情報保護學會論文誌
第 21 卷 第 1 號, 2011. 2

패킷검사시간을 단축하기 위한 혼합형 다중패턴매칭 기법*

이 재 국,† 김 형 식‡
충남대학교

A Hybrid Multiple Pattern Matching Scheme to Reduce Packet Inspection Time*

Jae-Kook Lee,† Hyong-Shik Kim‡
Chungnam National University

요 약

인터넷 공격으로부터 내부 네트워크를 보호하기 위하여 침입탐지/차단시스템이 광범위하게 사용되고 있다. 패킷검사 시간을 단축하는 것은 침입탐지/차단시스템의 성능을 개선하는데 중요한 과제이다. 침입탐지/차단시스템에서의 패킷검사는 한 번에 여러 개의 패턴을 검색해야 하므로 다중패턴매칭 기법이 사용되는데, 유한 오토마타를 이용하는 방법과 시프트 테이블을 이용하는 방법으로 크게 구분된다. 본 논문에서는 비교해야 할 패턴 집합이나 페이로드에 따라 각 방법들의 성능이 악화되는 사례들을 보이고, 어떤 경우에도 적정 수준의 패턴매칭 성능을 보장하기 위하여 두 방법을 결합하는 혼합형 다중패턴매칭 기법을 제안한다. 실제 트래픽을 이용하여 실험한 결과는 제안된 기법이 패턴매칭에 소요되는 시간을 효과적으로 단축할 수 있음을 보인다.

ABSTRACT

The IDS/IPS(Intrusion Detection/Prevention System) has been widely deployed to protect the internal network against internet attacks. Reducing the packet inspection time is one of the most important challenges of improving the performance of the IDS/IPS. Since the IDS/IPS needs to match multiple patterns for the incoming traffic, we may have to apply the multiple pattern matching schemes, some of which use finite automata, while the others use the shift table. In this paper, we first show that the performance of those schemes would degrade with various kinds of pattern sets and payload, and then propose a hybrid multiple pattern matching scheme which combines those two schemes. The proposed scheme is organized to guarantee an appropriate level of performance in any cases. The experimental results using real traffic show that the time required to do multiple pattern matching could be reduced effectively.

Keywords: Pattern Matching, Packet Inspection, Aho-Corasick, Wu-Manber, Algorithm, IDS, IPS

I. 서 론

네트워크 환경이 발전하면서 웜이나 바이러스, DDoS 등과 같은 다양한 형태의 공격이 증가하고 있다. 이러한 공격으로부터 내부 네트워크를 보호하기 위하여 기본적으로 침입탐지/차단시스템이 사용된다. 침입탐지/차단시스템은 유입되는 패킷에 미리 정의된 규칙이 존재하는지 탐색하여 공격 여부를 판단한다.

접수일(2009년 12월 30일), 수정일(2010년 5월 30일), 게재확정일(2010년 12월 20일)
* 이 연구는 2008년도 충남대학교 학술연구비에 의해 지원되었음.
† 주저자, jxempire@cnu.kr
‡ 교신저자, hkim@cnu.kr

예를 들어 대표적인 공개소프트웨어 침입탐지/차단시스템인 Snort[1]는 다음과 같이 규칙을 정의하여 패킷을 검사한다.

```
drop tcp any any -> any 25 (msg:"SMTP attack"; content:"attack"; id:1)
```

위와 같이 규칙이 정의되면 Snort의 탐색엔진은 유입되는 패킷의 헤더에서 프로토콜, 출발지주소 및 포트, 목적지주소 및 포트를 비교하고 페이로드 부분에 'attack'이라는 문자열이 존재하는지 검사하여 일치되는 패킷을 탐지하여 경고하거나 차단한다. 그러나 공격의 형태가 다양화되고 그 시도가 많아지면서 침입탐지/차단시스템의 규칙 개수도 증가하게 되었다. 따라서 페이로드에서 규칙에 주어진 문자열(이하 패턴)을 검색하는 침입탐지/차단시스템의 패턴매칭 알고리즘은 성능에 가장 큰 영향을 미치는 부분이 되었다. Snort의 경우에 패턴매칭 알고리즘에 따라 차이가 있지만 패턴을 검색하기 위하여 전체 수행시간의 40~70%를 소비하고 전체 수행 명령어의 60~85%가 이용된다[2].

