的协议错误.

2.2　等价变异体产生原因分析

在基于构造类别代数的协议描述上，应用变异算子会产生大量的等价变异体，这将影响变异分析方法的效率. 分析变异算子应用及公理集合匹配的过程，可得到如下几种产生等价变异体的原因.

(1)违反高优先级公理的约束条件. 公理匹配过程是按照优先级从高到低的顺序进行的，只有在高优先级公理的前条件匹配失败时，才会进行低优先级公理的前条件比较，因此每个公理的前条件都隐含了所有高优先级公理均匹配不成功的因素，所以对输入的报文数据存在约束条件. 例如，对于公理 a 和 b，有 $p^a_{\mathrm{pri}}<p^b_{\mathrm{pri}}$，若 x 为变量，有 $p^a_{\mathrm{pre}}:x<3$，$p^b_{\mathrm{pre}}:x=3$，则 p^b_{pre} 匹配成功的前提隐含了 $x\geqslant 3$ 的限定条件. 如果对 p^b_{pre} 应用变异算子 $(x\rightarrow x+1)$，便得到变异后的前条件为 $x=2$，而在 x 取值为 2 时，将匹配公理 a 返回，则 p^b_{pre} 无法到达公理 b.

(2)违反公理自身的约束. 在对公理施加变异算子的过程中，会出现公理自身行为不一致的情况，尤其是变异过程仅影响公理中的子条件或单个判定条件. 例如，公理 a 有前条件 $p^a_{\mathrm{pre}}:p_1\vee p_2$，其中 p_1 和 p_2 为判定条件，对 p_1 施加变异算子便得到 p'_1，如果 $(p_1\vee p_2)=(p'_1\vee p_2)$，则公理 a 行为不变.

(3)违反相同优先级的公理约束. 相同优先级的公理代表了数据流的不同分支，在程序代码中很可能对应于同一个判断语句的不同分支，因此这些公理的前条件表达式参数(如常量、变量等)之间常直接关联. 如果针对这些参数的变异算子只应用于单个公理，则得到的变异体很可能是违反正常程序行为的. 例如，在实际代码中用条件语句 if($x>3$) … else…实现分支操作，而在构造类别代数的数据描述中，可能描述为公理 a 和 b ($p^a_{\mathrm{pri}}=p^b_{\mathrm{pri}}$)，$p^a_{\mathrm{pre}}:x>3$，$p^b_{\mathrm{pre}}:x\leqslant 3$，对公理 a 施加变异算子 $(x\rightarrow x+1)$，则 $p^a_{\mathrm{pre}}:x>2$，显然不符合正常程序行为.

(4)相同的可观察结果. 安全测试是一种黑盒测试，用户必须通过可观察函数来接收协议，实现对输入数据的处理结果. 如果正常体和变异体之间执行结果的差异无法通过可观察函数判断的话，该变异体是无法区分的.

2.3　测试过程

2.3.1　约定

定义 1　数据流分析过程主要是计算协议报文字段和协议内部环境变量之间的关系，记环境变量集合 $V=\{v\mid v$ 为零元函数$\}$，输入参数的集合 $X=\{x\mid x$ 为构造函数的参数$\}$，X 和 V 中的元素统称为测试参数. 每个测试参数的类别规定了数据格式和取值范围. 例如，对数值性参数 v，可采用区间表示法限定其取值范围为 $r(v)=[\min v,\max v]\mid \min v\leqslant v\leqslant \max v$.

定义 2　数据约束 d 描述当前环境下测试参数的一个值集，定义为二元组 $d=<r,e>$，r 为各测试参数的取值范围，e 为测试参数之间满足关系的表达式集合，可基于 DNF 来描述. 定义 d 与基于 DNF 范式描述的关系表达式 e 之间的集合运算关系为

$$d\cap e=d';\quad d'_r\subset d_r;\quad d'_e=d_e\wedge e';$$
$$d-e=d';\quad d'=d-(d\cap e)$$

定义 3　用断言 $\mathrm{Assert}_i(d_i)$ 表示在不匹配任何优先级高于 i 的公理时，测试变量的取值满足数据约束 d_i.

