논문 2011-2-4

클래스 분리와 동적 적재를 사용한 불법 복제 Android 앱 실행 차단

문재찬*, 김동진*, 조성제*, 박민규**

Blocking Pirated Android App Execution using Class Separation and Dynamic Loading

Jaechan Moon*, Dong-Jin Kim*, Seong-Je Cho*, Minkyu Park**

요 약

스마트폰의 수요 확대에 따라, 스마트폰 앱 시장이 사회, 경제적으로 주목받고 있다. 하지만 유료 앱의 불법 복제 및 유포, 사용에 따른 피해가 급증하면서, 앱의 정상 구매 및 사용을 유도하기 위한 다양한 기술들이 요구되고 있다. 현재 앱 불법 사용을 차단하기 위해 Google의 LVL, SKT의 ARM 등의 사용자 인증 기반 앱 DRM 기술들이 연구 및 서비스 되고 있다. 스마트폰 앱은 역분석 및 변조가 쉽기 때문에 사용자 인증 과정만 추가한 이러한 앱 DRM 기술은 쉽게 우회할 수 있는 문제가 있다. 본 논문에서는 기존 기술의 문제점을 해결하기 위하여 클래스 분리와 분리된 클래스의 동적 로딩을 함께 사용하는 기법을 제안한다. 제안 기술은 앱 개발 시에 특정 클래스를 분리하고, 분리한 클래스를 별도로 컴파일 하여 서버에 보관한다. 앱이 실행될 때마다 사용자 인증 후, 서버로부터 분리된 클래스를 전송받고, 동적으로 적재하여 수행한다. 실험 결과, 기존 앱 DRM 기술에 비해 역분석이 어렵고, 발생한 오버헤드가 스마트폰 환경에 적용 가능한 범위에 있음을 보였다.

Abstract

Smartphone app market is getting important socially and economically. As damage of app piracy increases, we require a technique to encourage purchase and use of legal apps. Digital Rights Management (DRM) for apps such as Google's LVL and SKT's ARM is studied and used. They are techniques based on user authentication. But they are easy to circumvent because Smartphone apps are weak to reverse engineering and modification. In this paper, we propose a technique that enforces the existing techniques to solve the above problems. This technique use class separation and class dynamic loading. The technique separates a particular class when developing apps and servers must keep it. After user authentication on every run, the separated class is transferred from the server and dynamically loaded. Experimental results show the technique is feasible on Smartphone environment.

한글키워드 : 불법 복제 앱, 실행 차단, 클래스 분리, 동적 적재, Android

* 단국대학교 컴퓨터과학과
** 건국대학교 컴퓨터공학과
교신저자:조성제(email:sjcho@dankook.ac.kr)
접수일자: 2011.10.15 수정완료: 2011.12.15

※ 본 연구는 문화체육관광부 및 한국저작권위원회의 2011년도 저작권 기술개발사업의 연구결과로 수행되었고, 2011년 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행되었음(2011- 0026301)

1. 서 론

2011년 3월 기준 Apple iPhone의 앱 개수는 35만개, Google Android 앱은 25만개를 넘어서는 등, 스마트폰의 보편화와 함께, 스마트폰 앱 시장이 빠르게 성장하고 있다[1,2,3]. 하지만, 이중 적발된 불법 복제 및 유포 앱의 개수만 약 2만 3천개이고, 실제 더 많은 앱이 블랙마켓이나, P2P, SNS를 통해 불법적으로 유포되고 있다. 불법 복제 및 유포에 따른 피해액은 해마다 증가하여, 2010년에는 3천억 원을 넘었고, 이에 따라 유명 앱 개발사들은 앱의 유료 판매로 인한 수익을 포기하고, 광고 수익을 택하고 있다[4,5].

이러한 문제를 개선하기 위해, 앱에 대한 DRM 기술이 연구되고 있다. Google에서는 LVL(License Verification Library)[6], SKT에서는 ARM(Application DRM)[7] 등의 보호 기술을 제공하고 있다. 앱 DRM 기술은 정상 구매자 인증 과정을 통해 불법 복제 앱의 실행을 차단하는데, 스마트폰 앱의 경우 PC 소프트웨어에 비해 역분석 및 변조가 용이하기 때문에 정상 구매자 인증 과정만 우회하면 이를 무력화시킬 수 있는 문제가 있다[8]. 앱 전체를 암호화하고, 실행 시 복호화 하는 방법[9]은 암호화된 앱이 파일로 존재하기 때문에 언제든 분석될 수 있다는 단점이 있다. 이러한 문제를 개선하고, 앱 불법 유포 및 사용을 차단할 수 있는 새로운 방법에 대한 연구가 필요하다.

