情報保護學會論文誌
第14券 第6號, 2004. 12

정형명세 기법을 이용한 보안 프로토콜 코드 생성 도구의 보안 소프트웨어 개발 분석*

장 승 주[a)†], 류 대 현[b)‡], 이 철 수[c)], 박 일 환[d)]
동의대학교[a)], 한세대학교[b)], 경원대학교[c)], 국가보안기술연구소[d)]

Analysis of Developing Methodology on the Security Software by Comparing Function for Security Protocol Code Generation Tools

Seung-Ju Jang[a)†], Dae-hyun Ryu[b)‡], Chul-Sool Lee[c)], Il-Hwan Park[d)]
Dongeui University[a)], Hansei University[b)], Kyungwon University[c)], NSRI[d)]

요 약

본 논문에서는 다루는 보안 정형 명세를 지원하는 도구인 SPEAR II와 IFAD VDM-SL Toolbox의 주요 기능과 사용자 환경 및 동작 모듈, 코드 생성 과정을 비교하고, 보안 소프트웨어 개발을 위한 두 도구의 성능적 측면을 살펴본 후 용이한 개발을 제공해주는 도구로부터의 보안 소프트웨어 개발 방안을 제시한다. 본 논문에서 제시하는 보안 프로토콜 코드 생성 도구 기능 비교는 정형 명세 기법을 이용한 소프트웨어 개발 방법에서 정형 명세를 통한 보다 안전한 보안 소프트웨어를 개발하는 방향을 제시한다. 이러한 방향 제시를 통하여 보다 안전한 보안 소프트웨어 개발을 이룰수 있다.

ABSTRACT

Automatic code generating function for security protocol of SPEAR II and IFAD VDM-SL Toolbox supporting formal specification is presented in this paper. Among the functions of these tools we compare and analyze the aspects of functions, users, operation and code generation. And we suggest direction to the developing of safe security S/W. The automatic code generating function for security protocol gives the direction for developing of the safe secure software in formal specification method.

Keywords : *SPEAR II, IFAD VDM-SL Toolbox, formal specification*

Ⅰ. 서 론

인터넷 사용의 보편화로 통신망의 거대화, 정보자산의 고급화, 사용자 집단의 다양화에 반해 공격 기회의 증가, 공격 동기의 상승, 공격자 수의 증가 그리고 공격에 의한 피해가 증대되어 보안은 중요한 문제로 대두되고 있다. 보안 프로토콜 구현은 안전한 정보 전송과 정보시스템의 안전성을 보장할 수 있다. 하지만 프로토콜 개발 절차에서 초기의 요구 조건과는 다른 시스템 오류를 발생시키거나 예기치 못한 시스템 공격을 받을 수 있다. 이는 시스템 개발자가 개발 단계에서 명세와는 다

접수일 : 2004년 8월 11일 ; 채택일 : 2004년 11월 26일
* 본 논문은 국가 보안 기술 연구소 2004년도 연구비 지원에 의하여 연구되었습니다.
† 주저자 : sjjang@deu.ac.kr
‡ 교신저자 : dhryu@hansei.ac.kr

른 설계를 무의식 중에 진행하여 발생시킬 수 있는 부분으로 정형 명세를 통해 원래 설계된 내용을 정확히 반영할 필요가 있다.

정형 명세는 정형 기법의 한 분야로 시스템의 부정확성, 모호성, 불완전성 및 이해 오류를 해결하기 위한 방법으로 사용되고 있으며 시스템의 분석, 설계에 활발히 이용되고 있다. 이러한 정형 명세는 정형 기법에서 분류되고 정형 기법을 지원해주는 도구는 여러 가지가 있는데, 일반적으로 SCR(Software Change Report)를 이용해 명세화하는 SCR Toolset[14], PROMELA(Process Meta Language)라는 검증언어를 사용해서 설계하고 LTL(Linear Temporal Logic)의 구문으로 명세화하여 정확성을 증명하는 SPIN[15], LOTOS 정형 명세의 확인을 위한 분석 지원 도구인 ViLAT(a Visual LOTOS Analysis Tool), STATEMATE[16], Rational Rose등과 같은 그래픽한 정형 명세 도구 등 이 외 여러 가지 정형 도구들이 있다[1,2]. 정형 명세 기법을 적용하여 사용할 수 있는 도구로 GNY 로직을 바탕으로 한 보안 프로토콜 디자인 및 검증 방법을 제공하는 SPEAR II와 논리 기반 정형 명세 도구로 사용되는 IFAD VDM-SL Toolbox가 있는데 이 도구를 이용하여 보안 시스템 개발이 가능하며, 자동적인 코드 생성을 지원해 줌으로서, 안전한 프로토콜 명세를 보장한다. SPEAR II는 보안 프로토콜을 설계하고 분석하는 환경을 프로토콜 설계자에게 제공하는 통합된 다차원적 도구로서 보안 프로토콜의 빠른 설계와 분석 그리고 구현에 도움을 준다. 보안 체계를 구축함에 있어서 의미상의 모호함은 큰 장애요인이 된다. 이는 보안 표준 개발의 명확한 확립을 저해하는 요소로서 정형화된 표준 기술을 통한 보안 시스템의 개발이 필요하며, VDM-SL 명세 언어를 사용하게 됨으로서 좀 더 명확하고 고품질의 소프트웨어 개발 및 생산을 지원하게 된다.

