보안 위협위치에서 재사용 가능한 상태전이도를 이용한 보안요구사항 식별

서 성 채[†] · 유 진 호[††] · 김 영 대[†††] · 김 병 기[††††]

요 약

소프트웨어 개발 과정에서 보안 요구사항 식별은 그 중요성으로 최근에 관심이 주목되고 있다. 그러나 기존 방법들은 보안 요구사항 식별 방법과 절차가 명확하지 않았다. 본 논문에서는 소프트웨어 개발자가 보안 위협 위치의 상태전이도로부터 보안 요구사항을 식별하는 절차를 제안한다. 이 과정은 상태전이도를 작성하는 부분과 어플리케이션 의존적인 보안 요구사항을 식별하는 부분으로 구성된다. 상태전이도 작성은 1) 공격자가 소프트웨어 취약성을 이용하여 자산을 공격한다는 것에 기반하여 기존에 발생했던 보안 실패 자료를 이용하여 소프트웨어의 취약성을 위협하는 위치를 식별하고, 2) 식별된 위협 위치에 해당하는 소프트웨어 취약성을 방어, 완화시킬 수 있는 상태전이도를 작성하는 과정으로 이루어진다. 어플리케이션 의존적인 보안 요구사항 식별과정은 1) 기능 요구사항을 분석한 후, 위협 위치를 파악하고, 각 위협 위치에 해당하는 상태전이도를 적용하고, 2) 상태전이도를 어플리케이션 의존적인 형태로 수정한 후, 3) 보안 요구사항 추출 규칙을 적용하여 보안요구사항을 작성하는 과정으로 구성된다. 제안한 방법은 소프트웨어 개발자가 소프트웨어 개발 초기에 모델을 적용하여 쉽게 보안 요구사항을 식별하는데 도움을 준다.

키워드 : 보안, 보안 요구사항, 분석

Identifying Security Requirement using Reusable State Transition Diagram at Security Threat Location

Seongchae Seo[†] · Jinho You[††] · Youngdae Kim[†††] · Byungki Kim[††††]

ABSTRACT

The security requirements identification in the software development has received some attention recently. However, previous methods do not provide clear method and process of security requirements identification. We propose a process that software developers can build application specific security requirements from state transition diagrams at the security threat location. The proposed process consists of building model and identifying application specific security requirements. The state transition diagram is constructed through subprocesses: i) the identification of security threat locations using security failure data based on the point that attackers exploit software vulnerabilities and attack system assets, ii) the construction of a state transition diagram, which is usable to protect, mitigate, and remove vulnerabilities of security threat locations. The identification process of application specific security requirements consist of i) the analysis of the functional requirements of the software, which are decomposed into a DFD(Data Flow Diagram; the identification of the security threat location; and the appliance of the corresponding state transition diagram into the security threat locations, ii) the construction of the application specific state transition diagram, iii) the construction of security requirements based on the rule of the identification of security requirements. The proposed method is helpful to identify the security requirements easily at an early phase of software development.

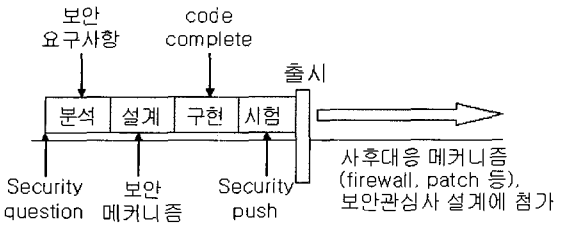
Key Words : Security, Security Requirement, Analysis

1. 서 론

인터넷이 활성화되면서 분산 비즈니스 어플리케이션이 주를 이루고 있다. 분산 환경 하에 운영되고 있는 어플리케이션은 공격자에 의해 유용한 자산을 공격 받고 있다. 공격자는 소프트웨어에 존재하는 취약성을 이용하여 보호되어야 할 자산을 위협하고 있으며, 점점 더 악의적이고 정교한 공격을 통해 자산의 기밀성, 무결성, 가용성에 심각한 손실을 일으키고 있다. 이는 어플리케이션이 네트워크로 연결되어 있는 환경과 보안을 고려하지 않고 설계되어 공격자의 공격에 취약한 상태가 되었다는 것을 의미한다.

보안 취약성을 방어, 완화하기 위한 방법은 크게 어플리케이션 보안과 소프트웨어 보안으로 나눌 수 있다[14]. 어플리케이션 보안은 (그림 1)의 출시 오른쪽과 같이 소프트웨

† 준 회 원 : 전남대학교 전산학과 박사수료
†† 정 회 원 : 전남대학교 대학원 전산학과(이학박사)
††† 준 회 원 : 전남대학교 대학원 전산학과 박사수료
†††† 종신회원 : 전남대학교 전자컴퓨터정보통신공학부 교수
논문접수 : 2005년 7월 19일, 심사완료 : 2005년 12월 5일



(그림 1) 보안 방법

어 개발이 완료되고 난 후, 보안 취약성을 방어, 완화, 제거하는 방법이다. 개발을 완료한 후 보안에 대해 생각하는 것은 보안 관점에서 기능을 설계했다고 볼 수 없으며, 비용이 많이 드는 방법이다[22]. 소프트웨어 보안은 (그림 1)의 출시 왼쪽과 같이 소프트웨어 개발 초기 단계에서부터 보안을 고려하여 소프트웨어를 구축하는 방법이다[14]. 즉 소프트웨어 보안은 보안성이 있는 소프트웨어를 설계하는 것으로, 보안 관심사(consideration)를 설계에 단순히 첨가시키는 것이 아니라, 소프트웨어 개발 초기인 분석 단계에서부터 추출하고, 이를 이용하여 설계, 구현, 시험, 배포 단계에 일관성(consistency) 있게 적용하여 소프트웨어를 개발하는 것이다[22]. 이는 버그를 구현단계에서 고치려면 설계 단계에서 고치는 것보다 10배나 많은 시간과 비용과 노력이 들어가고, 테스트 단계에서 고치려면 구현 단계에서 고치는 것보다 10배가 더 들어가기 때문이다[6, 25]