동시에 비교해야할 패턴의 수가 많다는 점에서 탐색엔진에서는 다중패턴매칭 기법이 필요하다. 다중패턴매칭 기법들은 크게 전처리 단계에서 유한 오토마타(finite automata)를 생성하고 이를 기반으로 패턴을 검색하는 방법과, 전처리 단계에서 불일치 문자 휴리스틱(bad character heuristic)을 이용하여 시프트테이블(shift table)을 구성하고 이를 기반으로 한꺼번에 2바이트 이상 시프트하며 패턴을 검색하는 방법으로 구분된다. 패턴집합을 이용하여 페이로드를 검색하는 문제에 앞에서의 기법을 적용하면, 패턴매칭에 소요되는 시간이 증가하는, 즉 패턴매칭 성능이 저하되는 경우가 발생되는데, 발생 조건은 두 기법에서 서로 다르다.

본 논문에서는 이러한 성능 저하 문제를 분석하고, 특정 조건에 따라 성능이 저하되는 문제점을 회피하기 위하여 동적으로 능동적으로 패턴매칭 기법을 전이하는 혼합형 다중패턴매칭 기법(hybrid multiple pattern matching scheme)을 제안하고 이에 필요한 전이조건을 보인다. 또한 실제 트래픽을 이용하여 제안된 기법이 패턴매칭에 소요되는 시간을 효과적으로 단축할 수 있음을 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 관련연구로 기존 패턴매칭 알고리즘에 대하여 기술한다. 3절에서는 패턴의 개수가 다양한 그룹들에 대하여 기존 패턴매칭 기법에서의 성능 한계를 보이고, 이를 기반으로 조건에 따라 패턴매칭 기법을 전이하는 혼합형 다중패턴매칭 기법을 제안한다. 4절에서는 혼합형 다중패턴매칭 기법을 서로 다른 전이조건에 따라 시험한 결과를 보이고 분석한다. 끝으로 5절에서 결론을 맺는다.

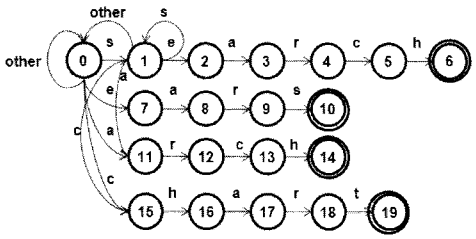
II. 관련연구

패턴매칭을 위한 간단한 방법은 Brute Force 탐색 알고리즘과 같이 차례로 페이로드와 패턴의 문자열을 순서대로 비교하는 것이다. 또는 Boyer- Moore 알고리즘[3]과 같이 불일치 문자를 이용한 휴리스틱을 적용하거나 일치하는 접미부를 이용한 휴리스틱(good suffix heuristic)을 적용함으로써 불필요한 비교 과정을 생략하면서 패턴을 검색하는 단일패턴매칭 기법을 사용하는 것이다. 그러나 패턴의 개수가 많으면 단일패턴매칭 알고리즘은 모든 패턴을 검색하기 위하여 패턴의 개수만큼 패턴매칭 알고리즘을 반복해야 하므로 처리속도가 저하될 수밖에 없다. 이런 문제를 해결하기 위하여 한 번에 모든 패턴을 검색할 수 있는 다중패턴매칭 알고리즘이 제안되었다. 다중패턴매칭 알고리즘은 크게 유한 오토마타를 이용하는 방법과 시프트테이블을 이용하는 방법으로 구분된다.

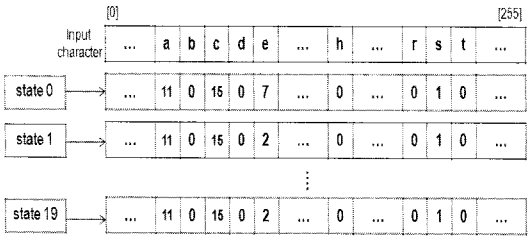
2.1 유한 오토마타 기반 패턴매칭 기법

유한 오토마타 기반의 다중패턴매칭 기법은 전처리 단계에서 여러 패턴을 종합하여 유한 오토마타와 전이함수를 구성하고 페이로드에서 패턴을 검색할 때 1바이트씩 입력으로 취하면서 상태를 전이하는 것을 특징으로 한다. Aho-Corasick 알고리즘(이하 AC 알고리즘)[4]은 대표적인 결정적 유한 오토마타 기반 패턴매칭 기법으로 다양한 영역에서 사용되어 왔다. 그런데 AC 알고리즘이 패킷검사에 적용되면서 성능 향상을 목적으로 개선된 알고리즘들이 제시되었는데, 결정적 유한 오토마타(deterministic finite automata)로 대체하는 것(이하 DFA-AC 알고리즘)[5,6]이 그것이다.

