논문 2012-2-5

SOA를 위한 모델기반 취약점 분석

김현하*, 김유경*, 도경구**

Model Based Vulnerability Analysis for SOA

Hyunha Kim*, Yukyong Kim*, Kyung-Goo Doh**

요 약

본 논문은 SOA 보안 취약점을 분석하기 위해 서비스 계층 구조상의 특징을 고려하여 취약점을 식별하고 평가한다. 제안된 분석방법은 모델기반 접근방법으로써, 안전한 서비스 모델을 도출하기 위해, 비즈니스 모델에서 보안요구사항을 표현하는 방법과 서비스의 취약점 식별 과정을 정의한다. 분석적 검증을 통해 제안된 방법이 유용성을 평가하였다.

Abstract

This paper is to identify and assess vulnerabilities of services considering the nature of service layers for analyzing vulnerability of SOA security. It is a model driven approach which provides the way to present security requirements of the business model and identify the vulnerabilities of the services to extract the secure service model. We validate the proposed method with the analytic evaluation because the predictive nature of our methodology poses some specific challenges for its validation.

한글키워드 : SOA 보안, 취약점 분석, 모델기반 보안공학

1. 서 론

현재 국가의 주요 시스템과 비즈니스 및 서비스들이 소프트웨어에 의해 운용되고 제어됨에 따라, 공정 제어 기기부터 상업적 응용 제품까지 다양한 주요 애플리케이션들과 에너지, 수송 및 항공 제어 시스템, 전기통신 등 사회 기반시설이 보다 안전하고 신뢰적인 소프트웨어에 의존하고 있다. 따라서 소프트웨어는 데이터와 자원 들을 보호하도록 설계되고 구현되어야 할 뿐만 아니라 소프트웨어 자체에 대한 보안을 제공해야 한다 [1]. 소프트웨어공학이나 보안공학이 개선된 소프트웨어 개발을 위한 여러 방법들을 제공하기는 하지만, 여전히 소프트웨어는 많은 보안상의 취약점(vulnerability)을 내포하고 있다. 소프트웨어 취약점은 코드 내의 결함(defect)이나 결점(flaw)으로 보안 공격에 악용 될 수 있으며, 일단 공격이 발생하면 다양한 영향을 끼치게 되며, 그 영

*, ** 한양대학교 ERICA 컴퓨터공학과
** 교신저자(email: doh@hanyang.ac.kr)
※ 본 연구는 교육과학기술부/한국연구재단 우수연구센터 육성사업의 지원으로 수행됨 (과제번호 2012-0000469)
접수일자: 2012.8.29 수정완료: 2012.10.2

향력은 포착하기 힘든 영역까지 분포할 수 있다. 심지어 이론적인 것보다 실제로 훨씬 더 파괴적인 결과를 야기할 수 있다.

SOA(Service-Oriented Architecture)와 웹서비스는 비즈니스 요구를 실시간으로 충족시킬 수 있는 유연성을 보장함으로써 기업의 생산성을 극대화 할 수 있는 소프트웨어 개발 패러다임이다. SOA는 명확한 인터페이스 정의와 느슨한 연결(loosely coupling)을 통해 유연성을 증가시키며, 사용자 애플리케이션의 기능을 서비스 형식으로 전달하는 개방적 구조를 가지고 있다[2]. 이런 개방적인 구조적 특징으로 인해 보안에 매우 취약했으나, OASIS 및 W3C 등의 국제표준기구에 의해 지난 수년 동안 SOA 보안에 관련한 다양한 표준이 마련되었다. 그러나 이들 표준은 메시지 수준의 보안과 프로토콜에 대한 보안 표준으로, SOA의 구조적 특징에 따른 위협요인에 대한 분석이나 서비스 계층구조에 따른 취약점 파급효과에 대해서는 고려하지 않고 있다.

SOA 기반 소프트웨어 개발 환경에서 전사적 위험관리 차원의 통합적인 체계 마련을 위해서는 서비스 수준의 보안 취약점 분석이 선행되어야 한다. 취약점은 보안공격의 원천이 되므로 발견되지 않은 취약점에 대한 근본적인 대책이 필요하다. 즉, SOA 서비스 취약점을 고려하는 것은 안전하고 신뢰할 수 있는 SOA 기반 시스템을 구축하는 중요한 출발점이 되기 때문이다.