몇몇 연구자들은 소프트웨어 개발 초기에 위협을 파악하고, 위협을 제거할 수 있는 대응수단을 만든 후, 이를 이용해 보안요구사항을 파악하였다. 소프트웨어 개발 초기에 보안 요구사항을 식별하는 방법은 크게 세 가지로 분류할 수 있다. 첫째, 보안과 같은 비기능요구사항(NonFunctional Requirement)을 NFR그래프 등을 통하여 추출하고, 이를 기능요구사항에 통합하는 과정을 통해 보안 요구사항을 파악한다[8-11]. 둘째, anti-model 작성을 통하여 보안 요구사항을 추출하였다. 소프트웨어 기능요구사항을 파악하고, 이를 위협하는 anti-model을 작성하고, 위협을 방어, 완화, 제거하는 대응수단을 만든 후, 대응수단을 통해 보안 요구사항을 식별한다[1, 2, 4]. 셋째, 기존모델인 UML(Unified Modeling Language)을 확장하여 보안요구사항을 추출한다[11, 12, 16-20, 30]. 기능요구사항을 모델링한 유스케이스(use case)를 작성하고, 이 유스케이스를 공격 또는 위협하는 유스케이스의 특별한 형태인 misuse case, abuse case를 작성한다. 이 misuse case, abuse case를 완화시킬 수 있는 새로운 유스케이스를 만들어 가면서, 최종적으로 보안 위협을 완화시킬 수 있는 보안 유스케이스를 작성하여 보안 요구사항을 추출한다.

소프트웨어 개발단계에서부터 보안 문제를 적용한 선행 연구는 개발자가 보안 목표를 설정하고, 기능요구사항을 파악한 후, 보안전문가들이 위협 모델, 공격 모델을 작성하여 보안 취약성을 완화해 나가는 방법이다. 소프트웨어 개발자가 소프트웨어를 개발할 때, 보안전문가의 도움 없이 보안요구사항을 추출하고, 보안기능이 첨가된 분석, 설계, 구현이 되도록 하는 방법이 필요하다. 즉 보안 전문가는 보다 효율적으로 문제와 해결책을 식별하고, 명명하고, 토론할 수 있도록 도와주는 모델을 작성하고, 소프트웨어 개발자는 소프트웨어 개발을 할 때 보안 전문가가 만들어 놓은 보안 문제 해결책 즉 모델을 이용해 보안 요구사항을 식별하는 방법이 필요하다.

본 논문에서는 보안 전문가가 아닌 소프트웨어 개발자가 쉽게 보안 요구사항을 추출할 수 있도록 보안 요구사항을 식별하는 과정을 크게 두 부분으로 나누어 진행할 것이다. 한 부분은 보안 요구사항을 쉽게 파악할 수 있도록 도와주는 모델을 작성하고, 다른 부분은 소프트웨어 개발자가 작성된 모델을 적용하여 어플리케이션 의존적인 보안요구사항을 추출하는 절차를 기술할 것이다. 먼저, 보안 위협 위치에서 보안 취약성을 방어하고, 제거할 수 있는 보안 요구사항 식별 모델인 상태전이도를 작성한다. 보안 실패 자료를 이용하여, 어플리케이션에서 보안취약성이 발생했던 위치, 즉 위협 위치를 파악한다. 그리고 위협 위치에 해당하는 보안 취약성을 방어, 완화시킬 수 있는 상태전이도를 작성한다. 소프트웨어 개발자는 소프트웨어 개발과정에서 보안 요구사항을 식별할 수 있는 상태전이도를 이용한다. 소프트웨어 기능요구사항을 DFD(Data Flow Diagram)로 모델링하고, 모델링된 DFD에 보안 위협 위치를 식별한 후, 보안 위협 위치에 해당하는 상태전이도를 적용한다. 적용된 상태전이도를 어플리케이션에 의존적인 형태로 재구성한 후, 보안 요구 사항 추출 규칙을 이용하여 보안요구사항을 파악한다. 본 제안 방법은 소프트웨어 개발자가 쉽게 보안요구사항을 식별하고, 보안 소프트웨어 개발에 일관성을 가지도록 도와줄 것이다.

이 논문의 나머지는 다음과 같이 구성되어 있다. 2장에서는 관련 연구로서 소프트웨어 취약성을 방어하고, 완화시키는 보안 요구사항 식별방법에 대해서 알아보고, 3장에서는 소프트웨어에 보안 위협 위치를 식별하고, 이 위협 위치에 해당되는 취약성 분석을 통하여 상태다이어그램 작성하는 방법을 논하고, 4장에서는 실제 소프트웨어 개발과정에서 보안 요구사항을 식별하는 과정을 기술하고, 마지막으로 이 논문의 기여도를 요약하고, 향후 연구를 개괄하고자 한다.

2. 관련 연구

2.1 소프트웨어 개발과 보안요구사항

보안은 위협으로부터 자산을 보호하는 것을 목적으로 하고 있으며, 위협은 보호되어야 할 자산이 남용될 수 있는 가능성이다[7]. 보안 요구사항이란 비기능 요구사항으로써 보호되어야 할 자산 및 서비스를 위협하는 것에 대한 분석을 기초로 하여, 이 위협을 완화 또는 제거하기 위한 보안 관련 요구사항들을 말한다.

소프트웨어 개발에서 요구공학은 기능요구사항을 합리적으로 다루고 있지만, 비기능 요구사항의 일종으로 소프트웨어 품질을 개선하기 위해 사용되는 보안은 일반적으로 포함하고 있지 않았다[31]. 단순히 안전하고 보안이 요구되어져야 한다고 제기만 하고 설계가 완성되고 나서 보안 기능을 첨가함으로써 분석, 설계, 구현의 불일치로 인하여 보안 취약성을 가지

게 된다. 보안 취약성은 보안요구사항이 빈약하게 표현되고, 잘못 이해하거나 일관성 있게 해석되지 않을 때 발생한다[5].

높은 추상화 수준에서 모든 어플리케이션은 기본적으로 가치 있고, 잠재적으로 취약한 자산을 가질 수 있다[25]. 즉, 보안 요구사항은 보안 메커니즘보다 훨씬 더 표준화될 수 있다. 이는 보안 요구사항을 식별하기 위해 모델을 만들어 재사용할 수 있음을 말한다. 이 모델을 이용해 소프트웨어 개발자는 불필요한 아키텍처 메커니즘을 명세하지 않고 보안 요구사항 파악에 주력할 수 있다. 또한 보안 요구사항의 잠재적인 재사용성은 요구공학자가 보안 요구사항을 식별하고, 분석하고, 관리하는데 많은 이득을 볼 수 있다[4, 12, 26-28].