Character Indexed Aho-Corasick 알고리즘(이하 CIAC 알고리즘)[5]은 대표적인 DFA-AC 알고리즘으로, 결정적 유한 오토마타에 덧붙여서 캐시효과를 통해 성능을 개선할 목적으로 오토마타에 필요



〔그림 1〕 결정적 유한 오토마타 및 전이함수

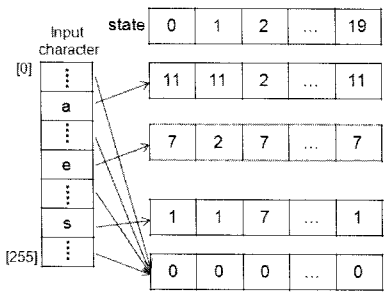


〔그림 2〕 DFA-AC 알고리즘의 자료 구조

한 메모리 사용량을 줄였다. 예를 들어, 4개의 패턴 {search, ears, arch, chart}을 CIAC 알고리즘에 적용하면 〔그림 1〕과 같이 전처리 단계에서 결정적 유한 오토마타와 전이함수가 구성 된다.

〔그림 1〕에서와 같이 형상화된 결정적 유한 오토마타를 구현하게 되면 〔그림 2〕와 같은 자료구조가 생성된다. 즉, DFA-AC 알고리즘에서 상태의 수를 n이라고 하고 문자가 1바이트를 사용할 경우 n × 256의 메모리가 필요하다.

그러나 CIAC 알고리즘의 경우 패턴에 포함된 문자의 분포가 다양하지 않다는 것을 활용하여 〔그림 3〕과 같이 자료 구조를 개선하여 메모리 사용량을 줄였다. 패턴들에 포함된 문자의 개수 $\Theta \leq 256$ 이므로 메모리 사용량은 $n \times (\Theta+1)$이 된다. Θ의 값이 적을수록 메모리 사용량을 줄이는 효과를 크게 할 수 있다.



〔그림 3〕 CIAC 알고리즘의 자료 구조

이외에도 유한 오토마타를 이용하려면 많은 양의 메모리가 필요하므로 이를 줄이기 위한 변형 기법들이 다양하게 제시되었다. Snort에서도 'sparse', 'banded', 'sparse-banded'로 불리는 변형된 유한 오토마타 기반의 알고리즘들이 사용되고 있다〔7,8,9〕. 그러나 메모리 효율성을 증대하기 위한 이러한 알고리즘들은 DFA-AC 알고리즘보다 최대 5배 이상 메모리양을 줄일 수 있지만 패킷검사 시간은 비슷하거나 오히려 증가하기도 한다〔4,7〕.

2.2 시프트테이블 기반 패턴매칭 기법

시프트테이블 기반의 패턴매칭 기법은 전처리 단계에서 모든 패턴에서 공통된 길이의 문자열을 취하고 일정 길이로 나누어 불일치 문자 휴리스틱을 적용하여 시프트테이블을 구성한 후에, 패턴검색 단계에서 페이로드를 1바이트 이상 시프트하며 패턴을 검색함으로써 검색시간을 단축하는 대표적인 다중패턴매칭 기법이다. Wu-Manber 알고리즘〔10〕(이하 WM 알고리즘)은 단일 패턴매칭 알고리즘인 Boyer-Moore 알고리즘을 다중 패턴매칭이 가능하도록 개선한 것으로, 다중 패턴매칭 성능이 평균적으로 가장 좋은 것으로 알려져 있다. WM 알고리즘을 패턴매칭에 적용하면 시프트가 성능에 미치는 영향이 크기 때문에 이 부분을 개선하는 변형된 방법들이 제시되어 왔다.