본 논문에서는 앱 불법 유포 및 사용을 차단하기 위한 기존의 사용자 인증 기반 앱 DRM 기술에 클래스 분리와 분리된 클래스의 동적 적재를 함께 사용하는 불법 복제 앱 실행 차단 기법을 제안한다. 이 기법에서는 앱 배포 시 하나의 지정된 클래스를 분리시켜 앱을 배포한다. 해당 앱을 실행시킬 때 기존 DRM 기반 방법과 같이 정상 사용자 인증 과정을 수행한다. 그런 후에 분리시킨 클래스를 온라인을 통해 전송받고 이를 동적으로 적재하여 앱을 실행한다. 앱의 실행이 종료될 때 메모리상의 분리된 클래스를 삭제한다. 실행 시에만 전송받아 동적으로 적재하여 실행시킨 후 삭제하기 때문에 분리된 클래스가 파일 시스템에 존재하지 않는다. 따라서 앱이 불법 복제되어 유포되더라도 앱을 정상적으로 실행할 수 없다. 어떤 클래스가 분리될 것인지를 개발자가 결정하기 때문에 역분석 및 변조가 어려워진다. 또한 제안 기법을 앱 불법 복제 문제가 심각한 Android 기반에서 Android 플랫폼 수정 없이, 사용자 수준에서 구현하여 범용성을 높였다.

결과적으로 클래스 분리와 동적 적재를 위한 코드 추가로 원래 앱에 비해 약 0.6%의 앱 크기 증가와 평균 약 1초의 수행 시간 증가를 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 앱 불법 복제 방지 기술에 대해 설명하고, 3장에서는 본 논문에서 제안하는 기법인 클래스 분리와 동적 적재를 통한 불법 복제 앱 실행 차단 기법에 대해 설명한다. 4장에서는 기존 앱과 본 논문이 제시하는 기법이 적용된 앱의 성능을 비교 분석하고, 5장에서 결론과 향후 연구에 대해 서술한다.

2. 관련 연구

PC 환경에서는 다양한 온라인 실행 코드 기술들이 존재한다. Java의 RMI(Remote Method Invocation)는 서로 다른 자바 가상 머신 상에 있는 객체의 메서드를 호출하는 기술이다[10]. MSRPC(Microsoft Remote Procedure Call)는 네트워크상의 다른 시스템에 위치하고 있는 프로그램에 함수 호출 등의 서비스를 요청하고, 반환

받는 기술이다[11]. 하지만 RMI, MSRPC는 IPC(Interprocess Communication)에서 발전된 기술로 네트워크의 다른 시스템에 함수 호출 및 서비스를 요청할 때마다 네트워크를 사용하기 때문에 스마트폰 등의 모바일 환경에는 부적합하다.

패킹(Packing)이라 불리는 실행 압축 기술은 네트워크를 사용하지 않고, 프로그램을 보호하는 기술로써, 프로그램의 일부 및 전체를 압축하거나 암호화 등을 사용하여 인코딩(encoding)하고, 프로그램 실행 시에 동적으로 디코딩(decoding)하여, 불법 복제 및 역분석을 어렵게 한다[12]. 하지만 프로그램 전체가 로컬 시스템에 존재하기 때문에 실행 기반의 동적 분석 방법이나, 메모리 덤프(dump)에 의해 역분석될 수 있다.

3. 앱 불법복제 차단 시스템

본 논문에서 제안하는 불법 복제 앱 실행 차단 기법은 클래스 분리, 사용자 인증 및 클래스 전송, 클래스 동적 적재로 구성된다.

3.1 클래스 분리

Android 앱은 Java 언어로 개발되고, 컴파일 시에 Android 앱의 바이트코드와 패키지 형태인 DEX와 APK로 변환된다. 앱 개발 과정 중 클래스 분리 단계에서는 배포용과 서버 보관용의 2개의 APK 파일을 생성한다. 앱 실행 시에만 전송 받아 동적으로 적재할 클래스를 분리하며, 분리된 클래스를 전송 및 동적 적재하기 위한 스텁(stub)을 추가 한다.