2.2 취약성 분석

보안 취약성은 소프트웨어가 정상적으로 운영되어도, 공격자가 시스템에 대한 권한을 침해하고, 운용을 통제하며, 자료를 변조하고, 주어지지 않는 신뢰를 획득하지 못하는 소프트웨어의 오류를 말한다. CERT®에서는 자산의 가용성, 기밀성, 무결성 손실을 일으키는 취약성 유형을 입력 검증 오류(Input Validation Error), 접근 검증 오류(Access Validation Error), 예외 조건 처리 오류(Exceptional Condition Handling Error), 환경변수 오류(Environmental Error), 설정 오류(Configuration Error), 경쟁 조건(Race Condition), 설계 오류(Design Error) 등으로 분류하였다.

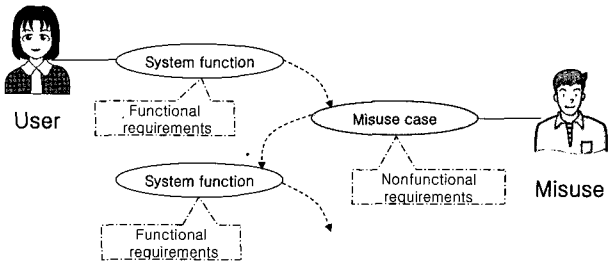
소프트웨어에 존재하는 보안 취약성을 제거하기 위해서는, 소프트웨어 개발과정에서 잠재적으로 발생하는 결함을 식별하는 취약성 분석 과정이 필요하다. 보안 취약성이 소프트웨어에 존재하는 이유는 소프트웨어 개발 시 보안 요구사항이 빈약하게 표현되거나, 잘못 이해되거나 불일치되었을 때 발생하게 된다[25]. 현재 운영되고 있는 소프트웨어는 유사한 보안 취약성으로 인해 보안 공격이 발생하고 있다. 정보시스템 공학자는 공격자가 유사한 보안 취약성을 이용한다는 것에 기초하여 보안 취약성을 방어하고, 완화하기 위해 보안 실패 자료를 사용하였다[1, 5, 12, 26]. 보안 실패 자료는 보안에러 발생경로, 사전 예방방법, 향후 동일한 에러(error)가 없도록 하는 방법을 찾을 수 있도록 도와준다.

Bishop[5]은 분류화 과정을 통하여 취약성을 분류하였으며, 취약성에 대한 기본 특징 집합(character set)을 추출하는 방법을 제시하였다. 특히, 소프트웨어의 보안취약성과 특징 집합 간의 관계를 파악하여, 취약성을 막는 방법을 제안하였다.

2.3 기존 보안요구사항 식별 방법

2.3.1 UML을 확장하여 보안요구사항 식별하기

소프트웨어의 보안 요구사항을 식별하기 위해 UML의 구문과 의미를 확장하는 방법을 사용하였다. (그림 2)와 같이, 사용자와의 상호작용을 담당하는 유스케이스로 모델링한 후, 이 유스케이스를 위협하는 모델인 *misuse case*를 작성하고, 작성된 *misuse case*를 완화하는 새로운 유스케이스를 작성하는 과정을 통하여 보안 유스케이스를 작성하고, 이를 통해 보안 요구사항을 식별하였다[16, 17]. *Misuse case* 방법



(그림 2) Use and Misuse Case 상호작용

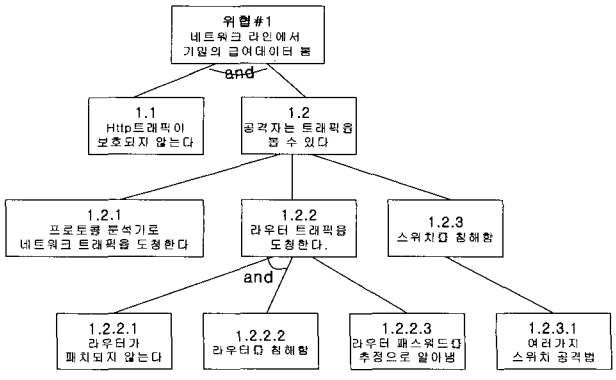
은 보안 위반을 찾으려는 공격자와 어플리케이션 사이의 상호작용에 중심을 두고 있다[16, 17]. *Abuse case*[20] 방법은 *misuse case*방법과 유사한 방법을 통해 보안요구사항을 분석하고 명세하는 방법이다.

UMLsec[30]은 소프트웨어 시스템의 보안 측면을 설계하고 모델링하기 위하여 UML을 확장하여 보안 요구사항을 식별하였다. 특히 UMLsec은 보안 분야에 대한 특별한 훈련을 받지 않고도, 보안기능이 첨부된 소프트웨어를 개발할 수 있는 도구를 제공한다.

2.3.2 Anti-model 작성을 통한 보안 요구사항 식별

보안 목표에 장애물을 생성하고, 이를 해결하는 목표지향 프레임웍에 기반한 접근법을 사용하여 어플리케이션 의존적인 보안 요구사항을 모델링하고, 명세하고, 분석하였다. Lamsweerde[4]는 다음과 같은 순서에 따라 보안 요구사항을 식별하였다. ① 기밀성, 개인보호, 무결성, 가용성, 인증, 부인방지 등과 관련된 명세 패턴을 초기화한다. ② 명세를 위협하는 anti-model을 만든다. ③ 위협에 대응수단을 유도하고, 이 모델로부터 품질요구사항에 부합하는 것을 선택함으로서 새로운 요구사항인 보안 요구사항을 정의한다.

Anti-model은 고장트리, 또는 위협트리를 이용하여 작성한다. (그림 3)과 같이 목표를 부정하는 위협을 트리의 뿌리노드에 놓고, 뿌리노드의 위협을 수행하기 위한 공격을 하위노드로 구성한다. 이 하위노드는 AND와 OR의 논리적 특성을 이용하여 계층적으로 구성한다[1, 4, 23, 25, 32]. Howard [25]는 트리기반 접근 방법인 위협 모델링을 통해 보안 요구사항을 모델링 하였다. 우선 위협(threat)이 무엇인지 파악하



(그림 3) 위협 트리

기 위한 과정으로 어플리케이션의 기능을 DFD로 작성하고, 이를 위협하는 위협을 파악한다. 파악된 위협을 (그림 3)과 같이 위협트리(threat tree)로 작성하고, 위험도(risk)를 결정하고, 이 위협에 대한 대응수단(countermeasure)을 만들어 위협을 제거 또는 완화시켰다. 이 과정을 통해 개발자는 소프트웨어에 존재하는 위험을 파악하고, 공격자가 어떻게 공격할 지 예상함으로써, 보안요구사항을 식별하였다. 이 방법은 소프트웨어 개발에서 보안관심사를 개선하기 위해 보안 실패 자료를 사용하였으며, 구조화되고 재사용 가능한 형태로 공격정보(Attack Information)를 문서화하였다.

