침입탐지시스템 탐지성능 향상 위한 해시기반 패턴 매칭 시스템

김병훈* · 하옥현** · 신제철*

요 약

네트워크 대역폭과 침입기술의 발달하는 상황에서 침입탐지 시스템의 패턴 매칭 방식으로는 대용량화된 모든 패킷을 기존의 침입탐지 시스템의 패턴 매치 방식으로 패턴을 분석하는 것에는 한계가 발생한다. 패킷들이 단편화되어 수신될 때 패킷들을 효율성 있게 탐지하기 위해서 Esnort (1)와 같은 운영체제에 일치되는 패킷들의 패턴만 매치하는 방법이 제시되었다. Esnort의 기본 매커니즘인 NMAP을 이용하여 동일한 네트워크의 시스템의 운영체제를 스캔하여 스캔된 정보와 수신된 패킷과 동일한 운영체제만을 선별하여 패턴 매치를 적용하여 패턴 매치의 성능을 개선하였다. 하지만 운영체제의 종류가 다양해지고 nmap의 운영체제 식별의 오류로 수신된 패킷이 무시되어 인입되는 경우가 발생할 수 있다. 본 연구에서는 유동적인 사용자의 시스템 환경과 독립적으로 침입탐지 시스템의 패턴의 해시화를 통해 해시테이블을 생성하여 패턴 매치의 시간을 단축는 개선된 침입탐지 시스템을 제시하고 검증하고자 한다.

Hash-based Pattern Matching System for Detection Performance

Byung Hoon Kim* · Ok Hyun Ha** · Jae Chul Shin*

ABSTRACT

In the environment of development of network bandwidth and intrusion technology there is limit to the pattern analysis of all massed packets through the existing pattern matching method by the intrusion detection system. To detect the packets efficiently when they are received fragmented, it has been presented the matching method only the pattern of packets consisting with the operation system such as Esnort. Pattern matching performance is improved through the use of NMAP, the basic mechanism od Esnort, by scanning the operation system of the same network system and appling pattern match selectively scanned information and the same operation system as the received packets. However, it can be appeared the case of disregarding the receivied packets depending on the diversity of the kind of operaation systems and recognition mistake of operation system of nmap. In this paper, we present and verify the improved intrusion detection system shortening the pattern matching time by the creation of hashy table through the pattern hash of intrusion detection system independently with the users system environment .in the state of flux.

Key words : IDS, Snort, Hash Table

접수일 : 2009년 10월 10일; 채택일 : 2009년 12월 12일

* 경기대학교 산업보안학과

** 교신저자, 호남대학교 경찰법행정학부

1. 서 론

네트워크를 비롯한 정보시스템의 의존도가 높아지며, 많은 정보처리를 온라인을 통해 처리하고 공유하는 환경이 자리 잡혀 감으로써 업무의 효율성을 증대 시키는 이점이 있다. 네트워크를 이용하여 공유함으로써 원하지 않는 자가 정보를 취득, 변조하여 업무의 효율성을 높이는 것의 부작용으로 해킹, 서비스 거부 공격, 인터넷웜, 이메일 바이러스, 피싱 등의 악의적인 공격도 빠른 속도로 증가하고 있다. 2009년 7월 7일 우리는 수많은 악성 봇을 이용하여 공격하는 DDoS를 경험함으로써 많은 정보 서비스를 이용하는데 불편함을 느꼈다. 악성 봇의 공격을 미리 예상하고 관리자가 정확하게 공격을 예상 하였다면 공격을 시작하기 전에 기술적으로 처리를 할 수 있는 시간적 여유를 가질 수 있었을 것이다. 침입을 탐지하는 방법에는 통계적인방법, 특징 추출, 예측 가능 패턴 생성, 신경망시스템, 전문가 시스템, 키 입력시스템, 모델에 근거한 침입탐지 시스템, 상태전이 분석등 이와 같이 다양한 침입탐지 시스템들이 존재한다. 패턴의 비교 분석을 통하여 악성 패킷들이 가지는 패턴을 분석하여 데이터베이스에 저장된 패턴과 매치 작업을 하는 방식의 침입탐지 시스템이 많이 이용되고 있다. 대용량의 데이터가 단편화되어 네트워크를 통해 패킷으로 유입될 때 침입탐지 시스템은 패킷이 가진 특정 패턴을 인식하여 기존의 존재하는 패턴들과 매칭과정을 통하여 일치하는 경우 침입으로 알람을 보낸다. 하지만 대용량의 데이터, 다양한 네트워크 서비스, 정보시스템의 규모가 커짐에 따라서 모든 패킷을 스니핑하여 침입탐지시스템에서 모든 패킷을 검사하는 것은 침입탐지의 성능을 저하시키는 원인이 되며 기존의 수많은 악성 패킷의 패턴들을 처음부터 마지막까지 매칭작업을 시행하기까지 시간이 많이 소요된다, 대표적인 오픈 침입탐지 시스템인 Snort가 가지는 탐지 룰은 48개가 있으며 이 룰안에는 적게는 10개에서 최고 630개 이상의 패턴을 가지고 있다, 하나의 패킷의 패턴을 비교 분석할 때 최고 630개의 패턴을 비교하는 경우에 침입탐지 시스템의 성능이 저하되며 시간이 흐를 수 록 패턴이 많아짐에 따라서 침입탐지 시스템의 하드웨어적인 성능 개선이 불가피하게 된다. 본 연구에서는 하드웨어의 성능 개선 없이 침입탐지 시스템에서 사용되는 패턴 매치 방법을 해시화된 테이블을 이용하여 패턴의 탐지하는 횟수를 줄임으로써 수많은 패킷들의 패턴을 짧은 시간에 매치하여 탐지 할 수 있는 시스템을 제안하여 기존의 패턴 매칭 방법과 비교하고자 한다.

