객체 지향 정형 명세 언어를 사용한 S/Key 시스템 명세*

유희준[1○], 최진영[1], 노병규[2]
고려대학교 컴퓨터학과 정형기법연구실[1]
한국정보보호센터[2]

{hyoo, choi}@formal.korea.ac.kr
nono@kisa.or.kr

Specification of S/Key System Using Object-Oriented Fo Specification Language*

Hee-Jun Yoo[1], Jin-Young Choi[1], Byung-Gyu No[2]
Formal Methods Lab. Dept. CSE Korea University[1]
Korea Information Security Agency[2]

요 약

컴퓨터 기술의 발전으로 정보화 시대를 맞이한 현대에 있어서 "보안 기능의 정형화 설계 방법 연구"는 정보 보호와 완벽한 보안 때문에 매우 중요하다. 이러한 추세에서 전 세계적으로 보안 시스템에 대한 등급을 나누고 있고, 국내에서도 한국정보보호센터에서 침입차단시스템에 대해 K1에서 K7까지의 등급을 매기고 있다. 이 등급을 살펴보면 정형 언어를 사용하여 시스템 명세를 수행하여야만 K5이상의 고 등급을 획득할 수 있다. 또한, 최근에 들어서 소프트웨어 개발 방법에 객체지향 방법론이 많이 사용되고 있는 시점에서 이러한 방법론으로 개발되는 시스템에 대한 정형적인 설계방법이 필요하다. 본 논문은 실제 간단한 규모의 보안 시스템 설계에 직접 적용될 수 있을 수준의 정형기법을 제시하는 것을 목표로 한다. 여기서는 passive attack에 대해 사용자의 패스워드를 보호하기 위한 간단한 스킴인 S/KEY 인증 시스템에 객체지향 정형명세언어인 ZEST와 UML을 적용해 본 결과를 기반으로 보안 시스템을 정형 명세한 경험을 기술한다.

1. 서 론

컴퓨터 기술의 발전으로 정보화 시대를 맞이한 현대에 있어서 "보안 기능의 정형화 설계 방법 연구"는 불특정 다수의 사용자가 자유롭게 사용할 수 있는 인터넷상에서 신뢰할 수 있는 대상하고만 통신을 하며, 자신의 개인 정보의 유출을 최대한 막기 위해서 정보 보호와 완벽한 보안 때문에 매우 중요하다.

어떤 시스템이 신뢰할 수 있는 지에 대한 연구가 진행이 되면서 세계 각 국의 표준 기구에서는 이러한 시스템에 대한 등급을 평가하고 있다. 각 등급을 살펴보면, 시스템 개발 초기부터 정형기법을 이용하여 개발되어진 시스템들이 고 등급을 획득하고 있다. 국내에서도 한국정보보호센터에서 보안 시스템에 대한 등급을 평가하고 있으며, 정형기법을 적용한 시스템에 대해서만 최고등급인 K7 등급을 주고 있다[1].

최근에 들어서 소프트웨어 개발 방법에 객체지향 방법론이 많이 사용되고 있는 시점에서 이 방법론으로 개발되는 시스템에 대한 정형적인 설계 방법이 필요하다. 본 논문은 실제 간단한 규모의 보안 시스템 설계에 직접 적용될 수 있을 수준의 정형기법을 제시하는 것을 목표로 한다. 여기서는 passive attack에 대해 사용자의 패스워드를 보호하기 위한 간단한 스킴인 S/KEY 인증 시스템에 객체지향 정형명세언어인 ZEST[2]와 UML[3]을 적용해 본 결과를 기반으로 보안 시스템을 정형 명세한 경험을 기술한다. 논문의 구성은 2, 3장에서 사용된 개념인 ZEST와 S/Key를 설명한 후에 4장에서 S/Key에 대한 정형 명세를 기술한 후에 5장에서 결론을 맺겠다.

2. ZEST

ZEST(Z Extended with Structuring)은 분산 시스템 모델링을 위해서 시작된 객체지향 개념이 포함된 Z의 확장 언어로 현재 진행중인 ISO Open Distributed Processing (ODP) project에서 개발되었다. ZEST는 특히 open distributed Systems의 명세에 대해서 객체지향 모델링 개념을 표준화하여 명확하게 표현하기 위해서 고안되었다. 객체는 persisting identity를 가지는 개체들을 구현하였다. 객체는 객체의 행동들과 상태에 의해 특성화된다(객체지향 프로그래밍 언어 용어의 'methods'와 'variables'). 각 객체는 객체 명세

1) 본 연구는 2000년 한국정보보호센터의 지원을 받은 것입니다.