2.3.3 보안 패턴

Moore[1]는 보안 문제와 같이 반복되는 설계 문제를 해결하기 위해 패턴을 적용하였다. 보안 패턴은 특별한 문맥에서 발생하는 보안 문제를 기술하고, 그 해결책을 제시한다. 이 방법은 유사한 공격으로부터 발생하는 보안 문제 해결을 위한 접근법으로, 체계적이고 구조적인 방법으로 증명된 보안 해결책을 찾는데 도움을 준다[26-29].

2.4 보안 요구사항 식별 방법 분석

보안 요구사항을 식별하기 위한 방법으로는 UML을 확장한 misuse case, abuse case, UMLsec 등의 방법, 보안 목표에 대한 anti-model 작성을 통해 식별하는 방법, 반복되어 발생하는 보안 문제를 보안패턴으로 만들어 식별하는 방법 등이 있다. 이러한 식별 방법들을 작성자, 소프트웨어 개발자 사용 용이성, 적용모델, 재사용성 등의 특성을 이용하여 정리하면 <표 1>과 같다. UML확장, anti-model, 보안 패턴을 통한 보안요구사항 식별 방법은 보안 전문가들에 의해 수행되어야 하며, 소프트웨어 개발자들이 사용할 때 보안 관련 지식을 습득해야 하는 어려움이 있다.

〈표 1〉 보안 요구사항 식별 방법 비교

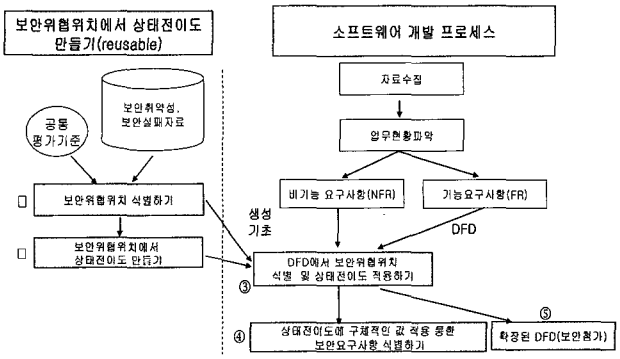
식별 방법 / 특성	UML확장	Anti-model	보안패턴	제안방법
보안 요구사항 작성자	보안전문가	보안전문가	보안전문가	SW개발자
SW개발자 사용성	어려움	어려움	보통	용이함
적용모델	UML 확장	트리기반	포괄적	상태전이도
재사용성	보통	보통	좋음	좋음
보안 요구사항 추출 용이성	보통(보안 지식 요함)	보통(보안 지식 요함)	보통	용이함

3. 보안 위협 위치에 상태전이도 작성하기

소프트웨어 개발자는 보안 요구사항을 식별할 때, 보안 전문가의 도움 없이 보안 취약성을 방어하고, 완화시킬 수 있는 보안 요구사항 추출 방법이 필요하다. 보안 요구사항 추출을 위한 기존 연구에서는 보안 문제의 원인이 소프트웨어에 있다고 판단하였다[22]. 즉 보안 문제를 발생시키는 보안 취약성은 소프트웨어의 빈약한 분석, 설계, 구현의 결과라 인식하고, 소프트웨어개발 초기에 보안 요구사항을 식별하고자 하였다. 그러나 기존 방법들은 보안 요구사항을 식별할 때 보안 전문가의 도움이 필요하였다.

본 논문에서는 보안 전문가가 아닌 소프트웨어 개발자가 보안 요구사항을 쉽게 식별할 수 있도록 하기 위하여 보안 요구사항을 식별하는 과정을 크게 두 부분으로 나누어 진행할 것이다. 3장에서는 (그림 4)의 왼쪽과 같이 보안 요구사항을 쉽게 파악할 수 있도록 도와주는 모델을 작성한다. 4장에서는 (그림 4)의 오른쪽과 같이 소프트웨어 개발자가 작성된 모델을 적용하여 어플리케이션 의존적인 보안요구사항을 추출하는 절차를 기술할 것이다.

이 장에서는 (그림 4)의 왼쪽과 같이 보안 요구사항을 쉽게 파악할 수 있도록 도와주는 모델을 작성한다. 우선, 보안 취약성으로 인해 발생한 보안 실패 자료를 이용하여 소프트웨어의 보안 위협 위치를 식별한다. 그리고 이 위협 위치에서 발생하는 보안 취약성을 파악하고, 취약성 분석을 통하여 취약성을 방어, 완화, 제거할 수 있는 대응수단을 파악한다. 파악된 대응수단을 이용하여 소프트웨어 개발자가 쉽게 사용할 수 있는 모델을 작성한다.



(그림 4) 보안요구사항 추출과정

3.1 보안 위협위치 식별하기

보안 요구사항은 자산의 취약성으로 인해 발생하는 보안 위험을 막거나 제거하는 것을 목적으로 한다. 보안 위험은 가치 있는 시스템 자산에 해를 일으킬 의도가 있는 공격자에 의한 위협으로 인하여 발생한다. 보안 요구사항 식별을 도와주는 모델을 작성하기 위하여 본 논문에서는 소프트웨어에서 공격 위협이 발생하는 위치를 파악하고, 이 위협 위치에 존재하는 보안 취약성을 파악하여, 취약성을 방어, 완화, 제거할 대응수단을 식별한다. 즉 공격정보(특히 보안 실패자료 등)를 체계적이고 재사용가능한 형태로 모델링 할 필요가 있다.

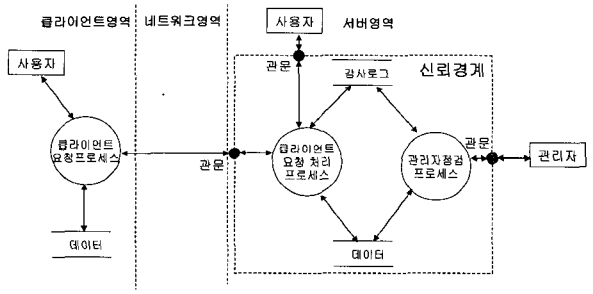
우선 보안 실패 자료를 이용하여 소프트웨어에서 공격 위협이 발생할 수 있는 위치를 식별한다. 식별과정은 다음과 같다. ① (그림 5)와 같이 단순화된 DFD(Data Flow Diagram)를 작성하고, 신뢰경계를 표시한다. 보안 위협은 신뢰경계 밖에서 신뢰경계(reliance boundary) 안으로, 신뢰경계

