보안성 정형화 설계 및 검증 기술에 관한 연구[1)]

전철욱[0*], 김일곤[*], 최진영[*], 강인혜[**], 강필용[***], 이완석[***]
[*]고려대학교 컴퓨터학과
{cwjeon, igkim, choi}@formal.korea.ac.kr

[**]서울 시립대학교 기계정보공학과
inhye@uos.ac.kr

[***]한국정보보호진흥원
{kangpy, wsyi}@yahoo.com

A Study on Formal Specification and Verification Mechanism for Security

Chul-wuk Jeon[0*], Il-gon Kim[*], Jin-Young Choi[*]
[*]Dept of Computer Science & Engineering, Korea University

In-Hye Kang[**]
[**]Dept of Mechanical and Information Engineering, Korea University

Pil-Yong Kang, Wan S. Lee[***]
[***]Korea Information Security Agency

요약

최근 보안성의 취약점을 이용한 보안 사고들이 증가하고 있다. 이러한 보안적 취약점을 극복할 수 있는 방법으로 정형적 설계 및 검증이 있으며 그 필요성은 국외뿐만 아니라 국내에서도 점차적으로 늘어 나고 있다. 고등급의 보안 시스템을 개발하기 위해서는 정형화된 설계 및 검증 방법론을 사용해야 하지만 국내에서는 아직 정형적 설계 및 검증 방법에 대한 이해가 부족할 뿐만 아니라 개발자 수준에서 보안성을 보다 쉽게 설계하고 안전성을 검증, 평가 할 수 있는 자동화 도구도 갖추어지지 않은 실정이다. 본 논문에서는 기존의 보안성을 정형적으로 설계하고 검증한 사례를 조사, 분류하여 국내 보안 시스템 개발자들이 활용할 수 있는 가이드를 만드는데 근간을 두고자 한다.

1. 서론

최근 들어, 컴퓨터와 컴퓨터 통신에 대한 사용이 많아짐에 따라, 여러 가지 보안적 취약성을 이용한 보안 사고들이 증가하고 있고 이러한 보안적 취약성으로 인하여 생길 수 있는 사회적 파장도 점점 더 증대되어 가고 있다. 이러한 보안적 취약점을 극복하고자 보안성이 강조되는 보안 프로토콜이나 보안 시스템의 필요성이 대두 되었고 여러 가지 제품 형태로 개발되고 있으며, 실생활에서도 쉽게 사용되고 있다. 하지만 보안 제품의 경우, 일반 제품과 달리 보안성이 보다 중요시 되기 때문에 사용자뿐만 아니라 개발자 또한 안전한 보안 제품을 요구하고 있다. 국외 뿐만 아니라 국내에서도 국제공통평가기준(CC: Common Criteria)[1]평가 제도를 이용해 시스템을 EAL1부터 EAL7으로 나누어 평가하고, 이 중 EAL5이상의 평가 등급을 받기 위해서는 준정형적 설계과정이 필요하다. 하지만 국내에서는 이러한 보안 프로토콜과 보안 시스템에 대한 정형적 설계 및 검증 방법에 대한 인식이 많이 부족하다. 본 논문에서는 기존의 보안성을 정형적으로 설계하고 검증한 사례를 조사, 분류하여 국내 보안 시스템 개발자들이 활용할 수 있는 가이드를 만드는데 근간을 두고자 한다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 제2장에서는 정형적 설계 및 검증 기술을 소개하고, 제3장에서는 기존의 보안성을 정형적으로 설계하고 검증한 사례를 조사, 분류하고 마지막 제4장에서는 결론 및 향후 연구 방향을 제시하고자 한다.