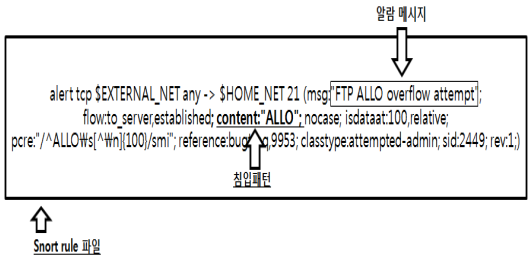
2. 관련연구

기존에 알려진 공격에 대해 침입탐지 시스템에서는 유입된 패킷의 패턴을 조사한다. 이러한 패턴 매칭을 위해서는 공격이 나타내는 패턴에 대해 정확하게 기술된 내용이 존재하여야 한다. Snort의 룰 파일에는 각 서비스 별로 라인단위의 패턴을 가지고 있으며 Snort에 패킷이 유입되면 서비스의 에 맞는 룰 파일에 있는 패턴들과 매칭과정을 거쳐 침입여부를 판단한다. 아래와 같이 Snort의 룰파일이 가지는 패턴라인이 48개의 룰파일중 48라인 이상의 패턴을 가지는 룰 파일의 이름과 가지는 패턴 수를 나타낸다.

〈표 1〉 룰이 가지는 패턴의 수

netbios.rules 630	oracle.rules 305
deleted.rules 243	rpc.rules 128
web-php.rules 127	smtp.rules 75
web-client.rules 49	backdoor.rules 78
web-cgi.rules 358	web-misc.rules 123
web-iis.rules 119	exploit.rules 106
ftp.rules 76	misc.rules 68
icmp-info.rule 93	sql.rules 48

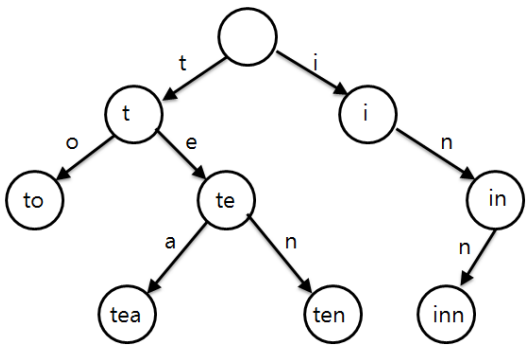
(그림 1)은 Snort가 가지는 Rule 파일의 패턴이다.