假定当前公理集合为 A，优先级为 $i(i=1,2,3,\cdots)$ 的公理集合为 $A(i)$，且有断言 $\mathrm{Assert}_i(d_i)$，则

易知

$$d_i = d_{i-1} - \{p^a_{\text{pre}} \mid a \text{ 为公理且 } a \in A(i-1)\}$$

定义 4 变异单元为三元组 $I_{\text{item}}=\langle d, A_n, A_m\rangle$，$d$ 为数据约束，A_n 和 A_m 分别对应变异前和变异后的公理且满足 $p^{A_n}_{\text{post}} \neq p^{A_m}_{\text{post}}$，表明对满足 d 的输入数据在正常体中匹配公理 A_n，在变异体中匹配公理 A_m.

变异单元适用于针对变异体生成的测试数据.

定理 若某个变异体不存在对应的变异单元，则为等价变异体.

证明 根据变异单元定义，若不存在变异单元，则说明此变异体对所有输入数据均执行相同的后条件，这显然为等价变异体.

本文主要考虑变异算子应用于单个或同优先级公理集合生成的变异体，变异体及对应变异单元的生成算法如下.

2.3.2 变异体生成 基于变异体自身约束条件的变异体生成算法(算法 1)如图 1 所示.

```
输入：优先级 i 的公理集合A(i),变异算子 a,设有断言 Assert_i(d_i)
返回值：对A(i)施加变异算子得到的非等价变异体，否则值为空
1. GenerateMutator(A(i), m) {
2.    将 m 应用于 A(i)，得到公理集合A'(i);
3.    for each(公理 a'∈(A'(i)-A'(i)∩A(i))) {
4.        if (a'违反自身约束); 返回值为空;
5.        if( p^{a'}_pre ∩ d_i = ∅ ) 返回值为空;
6.        else {//Assert 断言
7.            p^{a'}_pre ← (d∩p^{a'}_pre);
8.            if (p^{a'}_pre = p^a_pre)  返回值为空;
9.        }
10.   }//end for each a'
11. if (A'(i)的个数大于1且 A'(i)之间存在冲突)  返回值为空;
12. }
```

图 1 变异体生成算法

基于算法 1 生成的变异体可以排除 2.2 节中前 3 种情况导致的等价变异体，但无法识别执行结果与正常体相同的等价变异体.

2.3.3 变异单元生成 假定对某个公理 a 应用变异算子得到了变异公理 a'，该变异算子可应用于公理的前条件、后条件、优先级本身，其中公理的前条件是唯一用于公理匹配过程的部分，因此针对公理的后条件变异过程，恒有 $p^a_{\text{pre}} = p^{a'}_{\text{pre}}$，可直接生成此变异体对应的变异单元 $\langle d_{p^a_{\text{pri}}} \bigcap p^a_{\text{pre}}, a, a'\rangle$. 若变异算子作用于公理的前条件，有 $p^a_{\text{pre}} \neq p^{a'}_{\text{pre}}$，其他属性相同，则存在以下 2 种情况.

(1) 若 $d_{p^a_{\text{pri}}} \bigcap (p^a_{\text{pre}} \wedge \neg p^{a'}_{\text{pre}})$ 不为空，说明满足此条件的测试数据原本匹配公理 a，变异后不再匹配公理 a'，对应的三元组中有 $d = d_{p^a_{\text{pri}}} \bigcap (p^a_{\text{pre}} \wedge \neg p^{a'}_{\text{pre}})$，$A_n = a$，$A_m$ 未知，但肯定有 $A_m \neq a'$.

(2) $d_{p^a_{\text{pri}}} \bigcap (\neg p^a_{\text{pre}} \wedge p^{a'}_{\text{pre}})$ 不为空，说明满足此条件的测试数据在变异前不匹配公理 a，但变异后匹配公理 a'，对应的三元组中有 $d = d_{p^a_{\text{pri}}} \bigcap (\neg p^a_{\text{pre}} \wedge p^{a'}_{\text{pre}})$，$A_m = a'$，$A_n$ 未知，但肯定有 $A_n \neq a$.