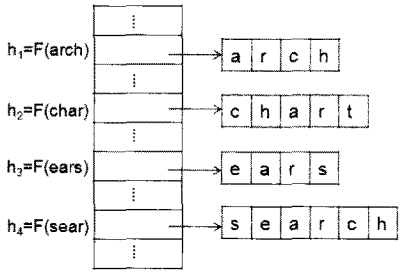
QWM 알고리즘〔11〕은 Wu-Manber 알고리즘(이하 WM 알고리즘)이 검색단계에서 패턴의 존재에 상관없이 1만큼 시프트하는 것을 개선하기 위한 것으로 Quick Search 알고리즘〔12〕을 적용하였다. 또한 패턴매칭을 위해 WM 알고리즘을 적용하고 있는 AGREP〔13〕이나 Snort의 사례를 보면, 접두사테이블(prefix table)을 사용하지 않고 시프트테이블과 해시테이블만을 사용하여 패턴매칭 가능하도록 WM 알고리즘을 변형(이하 MWM 알고리즘)하였다.

QWM 알고리즘과 MWM 알고리즘은 모두 전처리 단계에서 시프트테이블과 해시테이블을 생성한다. 〔그림 4〕는 주어진 패턴 {search, ears, arch, chart}에 대하여 전처리 단계에서 생성되는 MWM 알고리즘의 시프트테이블과 해시테이블을 나타낸다.

QWM 알고리즘은 MWM 알고리즘과 같은 해시테이블을 생성하지만 〔그림 5〕와 같이 시프트의 크기가 1만큼 큰 값으로 대체되는 대신 패턴의 블록의 길이에 해당되는 문자열을 저장할 헤드테이블이 추가로 필요하다.

B	se	ea	ar	rs	rc	ch	ha	others
시프트값	2	1	0	0	1	0	1	3

(a) 시프트테이블



(b) 해시테이블

〔그림 4〕 MWM 알고리즘의 자료구조

B	se	ea	ar	rs	rc	ch	ha	others
시프트값	3	2	1	1	2	1	2	4

(a) 시프트테이블

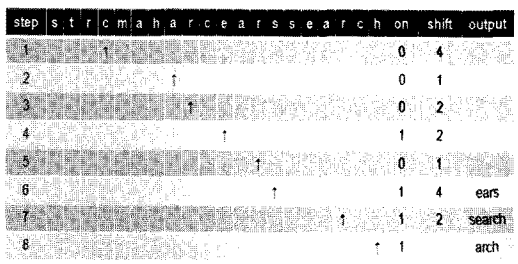
초기 B	se	ea	ar	ch	others
On/Off	1	1	1	1	0

(b) 헤드테이블

〔그림 5〕 QWM의 변경된 테이블

만약, 주어진 텍스트가 'strcmaharcearssearch'라고 할 때 검색 윈도우의 크기는 패턴의 최소길이인 4가 되고, 블록의 크기 B는 2가 된다. 〔그림 6〕은 QWM 알고리즘으로 〔그림 5〕와 같이 전처리 단계에서 생성된 패턴 테이블을 이용하여 실제 텍스트에 대한 패턴매칭 과정을 보인다.

1단계에서는 헤드테이블의 On/Off 값은(HEAD〔st〕)는 0이고 시프트값(SHIFT〔cm〕)은 4이므로 4만큼 시프트한다. 2단계에서도 헤드테이블(HEAD〔ma〕)의 On/Off값이 0이고 시프트값(SHIFT〔ar〕)이 1이므로 1만큼 시프트한다. 같은 방법으로 6단계에는 헤드테이블(HEAD〔ea〕) On/Off 값이 1