〈표 2〉 위협 위치 동치관계 만들기

v_1 R v_2 R는 위협의 위치 $[v_1] = [v_2]$ 동치 [T] = { x ∈ V | x의 위협이 되는 위치}

안에서 신뢰경계 밖으로 자료의 이동, 신뢰경계 내에서 자료의 권한 없는 사용에 의해 발생한다. ② 보안 취약성자료를 Bishop[5]의 취약성분석 방법과 <표 2>의 위협 위치 동치관계 만들기를 이용하여 동치류로 분류한다. 취약성 집합(set)을 V={v_1, v_2, v_3, ... , v_n}이라 하고, V와 관련된 특징 집합을 C={c_1,c_2,…, c_k}라 하자. 취약성의 특징 집합은 취약성이 가지고 있어야 할 조건이다. 동치관계 R을 두 취약성이 소프트웨어의 보안 위협이 발생될 수 있는 위치가 같은 관계라 하자. v_1과 v_2가 R의 관계가 있다면 이 취약성을 동치류로 분류한다. 동치류를 만들 때의 원칙은 취약성의 가장 주된 원인이 무엇인가를 기준으로 만든다. 파악된 동치류 집합은 사용자 입력처리 부분, 네트워크 처리 부분, 자료 저장 부분이다.

③ ②에서 파악된 동치류 집합은 보안 위협 위치로 대치한다. 이 과정을 통해 파악된 보안 위협 위치는 사용자 입력처리 부분, 네트워크 처리부분, 자료저장 부분이다. 파악된 위협 위치에서 발생하는 위협과 그 위치에 발생하는 보안취약성 유형을 정리하면 <표 3>과 같다.



(그림 5) 단순화된 DFD

〈표 3〉 보안 위협 위치와 보안취약성

위협 위치	소프트웨어 취약성	위협
사용자 입력처리 부분	- 입력검증오류 - 접근검증오류 - 설계오류 - 예외 조건 처리 오류	- 권한획득 - 속이기 등
네트워크 연결처리 부분	- 설계오류 - 접근검증오류	- 네트워크 도청, 감청 - 스푸핑 - 행동부인
자료저장 처리 부분	- 접근검증오류 - 설계오류 - 경쟁조건 - 예외조건 처리 오류	- 자료탈취 - 접근권한 통제 실패

3.2 보안 위협 위치에서 재사용 가능한 상태전이도 작성

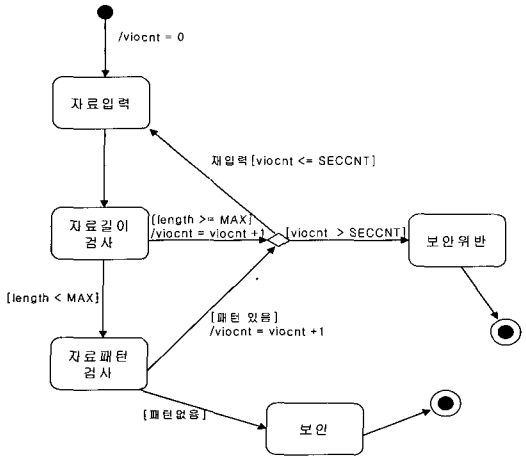
보안 위협 위치에 해당되는 보안 취약성을 파악하고, 대응수단을 식별한 후 이를 모델로 작성해야 한다. 우선, 보안 위협 위치에서 발생하는 보안 취약성들을 파악한다. 그리고, 파악된 취약성을 방어할 수 있는 대응수단을 만든다. 대응수단으로는 자산이 특정 상태에서 계속 진행될 수 있는지, 보안 문제를 유발시키는지 판단해야 한다. 즉 자산의 상태 변화를 검사, 추적할 수 있는 방법이 필요하다. 본 논문에서는 자료의 상태변화를 효율적으로 표현하는 UML의 상태전이도(State Transition Diagram)를 사용하여 모델링할 것이다. 이 상태전이도는 보안 위협 위치에 발생하는 위협의 대응수단으로서 보안 요구사항을 식별하는데 도움을 줄 것이다.

보안 위협 위치에 상태전이도를 만들기 위한 절차는 다음과 같다.

① 보안 위협 위치에 존재하는 취약성을 식별한다.
② 취약성으로부터 특징 집합을 추출한다(보다 자세한 정보는 bishop[5] 논문을 참고).
③ 특징 집합을 이용하여 상태전이도의 상태를 유도한다.
④ 특징 집합의 발생조건을 이벤트로 만든다.
⑤ 상태전이도 정교화하기. 이벤트와 조건이 올바르다면, 보안 상태가 되고, 그렇지 않으면 보안 위반상태가 된다. 최종상태에 도달할 때까지 ②, ③, ④단계를 반복한다.

보안 위협 위치에서 상태전이도는 소프트웨어 개발자에게 보안 요구사항을 식별할 수 있는 모델을 제공한다. 특히 이 모델은 새로운 보안 취약성이 발생하면, 이 보안 취약성이 발생한 보안 위협위치를 파악하고, 특징 집합을 파악한다. 그리고 보안 위협 위치의 상태전이도에 새로운 상태를 첨가한다. 2절에서 파악된 보안 위협 위치마다 상태전이도가 작성된다. 이와 같은 과정을 통해 작성된 상태전이도는 사용자 입력부분 상태전이도, 네트워크 연결부분 상태전이도(클라이언트쪽(client) 상태전이도, 서버쪽(server) 다루는 상태전이도), 자료저장 부분 상태전이도, 감사로그(audit log)를 처리하는 상태전이도 등이다.

본 논문에서는 보안 위협 위치가 사용자 입력 부분에서 상태전이도를 작성하는 것만 살펴보고, 나머지 보안 위협 위치의 상태전이도 작성은 생략하겠다. 사용자 입력 부분 보안 위협 위치에서 발생하는 보안 취약성은 사용자 입력 오류(Input Validation Error), 버퍼오퍼플로우(Buffer Overflow), 포맷스트링 버그(Format String Bug) 등이 존재한다. 이 취약성들의 특징 집합은 입력 자료 길이 검사, 입력 자료의 패턴검사 등을 하지 않는 것이다. 상태전이도의 상태는 이러한 특징 집합을 검사하는 것이다. 즉 모델링하려고 하는 상태전이도의 상태로 될 수 있는 것은 입력검사, 접근검사, 자료 패턴 검사 등이 될 수 있다. 상태전이도의 전이조건(Transition Condition)은 상태를 변화시키는 것이다. 이때 전이조건에 특정 값을 부여하지 않는다. 사용자 입력처리 자산에서 자료의 상태 변화는 다음과 같이 일어난다. 우선 초기 조건으로 보안 위반(violation)건수를 0으로 설정하고, 자료가 입력되면, 자료 길이 검사 이벤트에 의해 자료 길이 검사 상태로 변하고, 자료 길이를 검사하여 조건에 맞



(그림 6) 사용자 입력 부분 상태전이도

으면, 자료패턴 검사 상태로 가고, 맞지 않으면 재입력을 요청한다. 자료 패턴 검사도 마찬가지 과정을 거친다. 이와 같은 과정을 거쳐서 작성된 상태전이도는 (그림 6)과 같다.