본 논문에서는 정형 기법을 지원하는 여러 가지 도구 중 SPEAR II와 IFAD VDM-SL Toolbox에 정형 명세를 통한 보안 프로토콜 코드를 생성하는 두 도구의 기능을 비교한다. 각 도구 기능 중에서 기능적 측면, 사용자 측면, 동작적 측면, 코드 생성 측면을 비교 분석하여 각 도구의 특징적 요소를 바탕으로 한 보안 소프트웨어 개발 방안을 제시한다.

본 논문의 구성은 2장에서는 정형 명세 기법에 관한 관련 연구를 설명하고, 3장에서는 보안 프로토콜 코드 생성 도구 비교에 대하여 설명하고, 4장에서 보안 정형 명세 자동 생성 도구를 이용한 보안 소프트웨어 개발 방안에 대하여 설명하고, 5장에서 결론을 맺는다.

II. 관련 연구

일반적으로 정형 명세(formal specification)에 사용되는 도구들을 살펴보면, 수학적 표기법을 사용하여 소프트웨어 혹은 하드웨어가 시스템의 특성 및 기능을 서술하며 증명 가능한 사항을 증명하는 기법으로 정의할 수 있다. 즉, 명세에 대한 수학적 완전성, 정확성, 일관성을 증명할 수 있어야 한다. 정형 명세가 가지는 구조적인 특징은 시스템이 먼저 만족해야 할 특성들을 서술할 수 있는 명세 언어를 제공하는 것이다. 이러한 정형 명세의 대표적인 언어에는 Z[3], CSP(Communicating Sequence Process)[4] 방법, Petri Net[6], CCITT 표준인 SDL(Specification and Description Language)[7], LOTOS(Language of Temporal Ordering Specification)[12] 등 이 외에도 많은 정형 명세 언어가 있다. 정형 명세에서 사용하는 수학적 표기 방법은 논리체계의 엄밀성만을 고려하여 애매모호함을 제거한 표기법만을 허용하고 추상성과 구현 종속적인 정보를 초기에 제거함으로써 시스템의 핵심을 올바르게 이해할 수 있으며, 잘 조직된 정리와 공리 체계를 사용하여 제시된 명제에 관한 증명이 가능하다.

본 논문에서 살펴보고자 하는 내용은 보안 기능을 위한 정형 명세 후 코드 생성 도구로서 이용하는 SPEAR II, IFAD VDM-SL Toolbox의 기능 비교이다. SPEAR II 도구는 보안 프로토콜 설계시 그래픽 사용자 인터페이스를 사용할 수 있고, 잘 알려진 몇 가지 분석 기법을 적용한 자동화된 보안 분석과 프로토콜의 효율성을 결정하는데 돕는 실행 평가와 보고, Java와 SDL이 출력 언어가 되는 자동적인 코드 생성, 보안 프로토콜 실행에서의 메타 실행을 통해 프로토콜 명세를

지원하는 환경을 제공한다[1]. IFAD VDM-SL Toolbox에서는 정형화된 방법을 이용하여 생성된 실행코드를 정확한 결과값으로 산출하고 해당 VDM 규격을 표준 기술에 있어서 자연어 규격과 함께 사용하거나 더 나아가 단독적으로 사용될 경우에도 기존의 자연어만을 이용한 표준 기술의 경우보다 정확한 보안 표준 구현을 가능하게 할 수 있다. 즉, 보안 표준들을 VDM 언어로 기술하고 이에 대한 C++코드를 생성시킨 후에, 구현자 스스로가 구현물의 결과와 비교하면서 그 정확성을 시험할 수 있다[2].

Ⅲ. 보안 프로토콜 코드 생성 도구 비교

다음 그림 1은 보안 프로그램의 생성에 있어서 정형 기법을 통하여 명세 및 검증하는 절차를 설명하는 그림이다.

그림 1에서 보안 정형 명세 규격을 정의할 때에는 SPEAR 또는 VDM-SL Toolbox와 같은 도구를 이용하여 구현하고자 하는 소프트웨어를 개발한다. 코드 생성 도구를 이용하지 않고 준비된 보안 소프트웨어 소스 코드를 사용할 수 있다. 시스템 개발 단계에서 볼 때 요구사항의 문제점에 대한 방안으로 정형화된 명세 언어들이 출현했고, 정형 명세로 구현된 시스템이 가장 정확한 명세가 이루어져 신뢰성이 높다는 결론을 얻는다. 다음으로 보안 검증 도구를 사용하여 명세된 코드에 대한 검증과정을 거치게 된다. 검증 과정을 통하여 보안 소프트웨어의 수정 작업을 거침으로서 정형화된 명세 코드를 다시 생성하고 또 다시 검증 과정을 거치게 된다. 명세를 통해 생성된 보안 소스 코드의 최종 검증이 완료됨으로서 개발자는 안정된 보안 소프트웨어를 가지게 된다. 본 논문에서는 보안 정형 명세 코드를 자동 생성하는 경우에 많이 사용되는 SPEAR II와 VDM-SL을 중심으로 비교를 한다.