2. 정형적 설계 및 검증 기술

정형적 설계 및 검증 기술은 크게 두 가지로, 정리 증명(Theorem proving)과 모델 체킹(model checking)

[1)] 본 연구는 한국정보보호진흥원 위탁과제로 수행되었음

기술로 나눌 수 있다.[2]

정리 증명은 시스템의 행위 및 특성을 수학적 논리에 근거한 수학적 공식으로 표현하고, 이렇게 표현된 정리나 규칙에 대하여 해석적인 증명을 통해 검증하는 방법이다. 이 방법은 인간의 직관을 이용하여 검증이 가능하고 상태 수가 많더라도 적용 가능하다는 장점이 있지만, 수작업에 의존하므로 검증에 걸리는 시간이 길고 검증상 오류 가능성이 많다는 단점이 있다.

모델 체킹이란 디자인 된 모델 M이 특성 P를 만족하는지 체킹하게 된다. 이 방법은 정리 증명과 달리 자동화된 검증 과정을 거치며, 주로 CTL이나 LTL같은 시제 논리를 사용하여 해당 특성을 표현한다. 이러한 모델 체킹 검증방법은 자동화가 용이하므로 빠르게 검증할 수 있고, 반례(counterexample)를 생성할 수 있으므로 시스템의 정확성을 확인하는데 용이하다는 장점이 있다. 그러나, 모델링된 모든 상태에 대한 검사가 이루어져야 하므로 상태 폭발 문제(state explosion problem)가 발생가능하여 복잡한 상황에는 적용이 제한적이다는 단점이 있다.

3. 보안성 대상과 정형적 설계 및 검증 도구

보안성이란 일반적으로 비밀성, 무결성, 가용성을 의미한다. 비밀성이란 정보의 소유자가 원하는대로 정보의 비밀이 유지되어야 한다는 것이며, 무결성이란 비인가된 자에 의한 정보의 변경, 삭제, 생성등으로 부터 보호하여 정보의 정확성, 완전성이 보장 되어야 한다는 원칙이다. 가용성은 정당한 방법으로 권한이 주어진 사용자에게 정보 서비스를 거부하여서는 안 된다는 원칙이다. 이러한 3가지 특성을 만족해야 보안성을 만족하였다고 말 할 수 있다.

보안성이 적용되는 분야를 크게 보안 프로토콜과 보안 시스템으로 나눌 수 있다. 보안 프로토콜과 보안 시스템을 다음과 같이 정의 할 수 있다.

'보안 프로토콜이란 보안성을 만족하기 위해 암호 알고리즘을 이용하여 메시지를 전달하는 프로토콜이다. 보안 프로토콜에서 사용되는 암호 알고리즘으로는 비밀키 알고리즘, 공개키 알고리즘, 해쉬 알고리즘등이 있다. 대표적인 보안 프로토콜의 종류로는 SSL, Kerboros, TLS, X.509등 있다.

보안 시스템이란 보안성을 만족하는 보안 정책에 따라 만들어진 모델을 실제적으로 구현한 시스템이다. 다시 말해, 허가되지 않은 사용자의 시스템 접근으로 인한 정보 유출 및 변경을 차단하는 시스템이다.

보안 프로토콜을 설계, 검증하는 도구는 크게 1980년 후반부터 사용하기 시작한 정리이론과 1990년 후반부터 사용하기 시작한 모델 체킹으로 나누어 질 수 있다. 각각의 대표적인 도구로는 SPEAR(II)[3], PVS[4]등과 FDR[5], CADP[6], Murφ[7], NRL analyzer[8], SMV[9], SPIN[10]등이 있다.

표 1. 보안프로토콜 설계 및 검증 도구 분류

검증도구	검증 방법	입력 언어	명세 복잡성	프로토콜 복잡성	수동/자동	공개여부
SPEAR	T	BAN, GNY	L	M	◇	P
PVS	T/M	고차 술어논리	M	L	◇	P
FDR	M	CSP	H (L)	H	●	C
Murφ	M	Murφ 언어	H	H	●	P
CADP	M	lotos	H	H	●	R
SMV	M	SMV 언어	H	L	●	P
SPIN	M	ProMeLa	H	L	◇	P
NRL	M	NPATRL	H	M	●	R