(그림 1) 룰 파일에 존재한 패턴

2.1 Aho-Corasick 알고리즘

Aho-Corasick 알고리즘[2]은 과거 침입탐지 시스템을 위해 고안된 알고리즘으로 기존에 존재하던 텍스트 기반 검색 알고리즘을 혼합하고 수정하여 검색하고자 하는 대상이 여러 개일 때 빠른 검사 및 중복성 검사 등을 해결하기 위해 수정된 알고리즘이다. 비교하려는 패턴을 등록하는 과정에서 이를 컴파일하여 내부적으로 트리구조를 구성, 트리구조의 매칭을 진행하여 성능을 개선하고자 하였다. 또한 패턴간의 상관관계에 주목하여 이들의 상관관계를 이용하여 트리구조 내에서도 패턴 간에 연관성을 가질 수 있도록 하였고, 매칭이 진행되는 과정에서 트리의 부분 매칭 결과를 저장하여 재사용함으로써 성능 향상을 보였다. (그림 2)는 패턴들을 검사하고자 할 경우 Aho-Corasick 알고리즘이 이루는 트리 구조 및 매칭 구조를 나타낸다. (그림 2)에서 알 수 있듯이 Aho-Corasick 알고리즘은 트리구조를 통해 매칭 속도와 적용 가능한 룰의 개수를 비약적으로 증진시키는 결과를 보였지만 이는 기본적으로 각 패턴에 상당한 연관관계를 가져야만 효과를 낼 수 있다. 즉 패턴 간의 상관관계에 굉장히 종속적인 성능을 나타내는 알고리즘으로 불특정 패턴을 사용하게 되는 실제 네트워크에서 시스템의 성능에 대한 예측을 어렵게 하고, 확장성에 불리한 요인으로 작용한다. 또한 룰 내의 패턴간 상관관계가 우수하여 Aho-Corasick 알고리즘이 제 성능을 발휘하는데 적합하지만, 매칭 알고리즘 자체의 특성상 한 바이트의 문자 당 한 번의 메모리 접근이 필요하며, 한 번에 단 한 바이트만을 매칭 할 수 있다는 점은 결국 메모리 접근과 처리에 병목현상을 불러일으키며, 이로 인한 성능 저하를 피할 수 없다[3].



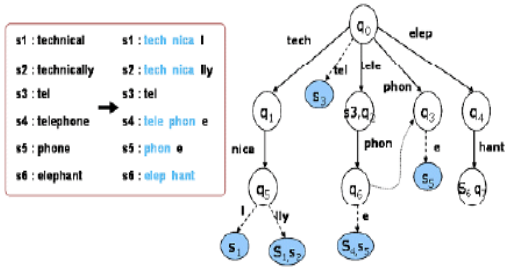
(그림 2)Aho-Corasick 알고리즘

2.2 Jump Aho-Corasick 알고리즘

기존의 Aho-Corasick 알고리즘이 트리구조를 통해 성능을 개선한 반면 트리구조를 한 바이트 단위의 각 철자 하나로 구성하고 이에 대한 연관관계를 기술하여 경우의 수가 너무 많다는 점에서 착안하여 이를 개선한 알고리즘이 Jump Aho-Corasick 알고리즘이다. Jump Aho-Corasick 알고리즘은 기존의 Aho-Corasick 알고리즘이 가지는 상태 수를 대폭 축소하고, 패턴를 트리구조화 하는 과정에서 연관관계가 있는 철자의 묶음으로 트리를 구성한다. 이를 통해 패턴과 비교 시 최초 트리 분기점에서 시작한 단어가 블록 단위의 묶음과 일치하지 않을 경우 그 묶음의 길이만큼 점프하면서 비교하여 성능을 개선한 알고리즘이다. (그림 3)에서 보는 바와 같이 Jump Aho-Corasick 알고리즘은 연관성 있는 부분들의 묶음으로 각 패턴을 조사 및 분해하여 묶음 단위의 블록을 이용하여

트리를 구성한다. Jump Aho-Corasic 알고리즘은 비록 Aho-Corasic 알고리즘이 가지는 메모리 접근의 증가 및 철자 단위의 매칭으로 인한 병목 현상등 의 문제를 어느 정도 완화시키기는 하지만, 실제 존재하는 패턴의 길이가 매우 다양하고 그에 따라 많은 메모리를 필요로 하는 단점을 가지고 있다. (그림 3)에서 볼 수 있듯이 이는 패턴의 범위 및 상태 수를 획기적으로 축소하고, 실제 매칭 진행시 건너뛰며 진행될 확률을 높임으로써 성능 개선을 나타냈다.

하지만 Aho-Corasic 알고리즘 보다 패턴간의 상관관계에 의존적이 되었다. Aho-Corasic 알고리즘이 가지는 한계를 그대로 유지하는 것으로 패턴간의 관계에 많은 영향을 받으며 성능 수치에 대한 예측이 어려운 알고리즘이다.