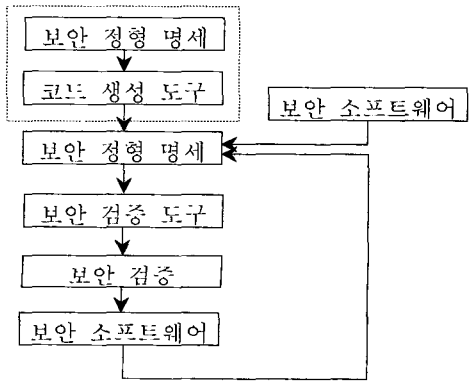


그림 1. 보안 소프트웨어 명세 및 검증

1. SPEAR의 내부 동작 구조

SPEAR II 프로토콜 명세는 GYPSIE 환경에서 그래픽하게 프로토콜 명세 모듈을 제공하고 있으며, Visual GNY와 GYNGER를 사용하고, 명시된 프로토콜의 자동적인 코드 생성을 위해서는 GENIE(GENeration Environment)와 프로토콜 코드 생성기(PCG : Protocol Code Generator)라 불리는 그래픽 코드 생성 환경을 통해 구현이 가능하다[5,10].

1.1 SPEAR에서의 코드 생성

프로토콜 코드 생성기는 추상 GYPSIE 명세를 분석하고 이 명세의 안전한 구현을 생성한다. 목적 언어로 사용되는 언어인 자바는 언어의 우수한 보안 구조와 버퍼 오버플로우 뿐만 아니라 복합 네트워크에 의해 야기되는 일반적인 보안 공격에 대처할 수 있기 때문에 PCG 구현을 위한 목적 언어로써 선택되어졌다. SPEAR에서 생성된 코드 구조는 다음 그림 2와 같다.

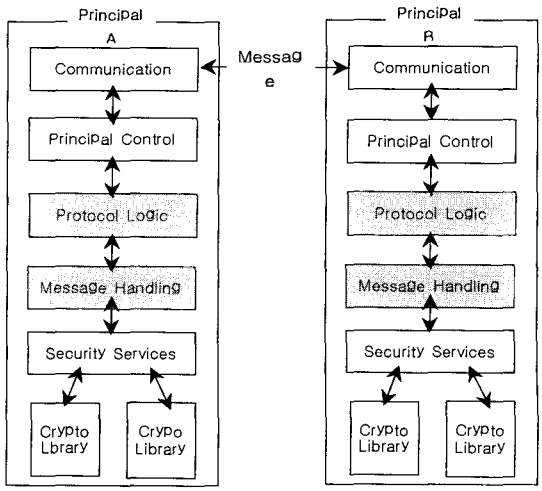


그림 2. SPEAR II 프로토콜 구현의 설계 및 구조

PCG에 의해 생성되는 코드는 그림 2에서 보여주는 구조를 따른다. 그림 2의 각 계층에 대한 설명은 다음 표 1에 나타내었다.

표 1. PCG를 통해 생성되는 코드 구조

Communication layer	principal에 대한 모든 네트워크 접속을 처리하고 SPEAR II를 위해 발전된 Java 런타임 라이브러리 부분
Principal Control layer	실질적인 principal 어플리케이션과 새로운 쓰레드의 생성을 제어
Protocol Logic laye	Principal Control과 Communication layer에 존재하는 구현과는 독립적이고 실질적인 프로토콜 구현 계층
Message Handling layer	
Security-Services layer	Java 런타임 라이브러리의 일부로, 생성되는 구현에 대해 지원되는 Cryptix와 Crypto-J 암호화 라이브러리에서 일반적인 인터페이스처럼 동작

코드 생성 차원에서 직관적이고 순응적인 그래픽 인터페이스의 사용을 통한 GENIE 환경은 추상적으로 코드 생성 설정, 인자, 동작의 명세에서 설계자를 도와준다. 이와 같은 그래픽 인터페이스는 생성 과정 후 소스 코드를 수동으로 수정할 필요없이 명세되어지기 위한 많은 주문형의 소스 코드 설정과 동작을 허용하여 프로토콜 구현 과정을 프로토콜 기술자가 제어할 수 있도록 사용한다. 또한, GYPSIE 명세에서 실행되는 논리적인 분석과 동적인 프로토콜 분석을 효과적인 SPEAR II 프로토콜 구현으로 전송한다. 코드를 생성하는 모든 PCG는 안전한 JAVA 개발을 위한 지침 개요에 따라 철저히 개발된다[1,9].

1.2 IFAD VDM-SL Toolbox의 내부 동작 구조

VDM-SL은 보안 표준의 기술에 적합한 형식 규격어(Formal Specification)를 사용한다. 형식 규격어는 수학적 방법으로 정의되어 원하는 의미를 정확하게 표현할 수 있도록 하고 정형화된 구조를 구현한다. 이와 같은 정형화된 구조를 가지는 보안 표준 기술을 통해 생성된 코드를 참조 구현 코드라고 한다. IFAD VDM-SL Toolbox는 VDM-SL을 통해 보안 표준의 표현 및 실행

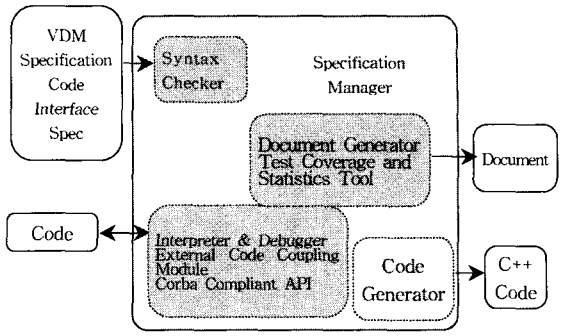


그림 3. VDMTools의 개요

코드 생성의 정확성과 편리성을 제공하는 가장 일반적인 도구로 알려져 있다.