이에 따라 본 논문에서는 SOA 취약점 식별을 위한 모델기반 접근 방법을 제안한다. SOA 계층구조에서 비즈니스 프로세스로부터 보안요구사항을 가지고 도출되는 서비스가 잘 알려진 취약점에 대해 얼마나 취약할지를 예측해 본다. 이로써 SOA 시스템을 위태롭게 하는 서비스의 취약점을 파악할 수 있다. 취약한 서비스는 공격가능성이 높아지고, 결국 취약한 서비스를 제공하게 된다.

본 논문에서는 SOA 환경에서의 서비스 취약점과 모델 기반의 취약점 식별 방법에 초점을 둔다. 이를 위해 모델기반 보안 공학적 접근방법인 UMLsec[3]을 이용하여 보안요구사항 및 취약점 정보를 표현한다. 잘 알려진 취약점 목록에서 SOA 서비스 특성을 고려한 일부 취약점을 선별하여 보안속성과 관계를 파악하여 정량화함으로써 서비스의 취약한 정도를 추정할 수 있다.

본 논문의 구성은 다음과 같다. 먼저 2장 에서는 모델기반 보안공학대한 소개와 SOA 취약점의 특징을 살펴본다. 3장에서 SOA 서비스 계층구조와 서비스 취약점 분석방법을 제안하고, 4장에서 제안된 방법에 대한 분석적 검증이 이루어진다. 마지막으로 5장에서 향후 연구과제와 함께 결론을 맺는다.

2. 관련 연구

2.1 모델기반 보안공학

모델기반 보안공학은 UML을 확장한 UMLsec을 이용하며, 특별한 보안 요구사항을 갖는 소프트웨어를 개발하기 위한 견고한 접근방법을 제공한다. 이 접근방법은 비밀유지, 무결성, 확실성등과 같이 반복되는 보안 요구사항과 시스템 환경에서 요구되는 보안에 대한 전제조건 등이 UML 명세로 기술되거나 소스코드 내에 주석으로 표현된다[4].

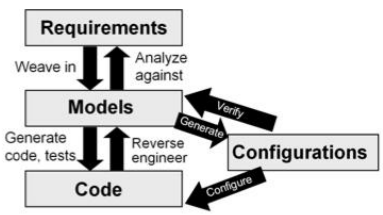

그림 1. 모델기반 보안공학

그림 1과 같이 모델기반 보안공학은 모델기반의 소프트웨어 개발 과정에 통합되어 사용된다. UMLsec에서는 보안에 관련된 여러 관점을 UML 다이어그램을 통해 표현한다. 예를 들면, 보안요구사항을 찾기 위해서 유스케이스 다이어그램을, 안전한 비즈니스 프로세스를 표현하기 위해 액티비티 다이어그램을 사용한다.

UMLsec은 UML이 제공하는 표준 확장 메커니즘을 사용하여 프로파일(profile)의 형태로 정의된다. UMLsec에서 정의되는 연관된 제한조건들은 시스템 설계의 보안측면을 평가할 수 있는 기준을 제공한다. 보안요구사항과 전제조건들을 형식화하기 위해 스테레오타입 ≪stereotype≫과 함께 한 쌍의 {tag, value} 값이 사용된다.

2.2 SOA 취약점

일반적으로 소프트웨어의 취약점은 시스템에 저장 또는 전송되는 정보의 무결성(integrity), 기밀성(confidentiality) 및 가용성(availability)을 위협할 수 있는 정보시스템의 설계 또는 구현상의 오류로 정의된다[5].

SOA는 유연성과 민첩성을 확보하기 위한 구조와 구현기술들의 특성으로 인해 다양한 보안 취약점들이 존재한다. 느슨한 결합 메커니즘이나 XML과 웹기반 기술 및 분산아키텍처에 기인하는 취약점이 대표적이라고 할 수 있다.