Java의 클래스 단위로 앱을 분리하는 것이 효과적이다. 분리된 클래스를 전송할 때 데이터 크기가 작을수록 효율적이기 때문에 클래스 보다 더 작은 메서드(method)나, 명령어 단위로 분리하는 것이 네트워크 사용 및 동적 적재 속도 측면에서 유리할 수 있지만, 앱의 복잡한 호출 구조나 명령어 간의 의존성 등에 문제가 발생할 수 있기 때문에 클래스 단위로 분리하였다. 분리된 클래스를 별도의 APK로 컴파일 하는 과정은 일반적인 앱 컴파일 과정과 동일하다.

배포용 APK와 서버 보관용 APK를 마켓에 등록한다. 실제로 판매/배포되는 앱은 배포용 APK 파일이며, 서버 보관용 APK 파일은 실제로 마켓에서 판매/배포 하지 않고 별도로 관리하여 앱을 구매한 사람들이 앱을 수행할 때마다 전송된다.

3.2 사용자 인증 및 클래스 전송

배포용 앱에 추가될 스텁은 사용자 인증 및 서버와의 통신, 동적으로 클래스를 적재하는 기능을 수행한다. 사용자 인증은 기존의 앱 DRM 기술을 그대로 적용할 수 있다. 사용자 인증 요청에 대한 응답과 함께 서버 보관용 APK에 포함되어 있는 클래스를 클라이언트에게 전송하기 때문에 사용자 인증 방법으로는 외부의 사용자 인증 서버를 사용하는 방식이 로컬에서 사용자 인증을 하는 방식에 비해 효율적이다.

사용자 인증에 실패하거나, 불법 변조를 통해 인증 방법을 우회하면 분리한 클래스를 전송 받을 수 없기 때문에 앱을 정상적으로 실행할 수 없으므로 기존 방법 보다 안전하다.

전송 받은 클래스는 앱이 수행되는 동안 임시로 Android 파일시스템에서 '/data/data/'에 저장하는데, 일반 사용자가 접근할 수 없는 영역이기 때문에 낮은 수준의 역분석으로부터 안전하다.

3.3 클래스 동적 적재

배포용 앱은 동적 적재 과정에서 APK에서 DEX를 추출하고, 분리된 클래스를 동적으로 로

드한다. APK 파일로 존재하는 분리된 클래스를 현재 실행 중인 앱에 동적으로 적재하기 위해서, Android SDK에서 제공하는 DexClassLoader 클래스를 사용한다. DexClassLoader의 생성자와 loadClass() 메서드를 통해 동적으로 분리된 클래스와 사용할 메서드를 정의하여 호출한다.

클래스 동적 적재를 위해 Android 플랫폼의 클래스 적재 부분의 코드를 수정하지 않고, SDK에서 제공하는 클래스를 사용하기 때문에 플랫폼 버전에 관계없이 사용 가능하다.

마지막으로 앱 종료 시, 전송 받은 분리된 클래스 파일을 삭제하여, 역분석 및 불법 변조를 통한 앱 불법 복제를 차단한다.

4. 실험 및 분석

스마트폰 환경에서는 저장 공간의 사용량과 소비 전력을 줄이는 것이 중요하기 때문에 논문의 제안 기법이 스마트폰 앱에서 사용 가능 여부를 검증하기 위해, 기법이 적용된 앱과 미적용 앱의 실행 속도와 앱 크기 차이를 측정하였다.

실험을 위해 Android 클라이언트 시스템은 Core2Duo 2.00GHz, 3GB RAM, Windows 7 32 bits의 호스트 시스템에 1 프로세서, VMware 가상머신(1GB RAM, Unbuntu 10.04)에서 512MB RAM을 할당한 에뮬레이터를 사용하였고, Android 2.3.3 버전을 사용하였다. 사용자 인증 및 분리된 클래스 관리를 위한 서버는 Core2Quad 2.66GHz, 8GB RAM, Windows Server 2008 R2 시스템에서 실행하였다.

클라이언트 시스템은 실제 스마트폰 환경에 가깝게 하기위해, 54Mbps의 무선 네트워크를 사용하였고, 서버는 1Gbps 유선 네트워크를 사용하였다.