그림 3의 IFAD VDM-SL Toolbox는 VDM 언어의 사용을 지원하는 도구로서 VDM-SL 규격에 대한 신텍스 및 시멘틱스 검사, C++ 코드 생성 외 몇 가지 기능을 가지고 있는 도구이다. 신텍스 검사는 IFAD VDM-SL로 작성된 규격을 입력으로 받아 그 규격에 대한 신텍스 에러를 찾아주는 기능이다. 시멘틱 검사(타입 검사)는 신텍스 에러 체크가 끝난 규격에 대해 시멘틱스 에러를 찾아주는 기능이다. 실행과 디버깅이 Toolbox에서 가능한데 이는 VDM 규격을 인터프리터를 통해 실행시키는 기능으로서 해당 VDM 규격에 정의되어 있는 각각의 함수들을 인터프리터를 통해 규격 수준에서 실행시켜 봄으로써 함수들의 정확한 기술 여부를 판단할 수 있다. 시험 적용 정보(Test coverage information)는 인터프리터를 통한 규격이 실행할 때 입력이 되는 테스트 데이터들이 해당 규격을 얼마나 잘 적용하는지에 관한 정보(coverage information)를 시험 스위트(test suite) 파일에 적재시켜 사용자로 하여금 이를 참조할 수 있게 해주는 기능이다. C++ 코드 생성기(C++ code Generator)는 주어진 VDM 규격에 대해 자동적으로 C++ 코드를 생성해 주는 기능으로서 코드 생성을 위해선 해당 VDM 규격에 대해 신텍스 및 시멘틱스 체크가 이루어져야 한다[2,8,11].

1.3 SPEAR와 IFAD VDM-SL의 기능 비교

SPEAR와 IFAD VDM-SL Toolbox에서 제공하는 각각의 기능들에 대한 비교를 다음 표 2에

표 2. SPEAR와 IFAD VDM-SL Toolbox의 기능 비교

Security Logics	O	X
Code Generation	O	X
Attack Analysis	O	X
Meta-Execution	O	X
Performance Analysis	O	X
syntax checking	X	O
Semantic checking(type checking)	X	O
Execution & Debugging	X	O
Test coverage information	X	O
C++ code Generator	X	O
Latex pretty printing	X	O

정리하였다.

SPEAR에서 사용되어지는 정형 명세 언어는 GNY logic을 통하여 프로토콜 구현을 수행하게 된다. 특히, GNY logic은 보안 프로토콜 개발을 위한 정확성을 제공하는 BAN 로직의 발전된 형태이다. 반면, IFAD VDM-SL Toolbox는 모듈 개념 기반의 구조를 통해 ISO-VDM-SL을 사용함으로써 정형 명세의 개발을 지원한다. SPEAR에서의 보안 로직은 Prolog GYNGER Analyser와 Visual GNY Interface를 통해 제공하고 GNY belief 로직 분석을 위하여 사용되며, 코드 생성은 GENIE 인터페이스 PCG 자바 생성기, SpearUtils 런타임 라이브러리를 통해 제공한다. IFAD VDM-SL Toolbox는 C++ Code Generator를 통해 명세와 구현 사이의 일관성을 유지하고자 VDM-SL/VDM++로부터 C++ 코드의 자동 생성을 지원한다. SPEAR에서의 ACE(Attack Construction Engine)를 통해 공격 분석 모듈을 지원하며 GRACE를 통해 프로토콜의 동적 분석 모듈의 데이터 흐름을 파악한다. VDM-SL Toolbox에서는 Test Coverage와 Statistics Tool을 제공하고 있다. 이는 VDM-SL 명세를 통한 해당 규격에 얼마나 잘 적용되었는지를 알 수 있다. 반면, SPEAR에서는 Scenario Simulation을 통한 Meta-Execution을 지원한다[4,12]. 메타 실행 모듈은 프로토콜의 실제 진행 과정을 보여주는 방법을 제공한다. SPEAR에서 GYPSIE 프로토콜 명세로부터 생성된 소스 코드는 프로토콜 구현의 몇 가지 관점에 대한 실행 측정을 제공하기 위해 증대되어질 수 있다. Performance Analysis는 일명 SPANA (The SPEAR II Performance ANAlyser)라 불려지고 명세를 위한 실행 가능한 소스코드를 제공하게 위해 코드 생성 모듈과 함께 상호 작용하는 SPEAR II 프레임워크의 컴포넌트이고, 제어되는 소스 코드 실행으로부터 실행 정보를 모으며, 프로토콜 기술자에게 그 정보를 보여준다. 다음 표 3는 SPEAR 와 VDM-SL Toolbox의 사용자 환경에 대한 비교 내용이다[1,8,11].