4. 모델 기반 보안요구사항 식별하기

이 장에서는 소프트웨어 개발자가 3장에서 작성한 모델로부터 어플리케이션 의존적인 보안 요구사항을 식별하는 과정을 기술한다((그림 4)의 오른쪽). 개발하고자 하는 어플리케이션의 기능 요구사항을 DFD로 작성하고, 작성된 DFD에 3장에서 파악한 보안 위협 위치를 표시한 후, 표시된 보안 위협 위치에 해당되는 상태전이도를 적용한다. 그리고 적용된 상태전이도의 전이 조건을 어플리케이션에 의존적인 구체적인 값으로 표시한 후, 보안요구사항 추출규칙에 적용하여 보안요구사항을 식별한다.

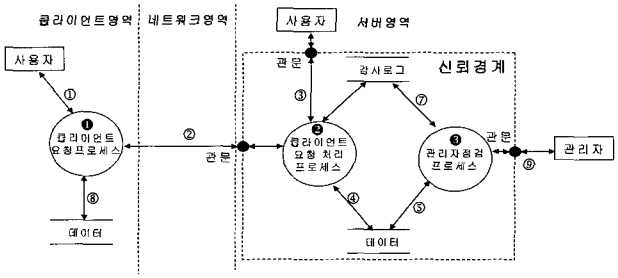
4.1 기능 요구사항에서 위협 위치 식별하기

어플리케이션 의존적인 보안요구사항을 식별하는 첫 번째 단계로서 보안 위협 위치를 식별한다. 우선 소프트웨어의 기능 요구사항을 DFD를 이용하여 작성한다. 작성된 DFD를 클라이언트 영역, 네트워크 영역, 서버영역으로 구분하고, 신뢰경계를 표시한다. 그리고 소프트웨어에서 보호해야 할 자산인 자료와 서비스를 파악한다. 기능 요구사항을 분석하여 작성된 DFD에 3장에서 식별했던 보안 위협 위치를 표시한다((그림 7)의 ① 에서 ⑨). 작성된 DFD에서 식별된 보안 위협 위치는 다음과 같다. 사용자 입력 부분 위협 위치는 (그림 7)의 ①③⑨, 네트워크 부분 위협 위치는 (그림 7)의 ②, 자료저장 부분 위협 위치는 (그림 7)의 ④⑤⑥⑦⑧이다.

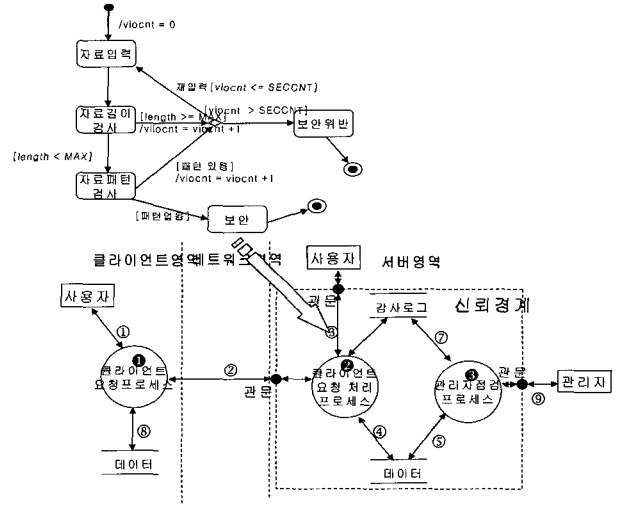
4.2 식별된 위협 위치에 상태다이어그램 적용하기

DFD에 표시된 보안 위협 위치에 3.2절에서 파악한 위협 위치에 상태다이어그램을 (그림 8)과 같이 적용한다. (그림 8)은 (그림 7) DFD에 ❷프로세스에 ③보안 위협 위치에 사용자 입력 부분 상태전이도를 적용한 모습이다. 그리고, 작성된 DFD에서 가치 있는 자산(자료, 서비스 등)을 다루는 프로세스를 찾는다((그림 7)의 ❶❷❸). 이 프로세스들은 하나 이상의 상태전이도를 가질 수 있다. 하나 이상의 상태전이도를 가지고 있다는 것은 프로세스에 다양한 형태의 위협이 존재한다는 것을 알 수 있다. 프로세스가 두 개 이상의 상태전이도를 가지고 있다면, 상태전이도를 적용하는 순서가 있다. 이 순서는 DFD의 프로세스에서 발생하는 자료 흐름 순서와 동일하게 적용한다.

(그림 7)의 DFD에 적용되는 상태전이도를 DFD의 프로세스별로 살펴보면 다음과 같다. (그림 7)의 ❶에는 사용자 입력 부분 상태전이도, 네트워크(클라이언트) 부분 상태전이도, 자료 저장 부분 상태전이도가 적용되어야 하고, (그림 7)의 ❷❸에는 사용자 입력 부분 상태전이도, 접근 권한 상태전이도, 네트워크(서버) 부분 상태전이도, 자료 저장 부분 상태전이도 등이 필요하다. 특히, (그림 7)의 프로세스 ❷에서는 네 개의 상태전이도가 적용된다. 이때 상태전이도 적용 순서는 자료 흐름 순서에 준해서 네트워크(서버) 부분 상태전이도, 사용자 입력 부분 상태전이도, 접근 권한 상태전이도, 자료 저장 부분 상태전이도 순으로 적용한다. 이 적용 순서에서 자산(asset)의 경중을 판단한 후에 상태전이도가 불필요한 경우는 생략해도 된다.