4.1 실험 시나리오

앱을 마켓에 등록하고 구매하는 과정은 피할 수 없는 과정이며 앱 실행 시의 성능에 영향을 미치지 않으므로 실험 시에 이 과정은 이미 진행되었다고 가정하였다. 또한 클래스 분리 및 스텁 추가는 수동으로 적용하였으나 향후 일반 개발자들이 사용할 수 있도록 라이브러리 형태로 제공될 예정이다.

사용자 인증을 위해, 클라이언트에서 USIM 번호와 전화번호를 서버로 전송하고, 서버에서 등록된 사용자 정보와 비교하도록 하였다. 기존의 앱 DRM은 사용자 인증 프로세스를 정확히 알 수 없고, 분리된 클래스 관리를 추가할 수 없기 때문이다.

제안 기법을 적용한 앱은 Google 마켓에 등록된 위치 기반 서비스를 제공하는 '수소문'이다. 개발자의 동의를 구하여 소스코드를 제공받아서 사용하였다. 총 6개의 사용자 정의 클래스로 구성된 해당 앱의 클래스 중 다른 사용자 정의 클래스들을 참조하지 않는 클래스 1개를 분리하여 실험하였다.

4.2 실행 결과 비교

정상 앱 구매자가 앱을 실행했다고 가정하고, 사용자 인증 이후, 분리된 클래스를 서버로부터 전송받아 동적 적재된 경우, 그림 1의 좌측 화면과 같이 정상 실행되었다. 하지만, 불법 복제 앱 사용자의 경우, 인증을 우회하거나 실패하여 그림 1의 우측 화면과 같이 앱 정상 실행에 실패하였다. 즉, 인증과정 우회만으로는 본 논문에서 제안하는 불법 복제 앱 실행 차단 기술을 우회하기 어렵다는 것을 의미한다.

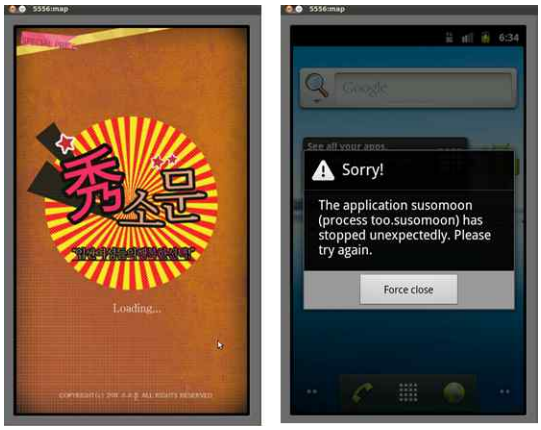

그림 1. 사용자 인증 성공 여부에 따른 앱 수행 결과(좌: 성공, 우: 실패)

Figure 1. Result of an app execution based on success or failure of user authentication (Left picture: Success, Right picture: Failure)

4.3 파일 크기 비교

제안 기법 적용 전, 후 앱 파일의 크기 변화는 <표 1>과 같다.

<표 1> 제안 기법 적용 전, 후 앱의 파일 크기 비교

Table 1. Comparison of file sizes before and after applying our proposed scheme

	적용 전 (Bytes)	적용후 (Bytes)	증가량 (Bytes)	증가율(%)
앱	3,141,932	3,144,919	+2,987	100.1
수신한 APK	-	13,417	+13,417	-
추출한 DEX	-	3,480	+3,480	-
전체	3,141,932	3,161,816	+19,884	100.6

제안 기법 적용 전과 후의 앱 전체 크기는 약 0.6% 증가하였다. 제안 기법을 적용한 후에는 분리된 클래스가 APK로 컴파일되고, 앱 실행 시에 서버로부터 APK 형태로 전송 받기 때문에 DEX 파일 이외의 APK 패키지를 구성하는 파일들이 추가되기 때문이다. 또한, 동적 클래스 적재 과정 중에 APK로부터 DEX를 추출하고 클래스를 적재하기 때문에 동일한 APK에 포함된 DEX와 추출된 DEX가 중복으로 존재하기 때문이다. PC에 비해 저장 공간이 작은 스마트폰 환경에서도 0.6%의 크기 증가는 성능 저하로 작용하지 않는다.