표 3. SPEAR 와 VDM-SL Toolbox의 사용자 환경 비교

사용자 환경	SPEAR	IFAD VDM-SL Toolbox
지원 환경	- GENIE(GENeration Environment) - GYPSIE 프로토콜 명세 환경과 PCG와의 상호 작용	Automatic C++ Code Generator[CGMan]

SPEAR에서는 그래픽 코드 생성 환경을 GENIE에서 제공하고 있다. GENIE는 프로토콜 구현 과정에서의 제어 방법을 제공하고 있다. SPEAR II 프로토콜 명세에서 각 암호화 기능을 위한 암호화 라이브러리와 알고리즘의 명세를 프로토콜 기술자에게 제공하고 GENIE의 4가지 상호 작용 컴포넌트와 GYPSIE, PCG를 통한 코드 생성 환경을 제공하고 있다.

VDM-SL Toolbox의 인터페이스에서 코드와 명세 사이의 인터페이스는 다른 두 단계에서 제공된다. VDM-SL 단계에서의 인터페이스는 구현된 함수들의 VDM-SL 타입들을 명세한다. 코드 단계에서의 인터페이스는 VDM C++ 라이브러리 기반인 코드와 인터프리터 사이의 타입 변환을 확립하는 함수 등, 타입 변환 함수의 정의에 기반하고 있다. 두 부분들의 정의가 강하게 관계되어졌을 때, 다음 그림 4에서 두 인터페이스를 표현한다.

"implmodule MY_MATH"와 "end MY_MATH"사이의 부분은 구현 모듈의 정의이고 VDM-SL에 인터페이스를 구성한다. 구현 모듈은 항상 외

```
implmodule MY_MATH
 exports
   functions
     MyCos : real -> real;
     MySin : real -> real;
   values
     MyPI : real

   uselib
     mymath.lib
end MY_MATH
```

```
#include "metaiv.h"
#include "<math.h>

Generic MyCos(Sequence sq)
{
  double rad;
  rad = Real(aq[1]);
  return (Real(cos(rad)));
}

Generic MySin(Sequence sq)
{
  double rad;
  rad=Real(sq[1]);
  return (Real(sin(rad)));
}
```

그림 4. VDM-SL Toolbox에서의 인터페이스에 사용되는 코드

부로 보내질 수 있도록 모든 생성자를 "exports "부로 선언해 항상 포함하도록 하고, "exports"부로부터 구성자는 다른 모듈로 나타낼 수 있다. "exports" 부는 함수 정의를 위한 서명과 코드에 연관된 값 정의를 위한 타입 정보를 구성한다. 구현 모듈에서 값 정의는 코드의 상수나 변수 정의에 관련한다. "uselib" 부분은 코드가 찾아질 수 있는 동적 링크 라이브러리에 대한 참조를 포함하고 있다. 구현 모듈은 "MY_MATH"와 동적 링크 라이브러리인 "mymath.lib" 사이의 관계를 명세하고 해석하는데 중요한 부분이다.

위 그림 5는 SPEAR에서 프로토콜 명세에서의 각 요소들을 표현하고 있다. 프로토콜을 명세하기 위해서는 프로토콜에 관련된 entities (communicating parties)를 정의하고, 프로토콜에서 사용되는 possessions(data items)를

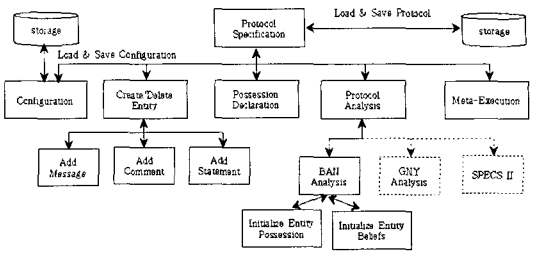


그림 5. SPEAR에서의 프로토콜 명세 과정

정의하여 possessions에 초기값을 할당하는 방법을 제공하는 external functions를 정의하고 사용자와 SPEAR에서 생성된 자바 소스 코드의 상호 작용을 허용하는 단계가 필요하다. 두 번째 요구되는 단계에서는 프로토콜의 BAN 분석을 허용하는 개체의 초기 및 최종 요구 BAN beliefs들을 정의한다. 초기 beliefs는 프로토콜의 실행을 우선적으로 잡아두는 개체이며, 반면 최종 요구 beliefs은 의사 교환의 완성에서 진실임을 나타내는 부분이며, SPEAR 인터페이스와 프로토콜의 설정을 구성할 수 있다. 이러한 과정들은 프로토콜에서 BAN 분석과 실행 분석, 메타-실행을 실행할 수 있다[13].

좀 더 자세한 SPEAR의 개요를 살펴보면 그림 6과 같다. 그림 6에서는 프로토콜 명세를 형성하기 위해 메시지는 본문을 통해 GUI에 들어오게 된다. 이러한 메시지들이 들어옴으로써, 메시지들은 Token Handler를 통해 Crypto-Expression Parser/Compiler를 통과하게 된다. 메시지들은 Crypto-Expression Parser에 의해 분석되어지고, token format에서 컴파일 된다. 어떤 신택틱 또는 시멘틱 오류는 이 부분에서 두

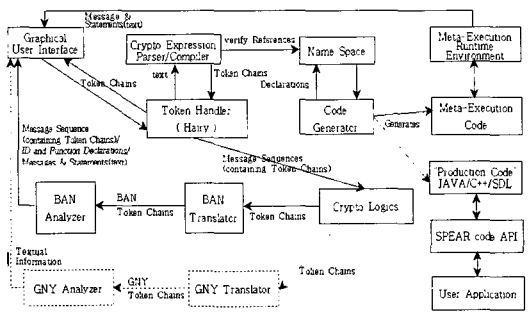


그림 6. SPEAR의 구조적인 개요

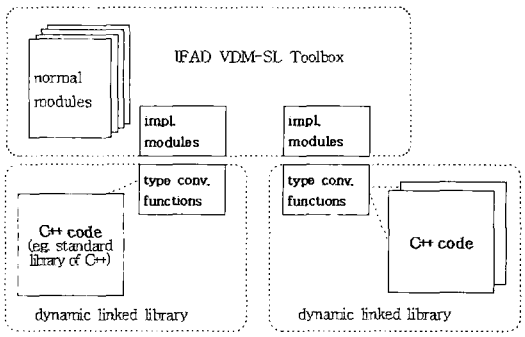


그림 7. Toolbox에서 코드 조합과 명세

루 나타난다.