(T: Theorem proving, M Model checking, H: High, M: Middle, L: Low, ●: automatic, ◇: Maunal, P: Public, R: Restrict, C: Commercial)

[표 1]은 검증 방법, 입력언어, 명세 복잡도, 프로토콜 복잡성, 수동/자동, 공개여부로 나누어 각 도구를 비교하여 보았다. 검증 방법이란 검증 기법인 정리검증과 모델 체킹을 말하며, 입력언어는 도구에서 입력으로 받아 들이는 언어를 말한다. 명세의 복잡도란 개발자나 검증자 수준에서 난해성을 나타낸다. FDR의 경우 만약 프로토콜을 CSP[11]로 명세한다면 난해성이 상당히 높겠지만(H) Casper[12]라는 컴파일러를 이용해 명세할 경우 난해성이 낮아진다(L). 현재 이러한 명세의 난해성을 해결하기 위해 많은 도구에서 명세를 간략하게 할 수 있도록 Casper와 같은 인터프리터와 그래픽을 이용하여 쉽게 명세 할 수 있도록 개발하고 있다. SPEAR에 경우 그래픽을 이용하여 명세하도록 되어 있다. 프로토콜의 복잡성이란 이론적인 간단한 프로토콜과 실세계에서 사용되는 프로토콜의 복잡성 정도를 나타낸다. 다시 말해, 이론적인 프로토콜을 검증 할 경우라면 낮은 단계(L)이며 실질적인 프로토콜을 검증을 할 경우라면 높은 단계(H)이다.

각각의 설계 및 검증 도구의 특성을 좀 더 정리하면, PVS는 정리 증명과 모델 체킹을 통합한 도구이다. SPEAR(II)는 C++, Java, SDL 코드를 생성해주는 특성을 가지고 있다. 위에서 보았듯이 PVS의 경우 정리 이론을 기초로 하고 있기 때문에 명세하기가 어렵다. FDR과 Murφ 는 다른 정형 도구가 이론적인 보안 프로토콜인 Needham-Schroeder, TMN등을 검증한 것에 반해 실제 사용되고 있는 SSL, Kerberos, X.509등을 검증하였다. 또한 비교적 간단하게 명세 할 수 있도록 만들어진 도구이다. SPIN, SMV는 보안 프로토콜을 검증하였지만 실질적으로 검증에 사용하기에는

조금은 거리가 있다. 이러한 상태 기계를 기초로 하여 만들어진 도구들은 무한 상태를 유한 상태로 모델을 축소하여 명세할 수 밖에 없기 때문에 무한 상태에서 나타날 수 있는 보안적 취약점이 유한 상태로 변화함에 따라 발견할 수 없는 경우가 발생한다.

보안 시스템은 크게 두 가지 접근 통제, 정보 흐름을 보안적 문제점을 가지고 있다. 접근 통제 모델로는 DAC(Discretionary Access Control), MAC(Mandatory Access control), RBAC(Rule Based Access Control)등이 있다[13]. DAC의 경우 사용하기는 편하지만 정보 흐름이나 접근 통제의 문제점이 있고, MAC은 정보 흐름을 막을 수 있지만 사용하는 것에 불편함이 생긴다. 이러한 두 가지 모델을 절충한 모델이 RBAC이다. 보안 시스템을 정형적으로 설계, 검증한 사례는 그리 많지 않으며, 대표적인 도구로는 정리 이론의 Z/eves와 모델 체킹의 CoSec[14]을 뽑을 수 있다.