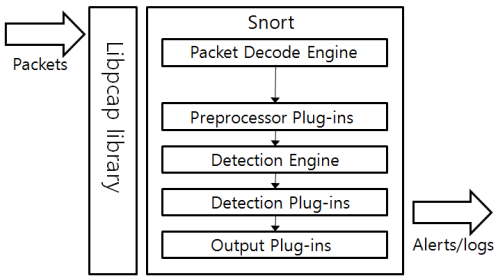


(그림 3) Jump Aho-Corasick 알고리즘

3. 해시테이블 기반의 침입탐지 시스템

3.1 Snort의 패킷 흐름

(그림 4)는 Snort의 패킷의 흐름을 보여준다. libpcap 라이브러리를 통하여 네트워크 트래픽을 캡처한다. 디코드 엔진은 패킷을 분석하여 TCP와 UDP 프로토콜과 같은 프로토콜용 패킷구조의 형태로 디코딩한다. 디코딩된 패킷은 Snort의 전처리기에서 탐지 엔진에 패킷이 도착하기 전에 패킷의 내용을 전처리기에서 패킷의 내용을 검사하여 경고를 보내거나 수정하는 작업을 수행한다. 탐지 엔진은 각 패킷을 룰 파일의 항목과 비교한다. 탐지 플러그인은 패킷에 대한 추가 탐지 기능을 제공하며 룰 파일에 포함되어 있는 각 패턴들과 비교할 수 있는 탐지 플러그인과 연결하여 탐지 플러그인의 저장되어 있는 탐지 패턴들과 비교 작업 후 패턴이 일치하는 것이 있으면 알람을 출력 플러그인으로 전달하여 경고, 로그 작성을 한다.

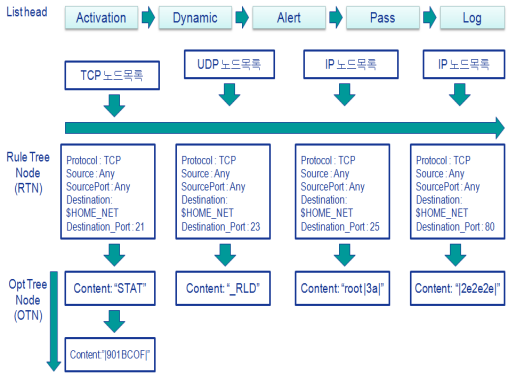


(그림 4) Snort의 패킷 처리

3.2 3D 연결 리스트

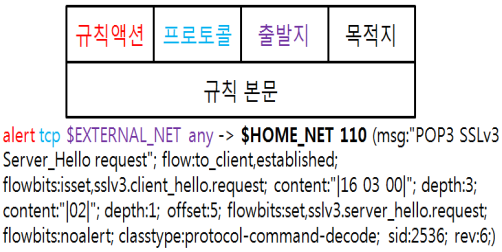
Snort는 패킷을 규칙과 잘 비교하기 위하여 규칙을 메모리에 효율적으로 저장해야 한다. 이를 위해 Snort는 3D 연결 리스트를 사용한다. (그림 5)는 Snort의 3D 연결 리스트로써 패킷이 디코딩 된 후에 패킷을 처리하는 단계를 나타낸다. Activation 는 경고를 발생시키고 다른 단계인 Dynamic 규칙을 활성화 한다. Dynamic은 activation 규칙에 의해 활성화 되며 트래픽을 로그로 저장하고 Alert는 경고를 발생시키고 패킷을 로그로 남긴다. Pass는 패킷을 무시하고 Log는 유입된 패킷을 검사하여 경고 발생 없이 로그로 저장한다. 이와 같은 각각의 일련의 과정 중 RTN에서는 TCP, UDP, ICMP, IP 등과 같은 프로토콜을 식별하여 각각의 Rule 파일과 연결하며 OTN(Option Tree Node) 규칙의 본문에는 Content에 정의된 패턴들과 분류된 패턴들과 매칭하여 침입을 탐지한다. Snort에서는 이와 같은 패턴을 Rule 파일에 추가 하여

사용할 수 있다.