그림 7는 VDM-SL Toolbox의 코드 조합과 명세의 접근을 보여주고 있다. 점선으로 표시된 부분은 다른 단계를 나타낸다. IFAD VDM-SL Toolbox는 명세 단계와 코드 단위들에 대응하는 코드 단계를 표현한다[11].

이러한 코드 단위들은 해석기 과정에서 동적 링크 라이브러리로써 표현되어진다. 볼드체로 표시된 부분은 코드와 VDM-SL 명세를 조합하기 위해 추가적으로 사용자가 개발해야 하는 부분을 가르키는 것이다. 명세 단계에서의 새로운 종류의 모듈은 구현 모듈(impl. modules)이라 불려진다. IFAD VDM-SL 명세 언어의 모듈 개념은 다중 모듈의 조합을 위한 용이함과 연구 수단으로 사용된다[11].

다음 표 4는 SPEAR와 VDM-SL Toolbox의 동작 모듈 특징을 보여준다.

표 4. 동작 모듈에 대한 비교

동작 모듈	SPEAR	IFAD VDM-SL Toolbox
특징	암호화된 기능의 모듈 생성	코드 조합과 타입 변환을 통한 모듈 생성

중요한 설계 결정은 명세의 재사용을 지원하기 위해서와 코드의 재사용을 강화하기 위한 것이다. 코드의 재사용은 편리한 표준을 확립함으로써 강요된다. 정형 명세의 재사용은 보통 모든 VDM-SL 모듈에 가져올 수 있는 구현 모듈에 의해 성립되어진다.

SPEAR에서는 네임스페이스와 메시지 순서에

표 5. 코드 생성 측면에서의 비교

코드 생성 방법	SPEAR	IFAD VDM-SL Toolbox
타겟 언어	JAVA	C++
생성 모듈	PCG	참조 구현 코드 생성
특징	우수한 보안 구조(GNY logic)	VDM 규격에 따른 코드 생성

서 items(entities, data types, full function references 등과 같은 것들)의 정의된 속성을 사용하는 것은, 코드 생성기가 메시지 구성과 처리(보내기와 받기)를 위해 필요로 하는 코드들을 생성할 수 있다. 현재 자바 소스 코드를 생성하며, 앞으로의 버전에서는 SDL/PR(Phrase Representation)[7]뿐만 아니라 C++도 선택적으로 생성할 수 있다.

VDM-SL Toolbox에서는 VDM-SL 명세로부터 C++코드의 자동적인 생성을 지원한다. 이는 명세와 구현의 일관성을 유지하기 위해 도움을 준다. 코드 생성기는 명세의 실행 가능하지 않은 부분에 대해 사용자 정의된 코드를 포함하기 위한 편리함을 벗어나 모든 VDM-SL의 95%정도 전체적인 실행 가능한 코드를 제공한다. 한 번에 명세가 테스트되어지면, 코드 생성기는 자동적으로 빠른 구현을 얻기 위해 적용될 수 있다.

IV. 보안 정형 명세 자동 생성 도구를 이용한 보안 소프트웨어 개발 방안

3장에서 설명하는 SPEAR II와 IFAD VDM-SL Toolbox의 각 도구들에 대한 특징적 요소를 비교하여 보안 소프트웨어 개발에 적합한 기능들을 적용할 수 있는 방안을 제시한다. 이러한 정형 명세 자동 코드 생성 도구에 대한 기능의 비교는 지금까지 시도된 적이 없다. 본 논문에서 제시하는 방식은 보안 소프트웨어 개발 전에 사전에 보안 취약점을 발견할 수 있는 장점을 제공한다.

보안 프로토콜 설계를 용이하게 하기 위한 도구로서 SPEAR II를 사용하여 빠른 프로토콜 명세가 가능하다. 이 보안 프로토콜을 정의하기 위해선 프로토콜과 관련된 개체들(principals)을 선언하고 메시지 경과 명세가 명확히 표현되어야

하고, 프로토콜 작업을 정확하게 묘사하여야 한다. MSC와 같은 그래픽 표기법은 상위 단계에서 메시지 흐름을 묘사하는데 사용되어질 수 있고, 메시지 구조는 계층적 트리 구조를 사용함으로서 표현되어질 수 있다. SPEAR II에서 설계 모듈을 사용하기 쉽고 보안 프로토콜 명세를 위한 강력한 인터페이스를 제공한다는 것이다. 코드 생성 구문과 관련한 정보의 중요성과 적절성은 명백히 지시되어져야 하며, 필요할 때 제공되어져야 한다. 프로토콜 코드 생성을 위한 보안 설계에 적합한 도구는 SPEAR II에서 제공하는 모듈을 통해 보안 모델의 설계를 용이하게 한다.