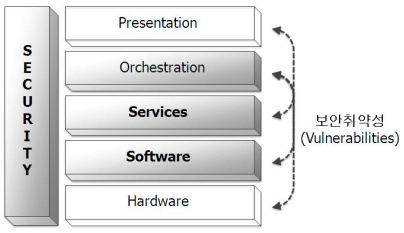

그림 2. SOA 계층과 취약점

그림 2와 같이 SOA 취약점은 서비스 계층의 어디에도 존재할 수 있다[6]. SOA에서 비즈니스 프로세스를 구성하는 행위(activities)는 더 이상 웹 애플리케이션 스크립트에 존재하는 단편적인 코드 섹션이 아니라 WSDL 명세와 SOAP 인터페이스로 구현되는 서비스들이다. 따라서 소프트웨어 애플리케이션의 취약점에 기반을 둔 새로운 취약점이 발생하게 된다. 예를 들면, SOA 서비스 취약점은 공격자가 비즈니스 프로세스의 활동이나 순서를 수정할 수 있게 만든다. 공격자가 사전에 정의된 액션의 내용과 다르게 실행되도록 SOAP 메시지를 변경하는 SOAP 액션 스프핑(action spoofing)을 예로 들 수 있다[7].

이런 문제를 해결하기 위해, 본 논문에서는 SOA 서비스의 취약점을 식별하기 위한 모델기반의 취약점 분석 방법을 제안한다.

3. 모델기반 서비스 취약점 분석

취약점 분석은 소프트웨어 시스템의 분석을 통해 도출된 소프트웨어 보안 위험 요인의 속성과 중요도를 바탕으로 소프트웨어 자산이 근본적으로 가지고 있는 약점을 찾아내고, 취약점이 전체적인 위험에 미칠 수 있는 영향을 분석하는 과정이다. 취약점 분석과정은 크게 취약점 식별과 평가의 두 단계로 이루어진다. 취약점 식별은 유형별로 취약점을 파악하고 평가 단계에서는 파악된 취약점의 수준을 산출하여 우선적으로 고려해야 하는 취약점을 파악하는 것이다.

3.1 보안요구사항의 표현

먼저 [4]에서 이용한 예제를 통해, UMLsec에서 보안 요구사항을 기술하는 방법을 살펴본다.

그림 3. 유스케이스 다이어그램: “fair exchange”

그림 3은 고객이 상품을 구입하는 유스케이스 다이어그램이다. 스테레오타입 ≪fair exchange≫는 “Buys goods”이나 “Sells items”와 같은 액션들은 둘 중 하나가 실행되면 결과적으로 나머지 한 액션도 반드시 실행된다는 것을 의미한다. 유스케이스 다이어그램이 스테레오타입을 사용하여 보안요구사항을 표현한다면, 비즈니스 프로세스의 표기는 액티비티 다이어그램을 사용한다. 보안과 관련된 정보는 한쌍의 태그 값을 사용한다. 그림 4의 {fair exchange, Buys goods}와 {fair exchange, Sells items}가 이에 해당된다.

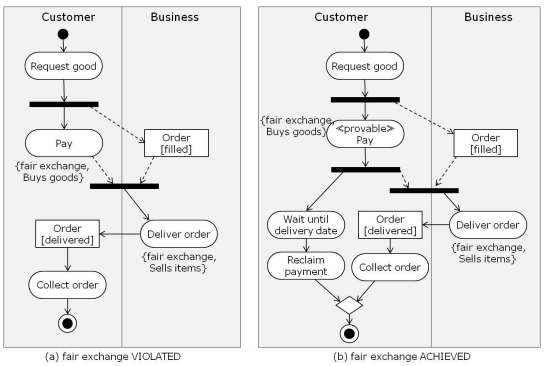

그림 4. 액티비티 다이어그램: “fair exchange”

행위로 이루어진 비즈니스 프로세스를 정의하기 위해 UML 액티비티 다이어그램이나 BPMN (Business Process Modeling Notation)이 주로 사용된다. OASIS의 SOA 참조모델과 참조 아키텍처에 따르면, 이렇게 정의된 프로세스는 비즈니스 프로세스를 기술하고 실행하기 위한 BPEL (Business Process Execution Language)로 자동 변환이 가능하다. 이렇게 BPEL로 표현된 비즈니스 프로세스의 행위들은 WSDL을 이용하여 서비스로 정의된다.

3.2 SOA 취약점 분석

그림 2의 SOA 계층 구조에서 볼 수 있듯이, 비즈니스 프로세스 수준에서 취약점을 분석하는 과정을 통해 서비스의 보안 요구사항이 포함된다. 보안 요구사항을 포함하는 서비스의 추출과 이에 영향을 받는 서비스들의 범위를 파악함으로써, 실제 서비스 도입과정에서 보다 안전한 서비스 선택이 이루어질 수 있다. 또한 이런 보안 요구사항은 이에 상응하는 SLA (Service Level Agreement)를 통해 서비스 제공자와 사용자 사이의 신뢰할 수 있는 계약이 이루어질 수 있다.