의 표현이다. 여기서 객체의 의미는 object specification 이다.[4]

3. S/Key 시스템

S/KEY 인증 시스템은 passive attack에 대해 사용자의 패스워드를 보호하기 위한 간단한 스킴이다. 이 시스템은 네트워크상의 avesdropping/replay attack으로부터 안전한 인증을 제공한다.

S/KEY One-Time Password 시스템에는 두 가지 측면이 존재한다. 하나는 사용자 혹은 클라이언트 측면인데, 적절한 one-time password 가 생성되어져야 한다. 다른 하나는 시스템 혹은 서버 측면으로, one-time password가 검사되어져야 한다.

one-time password는 MD4 one-way hash 함수를 이용해서 생성되고 검사되어진다. one-time password를 생성하기 위해 단 방향 함수를 여러 번 적용한다. 즉, 첫 번째 one-time password는 사용자의 비밀 패스워드(s)를 정해진 특정 수(n)만큼의 단 방향 함수를 수행함으로써 생성되어진다. 예를 들어, n=3이라하면, one-time password는 f(f(f(s)))이 된다. 함수를 적용하는 횟수는 1회 접속시 1회씩 줄게 된다. 그러므로 사용자는 어느 시점에 다다르면 사용자는 시스템을 재초기화해야 한다. one-time password 검사를 위해 처음에 호스트는 수신한 one-time password의 복사본을 저장하는데, 이 복사본에 단방향 함수를 적용한 값과 사용자가 입력한 one-time password를 비교한다. 이 때 그 값이 일치하면 호스트의 one-time password는 바로 전에 입력된 사용자의 one-time password로 갱신된다. 일치하지 않을 경우에는 사용자의 인증요구에 실패한다.

4. ZEST를 이용한 S/Key 명세

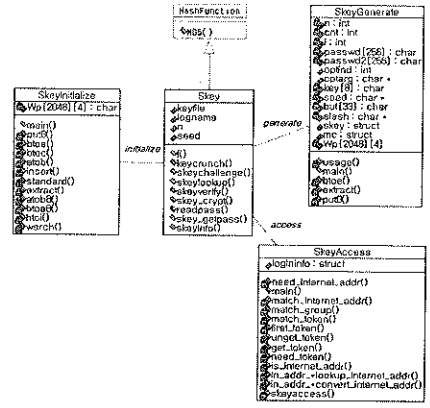
본 논문에서는 S/key 시스템에 대해서 UML의 class diagram을 이용해서 시스템의 클래스를 추출한 후에 UML에서는 명세에 대한 정확성을 명확하게 검사할 수 없기 때문에 ZEST를 사용하여 해당 클래스들에 대한 명세를 수행하였다.

[그림 1]은 Rational Rose98[5]을 사용해서 명세한 S/Key 시스템의 클래스도 이다. 이 시스템은 주 구조인 Skey를 중심으로 Skey를 초기화(initialize), 생성(generation)과 접근(access)의 총 4개의 클래스로 구성된다. 뒤의 3 클래스에서는 Skey의 구조를 상속받아서 사용하게 된다. 지금부터는 [그림 1]을 기반으로 작성한 ZEST 명세에 대해서 설명하겠다.

우선, 명세에 사용되는 형(type)을 정의한다.

[CHAR, FILE, LONG, STRUCT, ARRAY, VOID]

ZEST는 기본 Z에 클래스의 개념을 확장한 것이다. 명세에서는 앞서 기술한 대로 상속되는 클래스인 Skey를 Struct라는 클래스로 명세하고, 생성(generation)과 접근(access) 부분은 각 하나의 클래스로 초기화(initialize)는 주 클래스 안에 명



[그림 1] S/Key 시스템의 클래스도

세하였다.

```
Struct
interface external add Skey, MC, Timeval, Tm,
          In_addr, Passwd, Group, Login_info

[Timeval, Tm, In_addr, Passwd, Group]

  Skey
  keyfile : FILE
  skey_buffer : seq CHAR
  longname, seed, val : CHAR
  n : Z
  recstart :LONG
  ---------------------
  #skey_buffer = 256

  MC
  mc_buffer : seq CHAR
  skip, cnt : Z
  ---------------------
  #mc_buffer = 256

  Login_info
  host_name, port : CHAR
  internet_addr : In_addr
  user : Passwd
```

명세된 클래스를 보면, 다른 클래스와 연계가 되는 부분을 interface 명령어를 사용하여 선언하고, 클래스 내에서만 사용되는 지역형을 선언하였다. 내부에 선언된 스키마를 모두 상속해주었다.