(그림 5) Snort의 3D 연결리스트

3.3 Snort 규칙정의



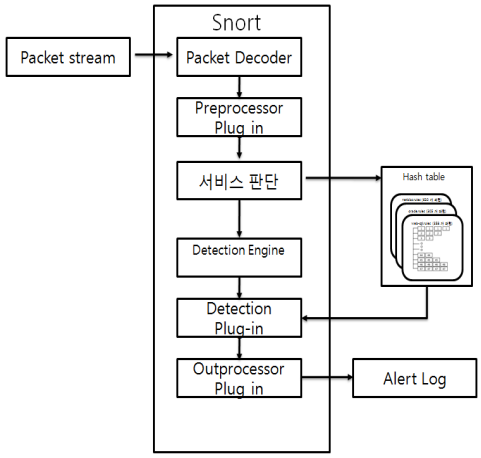
(그림 6) Snort의 규칙

규칙액션은 5개의 옵션(Activation, Dynamic, Alert, Pass, Log)을 가지고 있으며 프로토콜은 TCP, UDP, IP, ICMP를 지원한다. 출발지와 목적지는 포트를 선언을 통하여 패킷이 가지는 서비스를 분석할 수 있다. content 패킷의 데이터를 비교 할 때 사용(content :: "^|0E|1|C0 B0 3B 8D|~|0E 89 FA 89 F9|") depth 규칙과 일치하는 패턴을 찾기 위해 검사할 바이트의 수를 지정, offset 검색을 시작할 패킷의 특정 위치를 지정, nocase 규칙 content에서 대소문자를 구별하지 않음, 이 옵션을 사용하려면 규칙내부에 content 옵션이 문자열로 미리 정의되어 있어야한다.

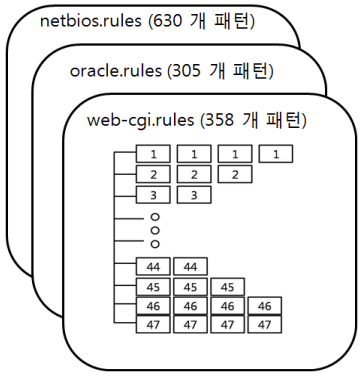
session 프로토콜 세션의 평문 데이터를 화면에 출력한다. 이와 같은 규칙 생성방법을 통하여 새로운 형태의 침입이 발생할 경우에 Rule 파일내의 content의 내용을 추가, 수정 할 수 있다.

3.4 패턴 해시테이블

Rule 파일의 content의 문자열을 해시함수를 통해 해시 값을 구할 수 있으며 구한 해시 값을 인덱스로 사용한다. (그림 7)은 패킷헤더를 분석하여 패턴의 수에 따라서 기존 패턴 매치 방법과 해시테이블 매치 방법으로 분류하여 패턴을 탐지하고 <표 1>



(그림 7) 패턴 매치 구성



(그림 8) 룰 파일 패턴의 해시 테이블

같이 패턴의 수가 많은 룰 파일들은 (그림 8)은 각 룰이 포함하고 있는 패턴들을 해시 테이블을 작성한 것이다.

해시 테이블은 content 내용의 문자열들의 각각의 아스키 코드 값에 상수의 거듭제곱을 곱한 후의 합을 인덱스 개수로 만들고 싶은 만큼의 수로 나머지를 구하면 해시 값을 생성 할 수 있다. 각각의 문자열을 해시 테이블로 생성시 문자의 아스키 코드를 더하여 해시 값을 생성할 수 있지만, 단순히 더한 값으로 해시 값을 생성하는 경우에는 content의 문자열들의 배열이 변경되어도 같은 해시 값을 가져 충돌을 발생 할 수 있게 된다. 따라서 아래 식 (1)은 으로 해시 테이블을 구성할 수 있다.

$$(X_1)*C^{\wedge n} + (X_2)*C^{\wedge (n-1)} + (X_3)*C^{\wedge (n-2)} + \cdots + (X_{n-1})*C + X_n$$

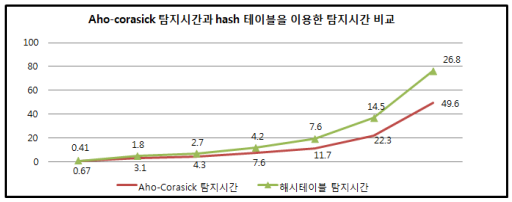
식 (1) 해시 함수 식

4. 시험 및 검증

Aho-Corasick과 해시 테이블로 변환된 패턴을 탐지하는 시간의 비교를 위해 임의의 문자열을 생성하였으며 생성된 패턴 중 Aho-Corasick과 해시 테이블에 이용하여 찾을 동일한 문자열을 선택하여 탐지 하였다.