基于上述方法可以得到针对变异体 M 的变异单元 I_{item}，但这个变异单元可能是不完整的，缺少的未知公理(A_n 或 A_m)必定在优先级等于或低于 a 的公理集合中. 对不完整的变异单元进行公理匹配的算法(算法 2)如图 2 所示.

```
输入：优先级为i 的公理集合 A(i),不完整的变异单元 I_item
返回值：匹配结束后存在完整的变异单元集合 M_set
1 MatchUP (I_item, A(i)) {
2    for each (a∈A(i)) {
3        if (I_item-d ∩ p^a_pre ≠ ∅ 且 p^a_post 不等于
         I_item 中已有公理的后条件 ) {
4            生成三元组 R=<I_item-d ∩ p^a_pre ,
             I_item-A , I_item-A_m >, 将缺少的公理赋于 a ,
             将 R 添加到 M_Set 中;
5            if (I_item-d - p^a_pre = ∅ )
                 I_item 赋值为空后返回;
7             else I_item-d ← I_item-d - p^a_pre ;
         }//end if
}      }//end for each a
9 }
```

图 2 变异单元匹配算法

根据上述 2 个算法，可以将变异算子集合 M 应用于公理集合，生成所有变异单元集合的算法(算法 3)，如图 3 所示.

```
输入：变异算子集合 M 以及公理集合 A,其中优先级为 i
      的公理集合为 A(i)
返回值：变异单元集合 M_Set
(1) for 公理优先级 i from 1 to n, 执行 (2)～(7);
(2) for each 变异算子 m∈M, 生成变异体
    q←GenerateMutator (A(i), m), 若 q=∅ , 执行 (3)～(7) ;
(3) 设 A(i) 施加m 得到的公理集合为A'(i), A 、B、 A'为公理集合,
    其中 B=A(i)∩A'(i), A=A(i)-B,
    A'=A'(i)-B, 公理 a∈A, 变异后的 a'∈A'; T_Set 和 N_Set
    为不完整的变异单元集合 ;
(4) for each (a'∈A'), 计算变异单元, 如果其为完整的, 则
    添加到 M_Set 中 , 否则添加到T_Set 中;
(5) for each (I_item∈T_Set), 调用 MatchUp(I_item, A'(i)), 计算
    匹配结果, 如果最终 I_item 为空, 则从 T_Set 中删除 I_item;
(6) for each (I_item∈N_Set), 调用 MatchUp(I_item, A(i)), 计算匹
    配结果, 如果 I_item 为空, 则从 N_Set 中删除;
(7) 若 T_Set 不为空, 则将 T_Set 中所有变异单元添加到 N_Set 中;
```

图 3 生成所有变异单元集合的算法

根据算法 3，可以得到所有变异体对应的变异单元集合 M_{Set}.

变异算子应用在全局描述和函数定义生成的变异体与应用在单个公理或同优先级公理集合生成的变异体的区别仅在于，前者允许变异公理集合中存在不同优先级的公理. 本文算法适用于这 2 类变异体，但公理自身约束判定和变异单元生成必须在生成的变异公理集合中进行，限于篇幅，这里略去了算法细节.

2.3.4　测试例生成　在通信协议中，测试例对应于一组输入输出报文的序列，在构造类别代数中表示为一系列可控制函数的嵌套序列. 假定 F 为规范中定义的可控制函数，s_{Seq} 为函数嵌套序列，向量 $\boldsymbol{v}$ 和 $\boldsymbol{x}$ 分别表示当前环境下环境变量和输入参数的取值，则 $F(s_{\text{Seq}},\boldsymbol{v},\boldsymbol{x})$ 表示在当前执行 s_{Seq} 的基础上，通过 F 对协议执行参数为 $\boldsymbol{x}$ 的输入激励. 又假定 $\boldsymbol{v}$、$\boldsymbol{x}$ 之间的计算结果匹配 F 中的公理 a，则使用 e^{a}_{equ} 等式的右边替换 $F(s_{\text{Seq}},\boldsymbol{v},\boldsymbol{x})$，并执行 p^{a}_{post} 中定义的操作，改变环境变量 $\boldsymbol{v}$ 的值，产生相应的输出结果. 限于篇幅，下面使用简单栈[6]的构造类别代数描述来解释上述的函数嵌套序列.