〔그림 6〕 QWM 알고리즘의 패턴매칭

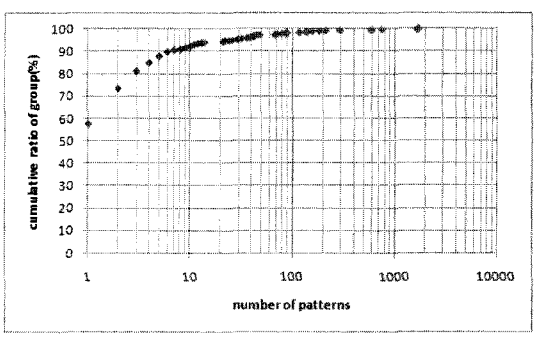
이므로 패턴이 정확하게 일치하는지 시프트테이블과 헤시테이블을 이용하여 MWM 알고리즘과 같은 방법으로 'ears'를 검색한다. MWM 알고리즘의 경우에는 다음으로 1만큼 시프트하여 다음 패턴검색을 계속 진행하지만 QWM 알고리즘은 시프트값(SHIFT〔ss〕)이 4인 것을 확인하고 4만큼 시프트할 수 있다. 같은 방법으로 7단계에서 'search'가 검색되고, 8단계에서 'arch'를 검색된다.

그러나 QWM 알고리즘은 찾고자 하는 패턴의 초기 블록 B값이 일치하는 텍스트가 많고, 하나의 텍스트에 그 값이 여러 번 반복되어 나타나는 경우에만 제한적으로 MWM 알고리즘보다 빠른 성능을 보인다. 또한 일반적인 환경에서 QWM 알고리즘은 MWM 알고리즘과 비교하여 시프트 크기를 증가시키는 대가로 헤드테이블 값을 비교하기 위한 시간이 별도로 소요되므로 성능이 오히려 저하되는 경향이 나타난다.

III. 혼합형 다중패턴매칭 기법

침입탐지/차단시스템에서 사용되는 규칙은 수치로 표현되는 헤더 부분과 수치로 표현되지 않는 페이로드 부분으로 구성된다. 그런데 페이로드 부분의 패턴과 비교하기 전에 패킷의 헤더를 먼저 비교하는 것이 바람직하므로 페이로드 부분과 비교해야 할 패턴은 전부가 아닌 일부로 국한된다. 따라서 실제 패턴 비교의 대상은 헤더 부분을 공유하는 규칙들에서의 패턴 그룹들이 되는데, 패턴 그룹에 속하는 패턴의 수는 1개부터 수천 개까지 존재한다.

〔그림 7〕은 Snort v2.8을 위해 2009년 1월 공개된 규칙 중에서 'content' 옵션을 갖는 8,122개의 패턴을 프로토콜과 포트번호를 이용하여 패턴 그룹을 생성할 때 각 그룹별 패턴의 개수를 보여준다. 패턴의 개수가 1개이면 다중패턴매칭이 필요하지 않는데, 여



〔그림 7〕 그룹별 패턴의 개수

기에 해당되는 그룹이 전체 580개 그룹의 58% 정도를 차지한다. 반면1672개 혹은 1712개의 패턴으로 구성된 그룹도 존재하는데, 이와 같이 패턴의 개수가 다양한 그룹들은 일정 정도 패턴매칭 속도에 영향을 미칠 수 있다.

3.1 기존 다중패턴매칭 기법의 한계점

시프트테이블 기반의 패턴매칭 기법 중에 MWM 알고리즘은 평균적으로 가장 빠른 다중패턴매칭 알고리즘으로 알려져 있다[9]. 그러나 최소패턴의 길이가 짧거나 패턴의 개수가 많으면 시프트 길이가 짧아져서 패턴매칭 속도가 저하된다. 또한 페이로드에 동일한 문자가 연속해서 나타나는 최악의 경우에도 성능이 급격하게 나빠지는 단점이 있다.

페이로드: 00000000000000000000000000000
패 턴: 0000a

예를 들어, 위와 같이 '0'이 연속적으로 나타나는 페이로드에서 마지막 문자만 다른 패턴을 검색한다고 가정하자. 침입탐지 목적으로 검사해야 하는 패킷들에서는 위와 같은 형태로 채워진 페이로드가 흔하게 존재한다. 패턴의 최소길이 m과 블록크기 B를 각각 4와 2로 정하고 MWM 알고리즘을 적용하면, 페이로드의 검색 윈도우의 문자열 '0000'의 블록 '00'의 시프트 값(SHIFT[00])이 0이므로 해시 값 비교를 통해 일치 여부를 판단하게 된다. 해시 값 F(0000)에 패턴 '0000a'이 연결되고 있으므로 결국 페이로드와 패턴의 문자열 비교 연산이 필요하지만, 마지막 문자 'a'가 일치하지 않으므로 비교는 최종적으로 실패한다. 이제 페이로드에서 다음 패턴을 찾기 위해서는 단지 1바이트만 시프트하여 지금까지 했던 것처럼 시프트 값 비교와 해시 값 비교, 문자열 비교를 반복적으로 수행해야 하므로, 동일한 문자를 기준으로 패턴의 길이를 넘어서는 페이로드의 문자 수만큼 위와 같은 과정을 반복해야 하는데 이것은 패턴매칭 성능을 크게 저하시키는 이유로 작용한다.