뒷 부분에 선언된 Generation과 Access 클래스도 Struct 클래스를 상속받아서 사용하며, 각각의 인터페이스를 선언하여 각 클래스가 연계될 수 있도록 정의하였다. 여기서 작성한 명세는 기능적인(functional) 부분이 아니라 구조적인(structure) 부분을 명세한 것이다. 따라서, 각 함수를 자세한 동작은 명시하지 않고 함수의 타입과 만족해야하는 조건을 명제 논리(propositional logic)를 사용

하여 명세하였다.

함수와 변수는 공리 기술을 사용하여 명세하고, 주 수행부는 스키마를 사용하여 명세하였다.

다음은 생성(Generation), 접근(Access) Skey Main 클래스에 대한 명세이다.

Generation

inherit Struct

interface external **add** optind, optarg, WP

n, cnt, i, optind : Z

optarg, seed, slash : CHAR

skey : Skey

mc : MC

WP : ARRAY

passwd, passwd2, key, buf : seq CHAR

#passwd = 256 ∧ #passwd2 = 256 ∧ #key = 8 ∧ #buf = 33

main : Z × seq CHAR → Z

Main

argc?, out! : Z

argv? : seq CHAR

---

out! = main(argc?, argv?)

Access

inherit Struct

interface external **add** login_info, skeyaccess

login_info : Login_info

main : Z × seq CHAR → Z

need_internet_addr : VOID → VOID

match_internet_addr : Login_info×Login_info→Z

match_group : Login_info × Login_info → Z

match_token : CHAR → Z

first_token : CHAR × Z × FILE → CHAR

unget_token : CHAR → VOID

need_token : VOID → CHAR

is_internet_addr : CHAR → Z

lookup_internet_addr : CHAR → In_addr

convert_internet_addr : CHAR → In_addr

skeyaccess : FILE×Login_info×Login_info → Z

Main

argc?, out! : Z

argv? : seq CHAR

---

out! = main(argc?, argv?)

Skey_Main

inherit Struct

interface external **add** keyfile, logname, seed, n, f, *keycrunch, skeychallenge, skeylookup,* skeyverify, skeyinfo

n : Z

keyfile, logname, seed : CHAR

f : CHAR → VOID

keycrunch : CHAR × CHAR × CHAR → Z

skeychallenge : Skey × CHAR × CHAR → Z

skeylookup : Skey × CHAR → Z

skeyverify : Skey × CHAR → Z

skeyinfo : Skey × CHAR × CHAR → Z

main : Z × seq CHAR → Z

Main

argc?, out! : Z

argv? : seq CHAR

---

out! = main(argc?, argv?)

5. 결 론

현재 중요한 이슈중의 하나인 정보보호에서 국제적인 기준에 부합하는 고 신뢰도의 시스템을 개발하기 위해서는 개발단계에 정형기법을 사용하는 것이 필수적이다. 각 등급에서 고 등급을 획득하는 것은 높은 부가가치를 얻기 위해서 사회적, 산업적인 측면에서 매우 중요한 과제이다.

본 논문에서는 고 등급의 정보보호 시스템 개발을 위해서 정형 명세를 수행하였다. 여기서 작성된 ZEST 명세는 *Fomaliser[6]*를 사용하여 명세의 정확성을 검사하였다.

앞으로는 시스템의 기능 명세에 대한 부분을 수행할 예정이다.

6. 참고문헌

[1] 국내.외 정보 보호 시스템 평가 가이드, 한국 정보 보호센터, 1998.11

[2] Object Orientation in Z, S. Stepney R. Barden D. Cooper, Springer-Verlag, 1992

[3] Sinan Si Albir, UML : In A Nutshell, A Desktop Quick Reference, O'Reilly, 1998

[4] Elspeth Cusack and Mike Lai, Object oriented specification in LOTOS and Z, In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Foundations of Obje Oriented Languages, Proceedings of the School/Workshop May/June 1990,* number in Lecture Notes in Computer Science. Springer Verlag, 1991

[5] *Rational Software Corporation,* Rational Rose 98 : Using Rational Rose, Rational Software Corporation, 1998.

[6] ZEST Specific Formaliser User Guide Logica UK Ltd., 1999