设初始空栈为空，Push 和 Pop 定义为函数，则嵌套函数 Pop(Push(Push(∅,1),2)) 表示对空栈按顺序压入数据 1、数据 2 后执行出栈操作的测试序列.

在安全测试中，测试序列包括 2 部分内容：从初始状态到待测状态的准备序列 s_{Pre}；异常数据的执行过程 s_{Exec} 和测试结果判断序列 s_{Res}. s_{Pre} 可沿用一致性测试中的序列生成方法得到，此处不予以讨论. s_{Exec} 通过调用可控制函数、输入符合为 $I_{\text{item-}d}$ 的数据报文进行，则在正常体和变异体上执行的结果分别为对应公理 $I_{\text{item-}A_n}$ 和等式公理 $I_{\text{item-}A_m}$ 的右边，以及针对环境变量 $\boldsymbol{v}$ 执行的后条件操作 $p^{A_n}_{\text{post}}(\boldsymbol{v})$ 和 $p^{A_m}_{\text{post}}(\boldsymbol{v})$.

对于 I_{item}，如果不存在 s_{Res}，可观察函数为最左边的嵌套函数，且满足 $s_{\text{Res}}(e^{I_{\text{item-}A_n}}_{\text{equ}}$ 等式公理的右边)! = $s_{\text{Res}}(e^{I_{\text{item-}A}}_{\text{equ}}$ 等式公理的左边) 或 $s_{\text{Res}}(p^{A_n}_{\text{post}}(\boldsymbol{v}))! = s_{\text{Res}}(p^{A_m}_{\text{post}}(\boldsymbol{v}))$，说明无法根据此变异单元生成可区分的测试例. 若变异体 M 对应的所有变异单元都无法生成如上结果的判定序列，则 M 为 3.2 节中的第 4 类等价变异体.

3　结　论

本文在基于构造类别代数的协议描述上引入了变异分析方法，并设计了几类变异算子，生成变异体集合，基于变异体自身的约束条件消除了等价变异体，最后基于变异体生成对应的变异单元所构造的安全测试例. 该方法为协议安全测试的研究提供了一个新的研究方向. 同现有研究相比，本文方法的优点是：更能满足安全测试对协议数据流描述的要求；可以限定协议可能存在的错误集合，有利于测试的量化和评估；可以更有针对性地进行测试例构造和测试结果判断，提高测试能力. 本文进一步的研究工作包括了优化变异算子、降低变异分析方法的代价等.

参考文献：

[1] KAKSONEN R. A Functional method for assessing protocol implementation security [EB/OL]. [2009-01-20]. http://www.vtt.fi/inf/pdf/publications/2001/P447.pdf

[2] SHU Xiao, DENG Lijun. Integrated TCP/IP protocol software testing for vulnerability detection [C]// Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing. Los Alamitos, CA, USA: IEEE Computer Society, 2003: 311-319.

[3] MARQUIS S, DEAN T, KNIGHT S. SCL: a language for security testing of network applications [C]// Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative Research. Lebanon, IN, USA: IBM Press, 2005:155-164.

[4] ALLEN W H, DOU C. A model-based approach to the security testing of network protocol implementations [C]// 31st IEEE Conference on Local Computer Networks. Piscataway, NJ, USA: IEEE, 2006:1008-1015.

[5] 周晓煜. 基于形式化描述的协议一致性测试方法的研究 [D]. 合肥：中国科学技术大学计算机科学与技术系, 2004.

[6] 陈伟琳，周颢，赵保华. 利用构造类别代数的协议安全测试方法 [J]. 西安交通大学学报, 2008, 42(12): 1481-1485.
CHEN Weilin, ZHOU Hao, ZHAO Baohua. Constructed type algebra based protocol security testing method [J]. Journal of Xi′an Jiaotong University, 2008, 42(12):1481-1485.

[7] WONG W E. On mutation and data flow [D]. West Lafayette, USA: Purdue University. Software Engineering Research Center, 1993.

（编辑　苗凌）