서버로부터 전송받은 APK와 DEX를 고려하지 않고, 기법 적용 전, 후의 앱 파일 크기를 비교하면 적용 후의 앱 크기가 약 0.1% 증가했다. 기법을 적용하면 분리한 클래스만큼의 앱 크기가 감소하지만 사용자 인증, 클래스 파일 전송 및 동적 적재에 필요한 스텁이 추가되기 때문이다. 하지만 이는 앱에서 분리한 클래스의 크기가 스텁에 비해 작기 때문이고, 클래스의 선택에 따라 기법 적용 후의 앱 크기가 달라질 수 있다는 것을 의미하며, 분리된 클래스의 크기가 크다면 오히려 기법 적용 후의 크기가 더 작을 수 있다.

4.4 실행 시간 비교

앱 실행 시간을 기준으로 비교하기 위해 다음과 같이, 앱의 실행 과정을 구분하였다.

- 앱 실행 *초기화*: 앱 실행 초기화 과정
- *인증*: 스마트폰 정보를 이용하여 서버에 사용자 인증을 요청하고, 서버에서 인증을 처리하는 과정
- 클래스 *전송*: 인증에 성공한 경우, 서버로부터 컴파일된 클래스(APK)를 전송 받는 과정
- 클래스 *동적 적재*: 전송받은 APK에서 DEX를 추출하고, 동적으로 클래스를 적재하는 과정
- 앱 실행: 동적 클래스 적재 이후의 나머지 앱 실행 과정

제인 기법은 서버에게 인증을 요청하면 서버에서 인증 확인 후, 클래스를 전송하지만, 본 실험에서는 인증 과정보다 클래스 전송에 걸리는 시간이 더 중요하기 때문에 인증과 클래스 전송 시간을 별도로 구분지어서 측정하였다.

<표 2> 앱의 전체 실행 시간 (단위: 초)
Table 2. Total execution time of apps (Unit: sec)

횟수	미적용 앱 (원본 앱)	제안 기법 적용 앱				
		초기화	인증	전송	동적 적재	전체
1	1.272	1.06	0.035	0.485	0.349	1.929
2	1.809	0.141	0.011	0.469	0.64	1.261
3	0.18	0.459	0.032	0.456	0.489	1.436
4	0.211	0.662	0.099	0.47	0.278	1.509
5	0.018	0.321	0.052	0.487	0.531	1.391
6	0.24	0.219	0.04	0.471	0.55	1.28
7	0.153	0.348	0.161	0.349	0.596	1.454
8	0.191	0.298	0.161	0.46	0.259	1.178
9	0.172	0.214	0.018	0.468	0.411	1.111
10	0.211	0.355	0.011	0.47	0.349	1.185
11	0.153	0.348	0.035	0.456	0.681	1.52
12	0.169	0.36	0.038	0.461	0.304	1.163
13	0.3	0.279	0.129	0.462	0.312	1.182
14	0.19	0.438	0.084	0.467	0.471	1.46
15	0.141	0.37	0.031	0.469	0.333	1.203
16	0.201	0.158	0.09	0.48	0.499	1.227
17	0.177	0.22	0.049	0.462	0.479	1.21
18	0.212	0.168	0.131	0.471	0.489	1.259
19	0.084	0.231	0.118	0.451	0.371	1.171
20	0.251	0.449	0.02	0.463	0.306	1.238
평균	0.317	0.355	0.067	0.461	0.435	1.318

실험결과 <표 2>와 같이, 제안 기법을 적용하지 않은 앱에 비해, 기법을 적용한 경우의 앱 실행시간이 평균 약 1초 증가했다. 네트워크를 통해 서버로부터 클래스를 전송 받는 단계의 오버헤드가 전체 실행 시간 중 가장 크고, 다음으로 전송받은 클래스의 동적 적재 과정의 오버헤드가 큰데, 동적 적재에 걸리는 시간은 클래스의 불리한 클래스의 크기에 따라 달라질 수 있다. 또한 에뮬레이터보다 실제 스마트폰 디바이스의 연산처리 및 앱 실행 속도가 더 빠르기 때문에 실제 디바이스에서는 더 좋은 성능을 기대할 수 있다.

5. 결 론

스마트폰의 보편화와 함께, 스마트폰 앱 시장이 빠르게 성장하고 있지만, 유료 앱의 불법 복제 및 유포, 사용이 큰 사회적, 경제적 문제로 떠오르고 있다. 앱 불법 사용을 차단하기 위해, 앱 앱 DRM 기술들이 연구 및 서비스되고 있다. 하지만 앱 DRM 기술은 역분석 및 변조가 용이하기 때문에 정상 구매자 인증과정만 우회하면 이를 무력화시킬 수 있는 문제가 있다.