IFAD VDM-SL Toolbox는 VDM-SL의 객체 지향 VDM++ 확장을 지원하는 도구이다. 정형 기법을 위한 대부분의 또 다른 CASE 도구들로부터 VDMTools를 분리하는 것은 분석되어지는 명세의 기능적인 경향들에 대한 방법이다. 명세의 속성에 대한 수학적인 증명을 제공하는 것은 가능하지만, 증명은 복잡하고 자원 요구가 많이 든다. 보통의 크기에 대한 명세도, 자동적인 증명 지원 도구는 여전히 충분하게 지원되지 못하고 있다. VDM-SL Toolbox를 통한 소프트웨어 개발 방법은 소프트웨어의 VDM 규격에 따른 코드를 작성한다. VDM-SL은 그래픽한 정형 명세 언어와 다르게 디자인하는 시스템에 대해서 보다 명확하고 상세한 명세를 할 수 있도록 해준다. 이런 종류의 언어들은 일반적으로 시스템에 대한 제약조건을 명세에 완벽하게 명시할 수 있으며, 어떤 동작이 이루어지기 전에 조건이 만족하는지 여부를 먼저 검사한 후에 동작을 가능하게 해준다. SPEAR II와 VDM-SL Toolbox의 여러 가지 측면에서 어떤 기능이 보안 소프트웨어 개발에 강점을 가지고 있는지를 살펴보면 다음 표 6과 같다.

보안 소프트웨어 개발을 위한 명세에서 VDM 규격을 통한 코드 명세는 SPEAR II에서 제시하는 GNY Logics보다 보안 기능이 강화된 코드를 생성할 수 있다. 하지만 코드 생성에 대한 제약조건을 추가하여 높은 보안성을 제공하는 코드를 생성할 수 있지만, 보안성을 높이게 되면 그에 따른 만족성 검사를 수행하기 위해 추가적인 시간과 공간적인 문제로 인해 시스템의 효율성을 떨어뜨릴 수 있다. 따라서 필요한 부분 이외에 불필요한 선언은 하지 않는 것이 좋다. 사용자 환경에서 제공하는 SPEAR II와 IFAD VDM-SL Toolbox는 코드 명세 인터페이스가 각 도구들마다 특징적으로 나타난다. SPEAR II는 코드 생성을 위한 GYPSIE 프로토콜 명세 환경을 제공함으로서 그림 2에서 보는 바와 같이 보안 프로토콜 구현에 따른 계층을 형성한다. VDM-SL Toolbox는 명세 관리자가 존재하여 VDM-SL 구성에 대한 실행 가능한 전체적인 코드를 생성할 수 있다.

표 6. 보안 소프트웨어 개발에 용이한 각 측면에서의 도구

	SPEAR II	IFAD VDM-SL Toolbox
명세	GNY Logic을 통한 프로토콜 규격 명세	VDM 규격을 통한 코드 명세
사용자 환경	GYPSIE 프로토콜 명세 환경	명세 관리자를 통한 모듈 Toolbox 제어
동작 모듈	암호화 기능의 모듈 생성	코드 조합과 타입 변환을 통한 모듈 생성
코드 생성	PCG를 통한 코드 생성	참조 구현 코드 생성

기존의 보안 정형 명세 방법은 정형 명세를 언어를 이용하여 수동으로 이루어지는 경우가 대부분이다. 본 논문에서는 보안 정형 명세를 자동으로 생성할 수 있는 SPEAR와 VDM-SL의 기능적인 특징들을 소개함으로써 보안 소프트웨어 개발을 용이하도록 한다.

V. 결 론

소프트웨어의 개발 활성화로 안정성을 보증하는 보안 소프트웨어의 개발이 필수적이다. 정형 명세 기법을 통한 소프트웨어 개발은 고안전성 시스템의 높은 신뢰성을 보장하는 방법을 제공해 준다. 본 논문에서는 다루는 보안 정형 명세를 지원하는 도구인 SPEAR II와 IFAD VDM-SL Toolbox의 주요 기능과 사용자 환경 및 동작 모듈, 코드 생성 과정을 비교하고, 보안 소프트웨어 개발을 위한 두 도구의 성능적 측면을 살펴본 후 용이한 개발을 제공해주는 도구로부터의 보안 소프트웨어 개발 방안을 제시했다. 이러한 정형 명세 자동 코드 생성 도구에 대한 기능의 비교는 지금까지 시도된 적이 없다. 본 논문에서 제시하는

방식은 보안 소프트웨어 개발 전에 사전에 보안 취약점을 발견할 수 있는 장점을 제공한다.