본 논문에서 정의하는 취약점 분석은 3단계로 수행된다. SOA를 구성하는 서비스 각각에 대한 보안 속성의 존재여부와 보안 속성으로 인해 발생할 수 있는 취약점 사이의 상관관계를 파악함으로써 서비스들이 어떤 취약점에 노출될 가능성이 있는지를 분석하고 평가할 수 있다.

1단계. 보안속성의 선택

비즈니스 프로세스는 워크플로우로 실행되므로, 어떤 순서로 수행되는 활동들의 집합이라고 할 수 있다. 오케스트레이션 계층은 워크플로우 로직을 포함하며 전형적으로 BPEL 문서로 기술된 파라미터를 가지고 일정한 순서에 따라 특정 서비스를 호출하는 실행엔진으로 구현된다.

비즈니스 프로세스로부터 서비스로 구현되어질 비즈니스 서비스들이 정의되면, 이 비즈니스 서비스들이 프레젠테이션 계층과 오케스트레이션 계층을 구성하는 주요 컴포넌트가 된다.

1단계에서는 비즈니스 서비스들에 대한 보안 속성의 선택이 이루어진다. 이 속성 선택 단계의

결과는 [보안속성 × 서비스] 행렬이다. 이 행렬의 행은 서로 다른 보안속성을 나타내고, 열은 SOA 시스템의 비즈니스 서비스이다. 모든 속성과 서비스에 대해, 행렬의 값은 보안속성이 그 서비스에 존재하는 정도를 나타낸다. 본 논문에서는 (1, 0, -1)의 값을 사용한다. 각 값의 의미는 "그 속성이 존재", "그 속성이 존재하지 않음", 그리고 "그 반대(opposite)의 속성이 존재"함을 나타낸다.

2단계. 보안속성과 취약점 관계 정의

다음 단계는 취약점의 선택이다. 본 논문에서는 새로운 취약점은 주로 기존 취약점의 변형이라고 가정하고, 취약점의 소스로서 공개된 취약점 데이터베이스 중 하나인 BugTraq[8]를 사용한다. 가장 잘 알려진 OWASP[9]의 취약점 목록은 웹어플리케이션에 초점을 맞추고 있기 때문에, SOA 서비스에 적용하기에는 너무 세분화 되어 있다. 예를 들면, BugTraq의 input validation error 클래스는 OWASP에서 SQL injection, encodig errors 등과 같이 여러 클래스로 나뉘어 있다.

BugTraq에서 제공하는 취약점 목록 가운데, 적용범위, 타당성 및 정보의 가용성을 기준으로 일부를 선택하여 영향력을 평가한다. 취약점에 대한 정보가 대부분 불완전하고 일관성이 없기 때문에 우선 정보의 가용성을 기준으로 취약점 목록의 일부를 선택하고, SOA 서비스에 적합한지 또한 적용범위에 해당하는지를 판단하여 54개 취약점 목록을 분석대상으로 정하였다. 이들 취약점은 소프트웨어 취약점에 초점을 두었으며, 잠금장치와 같은 물리적 보안 시스템에 있는 취약점은 제외하고 선택하였다.

이렇게 선택된 취약점 각각에 대해, 취약점에 선택된 각 속성의 영향력을 정의한다. 2단계의 수행 결과는 [취약점 × 보안속성] 행렬로 나타난다. 행렬의 각 셀은 보안속성에 대한 취약점의 영향력을 정량적으로 기술한다. 행렬의 값은 (1, 0, -1)로서 각각 "영향력이 증가", "영향력이 없음", 그리고 "영향력이 감소"를 의미한다.

예를 들면, 전송되는 데이터의 이해와 조작이 용이한 텍스트 기반 커뮤니케이션은 input validation 취약점의 가능성이 증가한다. 그러므로 그 영향력은 "1(영향력이 증가)"이다. 이와 유사하게, 고수준의 프로그래밍 언어를 사용하게 되면, buffer-overflow 취약점의 가능성이 감소하게 된다. 따라서 영향력은 "-1(영향력이 감소)"이다.