(그림 7) 요구사항을 분해한 DFD



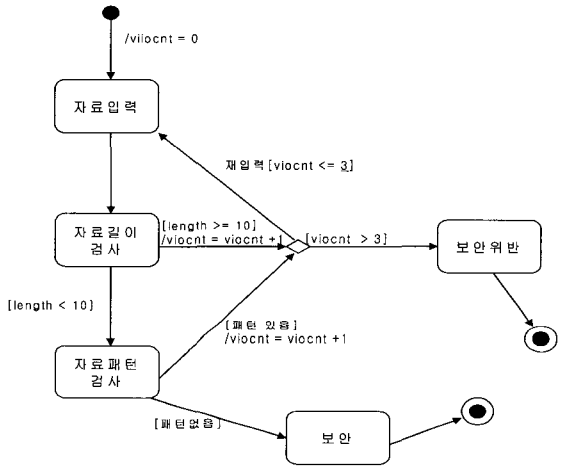
(그림 8) 상태전이도 적용하기

4.3 보안 요구사항 식별하기

앞 절에서 DFD에 적용된 보안 위협 위치에 상태전이도의

전이조건을 어플리케이션 의존적인 형태로 변환한 후, 보안 요구사항 추출 규칙을 적용하여 보안요구사항을 식별한다. 먼저, 어플리케이션에서 자산의 상태 변화를 일으키는 구체적인 조건을 상태전이도 전이 조건의 구체적인 값으로 적용한다. (그림 8)과 같이 사용자 입력 부분의 상태전이도를 위협 위치에 적용한 후, (그림 9)와 같이 어플리케이션에 의존적인 상태전이도로 변화시킨다. 그리고 나서, <표 4>의 보안 추출 규칙을 적용하여 어플리케이션 의존적인 보안 요구사항을 추출한다.

(그림 8)에 보이는 것처럼 DFD ❷프로세스의 보안 위협위치 ③에 적용된 사용자 입력 부분 상태전이도를 (그림 9)와 같이 상태의 전이 조건을 어플리케이션 의존적인 값으로 적용하여 어플리케이션 의존적인 상태전이도를 만든다. 즉 (그림 8)의 최대 입력 길이 MAX값을 10, 보안 위반 건수 SECCNT를 3 등으로 어플리케이션 의존적인 값을 적용하여 상태전이도를 만든다. 그리고 나서, <표 4>의 보안 요구사항 추출 규칙을 적용하여 보안 요구사항을 식별하면 <표 5>와 같다.



(그림 9) 어플리케이션 의존적인 상태전이도

〈표 4〉 보안 요구사항 추출 규칙

상태전이도에서 - 전이 조건은 보안요구사항으로 함 - 이벤트는 보안요구사항으로 함

〈표 5〉 식별된 어플리케이션 의존적인 보안 요구사항

1. 자료 입력의 길이는 10자 미만이어야 한다. 2. 자료 입력의 길이가 10자 이상일 때는 보안위반 수를 1 늘리고 가) 보안위반 수가 3이하이면 자료의 재입력을 요청한다. 나) 보안위반 수가 3보다 크다면 보안 위반 경고 메시지를 보내고 종료한다. 3. 자료 입력 문자가 보안위반을 일으키는 패턴(%,&,..) 문자가 있는지 검사한다. 가) 있다면 보안위반 수를 1 늘리고, 2로 돌아간다. 나) 없다면 입력된 자료는 안전하다고 판단하고 종료한다.

4.4 관련 연구와의 비교

제안한 방법과 기존 방법인 *anti-model, UML*을 확장한 모델 등과 비교하면 <표 1>과 같다. 제안한 방법은 보안 요구사항을 식별하기 위해 위협 위치를 식별하는 재사용 가능한 방법을 제시하고, 이 위협 위치에 해당하는 상태전이도를 재사용함으로서 소프트웨어 개발자가 보안 전문가의 도움 없이도 쉽게 보안 요구사항을 도출 할 수 있도록 하였다. 특히 기존 방법이 위협에 대한 대응수단을 통해 보안 요구사항을 추출하는 것과는 다르게, 상태전이도의 전이 조건 값을 어플리케이션 의존적인 값으로 변경한 후 보안 요구사항을 추출함으로서, 소프트웨어 개발자가 용이하게 보안 요구사항을 추출할 수 있다.

5. 결론 및 향후 연구

소프트웨어 개발자가 소프트웨어 개발과정 초기에 보안 요구사항을 식별하는 방법은 최근에 관심을 가지고 있는 분야이다. 기존의 연구에서는 소프트웨어 개발자가 어플리케이션 의존적인 보안요구사항을 식별하기보다는 보안 전문가에 의해 보안 요구사항을 식별하였다. 기존 방법인 *anti-model, misuse case, abuse* 등은 보안 목표에 위협을 설정하고, 이 위협들을 방어, 완화, 제거할 수 있는 대응수단(coutermeasure)을 통하여 보안 요구사항을 추출하였다. 그러나 이 방법들은 보안 전문가에 의해 수행되고, 설계와 구현에 추적성을 확보하기 어려운 면이 있다. 본 논문은 소프트웨어 개발 초기에 보안 위협을 방어, 완화, 제거하도록 하는 보안 요구사항을 식별하는 모델을 제공하고, 이 모델과 기능 요구사항 간의 자연스런 통합을 통해 보안 요구사항을 식별하도록 하는 방법을 제공하였다. 소프트웨어 개발자가 보안 요구사항을 식별하도록 하기 위해, 어플리케이션에서 보안 위협위치를 파악하는 방법과, 그 위치에 해당되는 취약성을 방어, 완화, 제거할 수 있는 상태전이도를 작성하였다. 이 모델은 소프트웨어 개발자가 새로운 어플리케이션을 개발할 때, 위협위치를 파악한 후 적용되어 보안요구사항을 쉽게 식별할 수 있도록 하였다. 또한 소프트웨어 개발 초기단계에 보안을 고려하여 개발함으로써 개발비용을 저렴하게 할 수 있다.

향후 연구로는 이 논문을 토대로 보안 요구사항의 자동 추출 도구를 개발하는 것이 필요하다. 또한, 분석된 보안 요구사항이 설계와 구현에서 일관성을 유지하고, 추적성을 보증하는 방법이 연구되어야 한다. 그리고 보안 요구사항 식별 방법을 기초로 하여 시험 단계에서 사용될 시험사례(Test Case)를 추출하는 방법을 개발하여야 한다.