SPEAR II는 보안과 실행 분석, 메타-실행과 암호화 프로토콜의 자동 코드 생성을 할 수 있는 프로토콜 명세를 지원하는 설계 환경을 제공하여 보안 프로토콜 설계에 유용하게 사용될 수 있는 구조를 가지고 이에 대한 가장 적합한 수준의 코드를 생성하여 준다. 반면, IFAD VDM-SL Toolbox는 보안 표준 기술에 적합한 VDM-SL을 통한 코드 생성으로 VDM 규격에 따른 정형화된 구조의 정확한 표현이 가능하여 SEPAR II와는 달리 보안 표준들을 VDM 언어로 기술하고 이에 대한 C++코드를 생성시킨 후에 구현자 스스로가 구현물의 결과와 비교하면서 그 정확성을 시험할 수 있다. IFAD VDM-SL Toolbox가 가지고 있는 컴파일러 및 GNU C++ 컴파일러에 보안상의 문제점이 없으면, 이들로부터 생성된 코드가 참조구현코드로 활용될 수 있다. 보안 표준들을 VDM-SL로 기술함으로서 해당 소프트웨어에 대한 분석, 이해하는데 있어서 개발자에게 보다 정확한 보안 기능의 소프트웨어를 제공할 수 있다.

참 고 문 헌

[1] Simon Lukell, Chris Veldman, "Automated Attack Analysis and Code Generation in a Unified, Multi-Dimensional Security Protocol Engineering Framework", COMPUTER SCIENCE HONOURS, UNIVERSITY OF CAPE TOWN. CS02-15-00. OCTOBER 2002.

[2] The VDM Tool Group, "User Manual for the IFAD VDM-SL Toolbox" IFAD, Odense, Danmark, 1999.

[3] J. M. Spivey, The Z Notation : A Reference Manual Second Edition, Prentice Hall, 1992

[4] S. Schneider, Concurrent and Real-time Systems : "The CSP Approach", Wiley, 1999.

[5] E. Saul and A. C. M. Hutchison, "SPEAR II : The Security Protocol Engineering and Analysis Resource." In Second Annual South African Telecommunications, Networks and Applications Conference, pp.171~177, Durban, South Africa, September 1999.

[6] http://www.csh.rit.edu/~rick/thsis/doc/PetriThesis.html

[7] http://www.sdl-foru.org/SDL/index.htm

[8] Rene Elmstrom, Peter Grom Larsen, Poul Bogh Lassen," The IFAD VDM-SL Toolbox : A Practical Approach to Fromal Specifications", The Institute of Applied Computer Science, Forskerparken 10, DK-5230 Odense, Denmark

[9] Elton Saul, Andrew Hutchison, "Enhanced Security Protocol Engineering Through a Unified Multidimensional Framework", IEEE JOURNAL ON SLELCTED AREAS IN COMMUNICATIONS, VOL. 21, NO. 1, pp.62~76, JANUARY 2003.

[10] Catherine Meadows, "Formal Methods for Cryptographic Protocol Analysis Emerging Issues and Trends", IEEE JOURNAL ON SLELCTED AREAS IN COMMUNICATIONS, VOL. 21, NO. 1, pp.44~54, JANUARY 2003.

[11] Brigitte Frohlich and Peter Gorm Larsen, "Combining VDM-SL Specification with C++ Code", The Institute of Applied Computer Science, Forskerparken 10, 5230 Odense M, Denmark

[12] D.Hogrefe, Estelle, LOTOS and SDL. Springer-Verlag, 1989.

[13] http://www.cs.uct.ac.za/Research/DNA/SPEAR/dist/spear.html

[14] James Kirby, SCR Toolset : The User Guide, January 1997

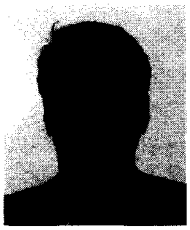
[15] Gerald J. Holzmann, "The model Checker SPIN", IEEE Transactions on Software Engineering, Vol. 23, pp.279~295, May. 1997

[16] I-Logix, "STATMATE MAGNUM Reference Manual ver1.0", 1996

〈著者紹介〉

장 승 주(Seungl-Ju Jang) 정회원
1985년 : 부산대학교 계산통계학과(전산학) 학사
1991년 : 부산대학교 계산통계학과(전산학) 석사
1996년 : 부산대학교 컴퓨터공학과 박사
1987년~1996년 : 한국전자통신연구원 시스템 S/W연구실
2000년~2002년 : University of Missouri at Kansas City, visiting professor
1996년~현재 : 동의대학교 컴퓨터공학과 부교수
〈관심분야〉 운영체제, 분산시스템, Active Network, 시스템 보안

류 대 현 (Dae-hyun Ryu) 정회원
1983년 2월 : 부산대학교 전기기계공학과 졸업
1985년 2월 : 부산대학교 전자공학과 석사
1997년 2월 : 부산대학교 전자공학과 박사
1987년 2월~1998년 2월 : 한국전자통신연구원 선임연구원
1998년 3월~현재 : 한세대학교 IT학부 부교수
〈관심분야〉 정보보호, 영상처리, 통신공학



이 철 수 (Lee Chul Soo) 정회원
1977년 2월 : KAIST 전산과 석사
1981년 2월 : KAIST 전산과 박사
1982년~1993년 : (주) 데이콤
1993년~1998년 : 한국전산원
1999년~2000년 : 한국 정보보호원
2000년~2002년 : 정보통신대학교
2003년~현재 : 경원대학교
〈관심분야〉 정보보호 정책, 공개키기반구조, 침해사고 대응 기술

박 일 환 (Il-Hwan Park) 정회원
1988년 2월 : 고려대학교 수학과 학사
1990년 8월 : 고려대학교 수학과 석사
1996년 2월 : 고려대학교 수학과 박사
1996년 5월~1999년 12월 : 한국전자통신연구원
2000년 1월~현재 : 국가보안기술연구소
〈관심분야〉 정보보호