패턴수	10	50	100	150	200	500	1000
Aho-Corasick	0.67	3.1	4.3	7.6	11.7	22.3	49.6
해시테이블	0.41	1.8	2.7	4.2	7.6	14.5	26.8

(그림 9) Aho-Corasick 방법과 문자열을 해시화 시킨 테이블에서의 동일한 문자열을 탐지하기까지의 시간을 기록한 것이다. 그림에서와 같이 Aho-Corasick 방법을 이용하여 패턴을 탐지하는 것보다 해시 테이블화 시킨 패턴에서 탐지하는 것이 시간이 단축되는 것을 알 수 있다.



(그림 9) 탐지시간 비교

5. 결 론

룰 패턴이 적은 경우에는 해시 테이블을 이용하여 탐지하는 것과 많은 시간의 차이를 가지지 못한다. 따라서 본 논문에서는 룰 파일이 가지는 패턴의 수를 분석하여 특정 서비스(패턴 수가 많은 룰 파일) 패턴을 해시 테이블화 된 것과 비교함으로써 매칭 대상이 되는 패턴의 수를 개수를 줄일 수 있으며. 패턴의 매칭 수를 줄임으로써 유입된 패킷이 가지는 패턴을 탐지 엔진에서 비교 하는데 시간을 기존의 탐지 방법과는 패턴 탐지 시간을 줄일 수 있다.

하지만 해시 테이블은 현재의 룰이 가지고 있는 패턴을 기준으로 생성된 것임으로써 새로운 탐지 패턴이 추가 되는 경우에는 룰 파일에 추가한 후에 다시 해시 테이블로 생성해야 하는 정적인 업데이트를 동적인 업데이트에 관한 연구가 필요하다.

참 고 문 헌

[1] 손만경 “침입탐지시스템의 처리성능을 개선한 ESnort 시스템 설계 및 구현”, 경기대학교 대학원 석사학위 논문, 2005.

[2] A. Aho, M. Corasick, “Efficient string matching : an aid to biliographic search”, Comm. ACM, Vol. 18, pp. 333-340, 1975.

[3] J. Lockwood, “Fast and Scalable Pattern Ma-

tching for Content Filtering", Architecture for Networking and Communication System (ANCS), 2005.
[4] 왕정석, 권희웅 "시그너처 해싱에 기반한 고성능 침입방지 시스템", 한국컴퓨터 종합학술대회논문집 Vol. 34, No. 1, 2007.
[5] 한국정보보호진흥원, 인프라보호단/보안관리팀 "SNORT를 이용한 IDS 구축", 2005.
[6] Zachary K. Baker and Viktor K. Prasanna, "High-throughput Linked-Pattern Matching for Intrusion Detection Systems", 2007.
[7] N. Paulauskas and J. Skudutis, "Investigation of the Intrusion Detection System 'Snort' Performance", 2008.
[8] H. Bos and Kaiming Huang, "Towards software-based signature detection for intrusion prevention on the network card", 2005.
[9] Fabiano C. Botelho Rasmus Pagh, "Perfect Hashing for Data Management Applications", 2007.

김 병 훈

2008년 안양대학교 전기전자 공학과(공학사)
2008년 경기대학교 정보보호학과 석사과정

하 옥 현

1978년 성균관대학교 정치외교학과(정치학사)
1980년 서울대학교 행정대학원(행정학석사)
1998년 프랑스 사회과학대학원(EHESS) 박사과정(DEA 취득)
2005년 고려대학교 정보보호대학원(공학박사)
2008년~현재 호남대학교 경찰법행정학부 교수

신 제 철

1980년 동국대학교 경찰행정학과(학사)
1985년 동국대학교 행정대학원(석사)
2003년 미 UCSB 연수
2009년 동국대학교 대학원 경찰행정학(박사)
2009년~현재 세명대 초빙교수
2009년~현재 동국대학교 겸임교수
2009년~현재 경기대학교 대우교수
2009년~현재 경기산업기술보호특화센터 부소장
2009년~현재 경기산업기술보호협의회 수석 부회장