3단계. 서비스에 대한 취약점 분석

이 단계의 목적은 서비스에 취약점이 존재할 가능성을 추정하기 위한 것이다. 이전 단계로부터 나온 두 행렬 [취약점 × 보안속성]과 [보안속성 × 서비스]의 결과를 조합해서 추정한다. 이 과정은 같은 가중치를 갖는 단순 선형 조합을 사용한다. 즉, 두 행렬의 곱 [취약점 × 보안속성] × [보안속성 × 서비스]로 계산한다. 이 결과는 서비스에 대한 취약점의 가능성을 기술한 행렬로 [취약점 × 서비스]가 된다. 다음 그림 5는 BugTraq의 7가지 취약점 클래스에 대한 결과 행렬의 예를 보여준다.

	Service 1	Service 2	Service 3	Service 4	Service 5	Serv3ice 6	Service 7
Access Validation Error	0.5	0.4	0.4	1.3	0.3	1.9	1.8
Boundary Condition Error	-1	-2	-0.2	2	2	1.8	1.8
Input Validation Error	2.1	1.2	2.3	3.3	2.2	4.3	4.2
Design Error	1	1	0.3	0.9	1	1.2	1.2
Exceptional Conditions Error	-0.2	-0.7	0.3	2	0.2	2	2
Configuration Error	1	1	-0.2	1.3	1.2	1.3	1.3
Unknown	0	-1	2	3	0.3	4	-1

그림 5. 취약점 분석 결과

그림 5의 열은 비즈니스 프로세스로부터 추출된 비즈니스 서비스들을 나타낸다. 행은 7가지 서로 다른 취약점 클래스이다. 이 행렬의 값은 수학적 확률이 아니라, 가능성에 대한 추론 값이다. 값이 높다면, 해당 취약점에 대해 그 서비스가 취약할 가능성이 높다는 것을 의미한다. 조합의 경우가 아닌 단순 서비스의 경우, 이 행렬의 값들이 각 가장 의미 있는 취약점을 알려줄 수 있다.

이렇게 도출된 취약점 정보들을 바탕으로 서비스와 취약점 사이의 매핑이 얻어지면 이들 링크는 분석대상 시스템의 UMLsec 모델에 포함한다. 그러면 새로운 취약점이 발표될 때마다 모델이 자동적으로 분석될 수 있고, 결과적으로 영향 받는 서비스의 식별이 가능해진다. SOA에서 서비스 조합은 비즈니스 프로세스에 상응하는 BPMN 또는 BPEL이기 때문에, 이 모델 기반 접근방법은 SOA 기반 비즈니스 프로세스를 위한 취약점 식별을 제공한다.

4. 분석 및 평가

그림 5의 취약점 분석 결과에서, 모든 서비스들이 input validation 취약점에서 일관되게 높은 값을 갖는다. 이것은 input validation error가 모든 서비스에 존재할 가능성을 나타내며, 결국 공격의 가능성을 내포한다. input validation에 대한 특성이 [10]에서 정의되었으며, 이에 따라 입력 형식(input format)의 관점에서 분석해 본다.

실제로 SOA 서비스의 느슨한 연결 구조와 조합 가능한 특징은 서비스 처리가 이루어지는 단계마다 input validation을 요구한다. 이것은 서비스의 계층적 특성에 따라 input validation이 서로 다른 계층에서 이루어져야 한다는 것을 의미한다.

현재 웹어플리케이션의 경우 전형적인 input validation error인 SQL-injection을 피하기 위해, Prepared Statement를 통한 지원이 이루어지고 있다. SOA의 구현환경인 웹서비스의 경우, 이와 유사한 취약점은 XPath-injection을 들 수 있다. 그러나 현재 이를 지원하는 표준은 없는 실정이다.

또한 서비스 자체에 대한 취약점뿐만 아니라 그들이 공격당할 수 있는 경로들을 고려하는 것도 매우 중요하다.

현재까지 무결성, 기밀성, 가용성을 기반으로 하는 보안모델의 연구와 국제표준이 있으나, 이들은 모두 메시지 레벨의 보안과 프로토콜에 대한 보안메커니즘을 제공한다. SOA 서비스 계층 구조의 특성을 고려하여 이에 따른 취약점과 취약한 비즈니스 서비스로 인한 영향력은 고려되지 않는다. 따라서 서비스 모델링 과정에서 보안요구사항과 이에 따른 서비스 취약점을 고려한 모델 기반의 취약점 분석이 필요하다.