참 고 문 헌

[1] A. P. Moore, R. J. Ellison, R. C. Linger, "Attack Modeling for Information Security and Survivability", CMU/SEI-2001-TN-001, Mar., 2001.
[2] A. P. Moore, R. J. Ellison, L. Bass, M. Klein, F. Bachmann, "Security and Survivability Reasoning Frameworks and Architectural Design Tactics", CMU/SEI-2004-TN-022, 2004.
[3] A. Hall and R. Chapman, "Correctness by Construction", IEEE Software Vol.19, No.1, pp.18-25, 2002.
[4] A. V. Lamsweerde, "Elaborating Security Requirements by

Construction of Intentional Anti-Models", Proceedings of the 26th International Conference on Software Engineering (ICSE'04), pp.148-157, 2004.
[5] M. Bishop, "Vulnerabilities Analysis", Web proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (RAID'99), 1999.
[6] B. Boehm, 'Software Engineering Economics', Prentice-Hall, 1981.
[7] CC, Common Criteria for Information Technology Security Evaluation, Version 2.1, CCIMB-99-031, Aug., 1999.
[8] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, 'Non-Functional Requirements in Software Engineering', Kluwer Academic Publishers, 1999.
[9] L. M. Cysneiros and J. C. S. P. Leiter, "Using UML to Reflect Non-Functional Requirements", Proceedigns of the 11 CASCON, IBM Canada, Toronto Nov 2001, pp.202-216, 2001.
[10] L. M. Cysneiros, J. C. S. P. Leiter and J. S. M. Neto, "A Framework for Integrating Non-Functional Requirements into Conceptual Models", Requirements Engineering Journal, Vol.6, Issue2, pp.97-115, Apr., 2001.
[11] L. M. Cysneiros and J. C. S. P. Leiter, "Integrating Non-Functional Requirements into data modeling", Proceedings of the 4th International Sysmposium on Requirements Engineering, pp.162-171, 1999.
[12] D. G. Firesmith, "Specifying Reusable Security Requirements", Journal of Object Technology(JOT), Vol.3, No.1, 2004.
[13] D. G. Firesmith, "Security Use Case", Journal of Object Technoly(JOT), Vol.2, No.3, pp.53-64, May/Jun, 2003.
[14] G. McGraw, "Software Security", IEEE Security & Privacy, pp.80-83, Mar/Apr., 2004.
[15] G. Hoglund, G. McGraw, 'Exploiting Software : How to break code', Addison Wesley, 2004.
[16] G. Sindre and A. L. Opdahl, "Capturing Security Requirements through Misuse Cases", Proc. 14th Norwegian Informatics Conference, Norway, pp.26-28, Nov., 2001.
[17] I. Alexander, "Misuse Cases: Use Cases with Hostile Intent", IEEE Software Jan/Feb, 2003, pp.58-66, 2003.
[18] I. V. Krsul, 'Computer Vulnerability Analysis', PhD thesis, Purdue University, 1998.
[19] J. McDermott, "Extracting Security Requirements by Misuse Cases", Proc. 27th Technology of Objected-Oriented Languages and Systems(TOOLS-37 Pacific 2000), Sydney, Australia, pp.120-131, 2000.
[20] J. McDermott, C. Fox, "Using Abuse Case Models for Security Requirements Analysis", Proc. Annual Computer Security Applications Conference (ACSAC'99), pp.55-64, 1999.
[21] J. A. Whittacker and M. Howard, "Building More Secure Software With Improved Development Processes", IEEE Security & Privacy, Vol.2, Issue 6, pp.63-65 Nov/Dec., 2004.
[22] J. Viega, G. McGraw, 'Building Secure Software', Addison Wesley, 2004.
[23] L. Liu, E. yu, J. Mylopoulos. "Security and Privacy Requirements Analysis within a Social Setting", Proceedings of the 11th IEEE International Requirements Engineering Conference, pp.151-161, 2003.
[24] L. M. Cysneiros and J. C. S. P. Leiter, "Nonfunctional requirements: from elicitation to conceptual models", IEEE Transactions on Software Engineering, Vol.30, No.5, pp.328-350, May, 2004.
[25] M. Howard and D. C. LeBlanc, 'Writing Secure Code', 2nd Ed., Microsoft, 2003.
[26] M. Schumacher and U. Roedig, "Security Engineering with Patterns", In PLoP Proceedings 2001.
[27] M. Schumacher, "Security Patterns And Security Standards", in PLoP Proceedings 2001.
[28] M. Schumacher and U. Roedig, "Security Engineering with Patterns," in PLoP Proceedings 2001.
[29] G. McGraw, B. Potter, "Software Security Testing", IEEE Security & Privacy, Vol.2, Issue 5, pp.81-85, Sep/Oct., 2004.
[30] J. Jürjens, "UMLsec : Extending UML for secure systems development", In UML 2002, 2002.
[31] P. T. Devanbu, S. Stubblebine. "Software Engineering for Security : A Roadmap", ICSE 2000, pp.227-239, 2000.
[32] 서정국, 최경희, 정기현, 박승규, 심재홍, "인터넷 보안 시뮬레이션을 위한 공격모델링", 정보처리학회논문지C, 제11-C권 제2호, pp.183-192, 2004.
[33] 장세진, 최상수, 이강수, 최희봉, "보안 요구사항 도출 및 명세를 위한 CC기반 Misuse Case 모델", 정보과학회 2004년 춘계 학술대회 Vol.31, No.1, pp.0277-0279, 2004.

서 성 채

e-mail : scseo@chonnam.ac.kr
1994년 전남대학교 전산학과(이학사)
1997년 전남대학교 대학원 전산학과 (이학석사)
2001년 전남대학교 전산학과 박사수료
관심분야 : 소프트웨어 모델링, 요구공학, 소프트웨어 보안, 정형기법 등

유 진 호

e-mail : jhyou@chonnam.ac.kr
1991년 전남대학교 전산통계학과(학사)
1995년 전남대학교 대학원 전산학과 (이학석사)
2006년 전남대학교 대학원 전산학과 (이학박사)
관심분야 : 소프트웨어공학, 시스템 보안, 프로그램 분석 등

김 영 대

e-mail : utan@chonnam.ac.kr
1997년 전남대학교 전산학과(이학사)
1999년 전남대학교 대학원 전산학과 (이학석사)
2001년 전남대학교 대학원 전산학과 박사수료
관심분야 : MDA, Formal Method, 소프트웨어 보안 등

김 병 기

e-mail : bgkim@chonnam.ac.kr
1978년 전남대학교 수학교육과(이학사)
1980년 전남대학교 대학원 수학과 (이학석사)
2000년 전북대학교 대학원 수학과 (이학박사)
1981년~현재 전남대학교 전자컴퓨터정보통신공학부 교수
1995년~현재 한국정보처리학회 이사 및 부회장
관심분야 : 소프트웨어공학, 객체지향시스템, 컴포넌트 개발 등