표2. 보안 시스템 설계 및 검증 도구 분류

검증도구	검증 방법	입력언어	명세 복잡성	수동/자동	공개 여부
Z/eves	T	Z	M	◇	P
CoSec	M	SPA	M	●	R

(T: Theorem proving, M Model checking, H: High, M: Middle, L: Low, ●: automatic, ◇: Maunal, P: Public, R: Restrict, C: Commercial)

Z/eves는 일차 논리와 집합론과 같은 수학적인 기반을 가지고 있는 Z언어[15]를 입력으로 받는다. Z는 초기에는 학술적인 범위에서 사용되었으나 산업 환경에서 실질적인 결과를 내면서 성장하였다. 하지만 정리증명의 단점인 명세의 어려움을 가지고 있고 있다. CoSec은 CCS의 확장형인 SPA(Security Process Algebra)를 명세언어로 사용하고 정보 흐름 속성을 위배하는 상태가 발생하는지 검증해 준다.

4 결론 및 향후 연구 방향

본 논문에서는 보안 프로토콜과 보안 시스템 두 분야를 대상으로 보안성을 정형적으로 설계하고 검증한 사례를 조사하여, 각 대상에 대한 정형적 검증방법 및 검증 도구등을 기준으로 분류해 보았다. 그 결과, 보안 프로토콜의 경우에는 정형적 방법론을 사용한 사례를 다수 찾아 볼 수 있었고 실제로 많이 사용되고 있는 SSL, Kerberos, TLS와 같은 보안 프로토콜을 검증한 사례가 있었다. 하지만 이에 비해 보안 시스템을 정형적으로 설계하고 검증한 사례는 많이 존재하지 않고 있었으며 이러한 사례 또한 실질적인 보안 시스템을 검증하기보다는 이론적인 보안 시스템을 검증하는 한계가 있었다. 향후 연구 방향으로는 실제 사용하는 보안 프로토콜과 보안 시스템을 설계 및 검증해야 하겠으며 더 나아가 보다 명세를 간략하게 할 수 있는 도구 뿐만 아니라 실질적인 설계와 검증을 할 수 있는 도구 개발을 해야겠다. 또한 실제 보안 프로토콜 및 보안 시스템의 설계자 및 개발자들이 참조 할 수 있는 활용가이드를 작성하여야겠다.

5. 참고문헌

[1] 정보보호시스템 공통평가기준, 한국정보보호진흥원, 2002.

[2] Y. Halpern, Y.Vardi, Model checking vs Theorem proving: A Manifesto, Artificial Intelligence and Mathematical Theory of Computation, 1991.

[3] E. Saul and A.C.M. Hutchison. SPEAR II: The Security Protocol Engineering and Analysis Resource, Networks and Applications Conference, 1999.

[4] S. Owre, PVS: Combining Specification, Proof Checking, and Model checking, 8th International Conference on Computer Aided Verification, 1996.

[5] C. Tan, Design and Validation of Computer Protocols Seminar FDR Tool, 2002.

[6] J. Fernandez, CADP A Protocol validation and verification Toolbox, CAV'96, LNCS 1102, pp. 436-440, 1996. , 1996.

[7] C. Mitchell, Automated Analysis of Cryptographic Protocols Using Murphi, IEEE Symposium on Security and Privacy, 1997.

[8] C. Meadows, The NRL Protocol Analyzer: An Overview, Journal of Logic Programming, 1994.

[9] E. Clarke, S. Jha and W. Marrero, Using State Space Exploration and a Natural Deduction Style Message Derivation Engine to Verify Security Protocols, Proceedings of the IFIP Working Conference on Programming Concepts Methods, 1998.

[10] R. Maggi and R. Sisto, Using SPIN to verify security properties of cryptographic protocol, Proceedings of the 9th SPIN Workshop, 2002.

[11] C.A.R Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.

[12] G. Lowe, Capser: A compiler for the analysis of security protocols, 10th IEEE Computer Security Foundations Workshop, 1997.

[13] P. Smarati, Access Control: Policies, Models, and Mechanisms, Foundations of security analysis and design: tutorial lectures, 2002.

[14] V. Peri, Specification and Verification of Security Policies, Foundations of security analysis and design: tutorial lectures, 2002.

[15] R. Focardi and R. Gorrieri, The compositional Security Checker: Tool for the Verification of Information Flow Security Properties. IEEE Transactions on Software Engineering, 1997.