본 논문에서는 앱 불법 유포 및 사용을 차단하기 위한 기존의 사용자 인증 기반 앱 DRM 기술에 클래스 분리와 분리된 클래스의 동적 적재를 적용한 불법 복제 앱 실행 차단 기법을 제안하였다. 이 기법은 기존 앱 DRM 기술에 비해 역분석이 어렵고, 스마트폰 환경에서 적용 가능한 범위의 오버헤드를 보였다.

향후, 본 연구를 발전시켜, 앱 불법 유출자 추적 기술과 스마트폰 앱 변조방지 기술 등에 대해 연구할 계획이다.

참 고 문 헌

[1] Androlib, “Estimated number of Applications downloaded in the Android Market”, 2011, “http://www.androlib.com/appstats.aspx”.

[2] 한국콘텐츠 진흥원, “대한민국 게임백서”, 2009.
[3] J. Yarow, K. Angelova, “Chart of the day: Google is closing the gap on Apple’s App store”, Business Insider, Mar. 2011, “http://www.businessinsider.com/chart-of-the-day-smartphone-apps-2011-3”
[4] 유주희, “안드로이드 앱 불법복제 성행, 개발자들 ‘시장 죽는다’ 한숨”, 서울경제, 2010.10.28.
[5] 한민옥, “스마트폰 불법복제 앱 뿌리 뽑는다” 디지털타임스, 2011.10.17.
[6] 구글 LVL, “http://developer.android.com/guide/publishing/licensing.html”.
[7] SKT ARM, “http://dev.tstore.co.kr/devpoc/download/downloadApplicationDRM.omp?ctgrCd=applicationDRM&menuId=1”.
[8] 홍석희, “구글, 새 라이선스 시스템도 뚫렸다, 저작권 관리 ‘몸살’”, 파이낸셜 뉴스, 2010.8.24.
[9] 김희문, “DEX 파일 암호화를 통한 안드로이드 애플리케이션 보호 프레임워크”, 한양대학교 석사학위 논문, 2011.2.
[10] Microsoft RPC, “http://en.wikipedia.org/wiki/Microsoft_RPC”.
[11] Java RMI, “http://en.wikipedia.org/wiki/Java_RMI”.
[12] M. Kim, J. Lee, H. Chang, S. Cho, Y. Park, M. Park, and Philip A. Wilsey, "Design and Performance Evaluation of Binary Code Packing for Protecting Embedded Software against Reverse Engineering", ISORC, May 2010.

저 자 소 개

문 재 찬

2011년 단국대학교 컴퓨터과학과 학사 졸업
2011 – 현재 : 단국대학교 컴퓨터학과 컴퓨터과학 전공 석사과정

<주관심분야 : 웹 보안, 스마트폰 앱 보안>

김 동 진

2009년 단국대학교 컴퓨터과학과 (이학사)
2011년 단국대학교 컴퓨터학과 (공학석사)
2011년 단국대학교 컴퓨터학과 박사과정

<주관심분야 : 컴퓨터보안, 소프트웨어 보증, 시스템소프트웨어 등>

조 성 제

1989년 서울대학교 컴퓨터공학과 (공학사)
1991년 서울대학교 컴퓨터공학과 (공학석사)
1996년 서울대학교 컴퓨터공학과 (공학박사)
2001년 미국 University of California, Irvine 객원연구원
2009년 미국 University of Cincinnati 객원연구원
1997년 3월 ~ 현재 : 단국대학교 소프트웨어학과 교수

<주관심분야 : 컴퓨터보안, 소프트웨어 보증, 시스템소프트웨어, 실시간스케줄링, 임베디드 소프트웨어 등>

박 민 규

1991년 서울대학교 컴퓨터공학과 (공학사)
1993년 서울대학교 컴퓨터공학과 (공학석사)
2005년 서울대학교 컴퓨터공학과 (공학박사)
2006년 9월 ~ 현재 : 단국대학교 컴퓨터학부 교수

<주관심분야 : 운영체제, 실시간 스케줄링, 베디드 시스템, 컴퓨터보안 등>