5. 결론 및 향후 연구방향

본 논문에서는 SOA 시스템의 취약점을 예측하기 위한 분석 방법을 제안한다. 이것은 과거에 나타난 취약점의 목록을 기반으로 이런 유형의 취약점이 나타날 가능성을 평가하는 것이다.

현재까지 SOA 서비스의 취약점에 대한 연구가 거의 이루어지지 않고 있는 실정에서 SOA와 웹서비스의 중요성 때문에 서비스 취약점 분야 연구의 시초로서 의미를 갖는다.

제안된 방법에서 보안속성을 UMLsec을 통해 표현하지만, 실제로 모든 측면에서 시스템을 잘 기술할 수 있는 속성에 대해 정의된 것이 없기 때문에 서비스에 대한 보안속성의 정의가 매우 어렵다는 한계를 가진다.

또한 모든 보안속성에 대해 모든 취약점이 같은 가중치를 갖는다는 가정하에 선형 조합을 했

다. 이것은 분석 과정을 단순화하기 위한 것이지만 모든 속성이 똑같이 중요하다는 한계를 드러낸다. 취약점이나 보안속성에 대해 다른 가중치를 할당하는 방법을 통해 보완함으로써 좀 더 정교한 분석 모델로 개선해 나가야 할 것이다.

그럼에도 불구하고 제안된 분석 모델은 SOA 시스템의 상위 서비스 계층에서 보안을 가장 위협할 것 같은 서비스의 보안속성을 통해 취약점을 추정해 볼 수 있는 방법을 제공한다. 이를 통해 비즈니스 프로세스와 서비스 및 서비스 조합을 통한 취약점의 영향력을 파악할 수 있다.

참 고 문 헌

[1] 조성제, 김동진, “소프트웨어 취약점, 보증 및 보안 테스팅”, 정보과학호지 제30권 제2호, 2012.

[2] 김유경, 도경구, “실행시간 의존성 측정을 통한 SOA 취약성 평가”, 한국전자거래학회지 제16권 제2호, 2011.

[3] Jürjens, J., “Model-based Security Engineering with UML”, Lecture Notes in Computer Science, vol.3655, 2005, pp.42-77.

[4] Höhn, S., Lowis, L., Jürjens, J., Accorsi, R., “Identification of Vulnerabilities in Web Services using Model-based Security”, Web Services Security Development and Architecture: Theoretical and Practical Issues, 2010, pp.1-31.

[5] Bishop, M., “Introduction to computer security”, Addison-Wesley, 2005.

[6] Arsanjani, A., Zhang, L. J., Ellis, M., Allam, A., Channabasavaiah, K., “Design an SOA solution using a Reference Architecture,” IBM Developer Works, 2007.

[7] Lowis, L., Accorsi, R., “Vulnerability Anslysis in SOA-Based Business Processes”, IEEE Transactions on Service Computing, Vol.4, No.3, 2011, pp.230-242.

[8] Security Focus. BugTraq mailing list at http://www.securityfocus.com/archive/1.

[9] OWASP. The Open Web Application Security Project. http://www.owasp.com/.

저 자 소 개

김현하

2003년 한양대학교 전자컴퓨터공학부 (학사)
2005년 한양대학교 컴퓨터공학과 (석사)
2008년 한양대학교 컴퓨터공학과 (박사과정 수료)
2006년 - 현재 한양대학교 컴퓨터공학과
<주관심분야 : 프로그래밍언어,프로그램 분석, 소프트웨어 보안>

김유경

1991년 숙명여자대학교 수학과 (학사)
1994년 숙명여자대학교 전산학과 (석사)
2001년 숙명여자대학교 컴퓨터과학과 (박사)
2001년-2005년 숙명여자대학교 정보과학부 초빙교수
2005년-2006년 미국 UC Davis, Post-doc.
2006년- 현재 한양대학교 ERICA 컴퓨터공학과 연구교수
<주관심분야 : 소프트웨어품질, SOA, 웹서비스, 소프트웨어보안>

도경구

1980년 한양대학교 산업공 학과 (학사)
1987년 미국 아이오와주립대학교 전산학과 (석사)
1992년 미국 캔사스주립대학교 전산학과 (박사)
1995년 - 현재 한양대학교 ERICA 컴퓨터공학과 교수
<주관심분야 : 프로그래밍언어, 소프트웨어 보안, 프로그램 분석>