유한 오토마타 기반의 패턴매칭 기법인 DFA-AC 알고리즘에서는 페이로드에서 1바이트씩 문자열을 비교하므로 위와 같은 최악의 경우에 대한 문제점을 갖고 있지 않다. 그렇지만 패턴매칭을 위하여 항상 1 바이트씩 시프트하면서 페이로드의 처음부터 끝까지 비교하므로 이것도 역시 성능저하 요인이 된다.

유한 오토마타 기반의 패턴매칭 기법과 시프트테이블 기반의 패턴매칭 기법에는 공통적으로 위와 같은 한계점이 존재하는데, 두 기법에 대한 포괄적인 분석을 토대로 그 한계점들을 정리하면 다음과 같다.

문제 1:

유한 오토마타 기반의 패턴매칭 기법은 항상 1 바이트씩 시프트하면서 페이로드의 첫 번째 바이트부터 마지막 바이트까지 패턴을 검색하므로 처리속도가 저하된다.

문제 2:

시프트테이블 기반의 패턴매칭 기법은 패턴의 개수가 많으면 비교하지 않고 이동할 수 있는 시프트 길이가 줄어들게 된다. 또한 패턴에 존재하는 동일한 문자가 페이로드에 연속적으로 나타나면 시프트 값이 0이 되어 해시 값 비교와 문자열 비교를 반복적으로 해야 되므로 처리속도가 저하된다.

3.2 해결방안

본 절에서는 3.1절에서 제기한 유한 오토마타 기반의 패턴매칭 기법과 시프트테이블 기반의 패턴매칭 기법의 한계점을 극복하고, 패턴의 개수가 다양한 그룹들로 구성된 규칙을 검색하는 경우에도 항상 패턴검색 시간을 단축하기 위한 목적으로 혼합형 다중패턴매칭 기법을 제안한다. 즉, 앞에서 보인 각 기법에서의 취약점에 대한 분석을 바탕으로 특정 기법에 대한 성능이 저하되는 조건을 정형화하고, 이 조건이 만족될 경우 즉시 다른 기법으로 전이되도록 두 기법을 결합하는 것이 핵심이다. 그리고 앞에서 제시한 예외적인 상황이 없다면 통상 시프트테이블 기반의 패턴매칭 기법이 우월하므로 그 기법을 기본으로 삼는다.

따라서 시프트테이블 기반의 패턴매칭 기법의 성능이 저하되는 경우, 즉 패턴의 개수가 많거나 패턴에 존재하는 동일한 문자가 페이로드에 연속적으로 나타나면 유한 오토마타 기반의 패턴매칭 기법으로 전이한다. 이때 유한 오토마타 기반의 패턴매칭 기법에서 일정 조건을 만족하면 다시 시프트테이블 기반의 패턴매칭 기법으로 복귀한다. 3.1 절에서의 한계점을 해결하기 위한 원칙을 정리하면 다음과 같다.

해결원칙 1:

시프트테이블 기반의 패턴매칭 알고리즘을 기본 패턴매칭 알고리즘으로 사용하여 1 바이트 이상 시프트

〔표 1〕 전이조건

번호	전이조건	설 명
1	counter[B] >= TH_{shift}, where counter[B] = counter[B] + 1 if(SHIFT[B] == 0)	블록 B에 대하여 시프트 값(SHIFT[B])이 0이 되면 카운터(counter[B])를 증가시키고, 그 값이 임계값(TH_{shift}) 이상이면 전이
2	counter[h] >= TH_{hash}, where counter[h] = counter[h] + 1 if(HASH[h] exists)	해시 값 h에 해당하는 패턴이 존재하면 해시 값의 카운터(counter[h])를 증가시키고, 그 값이 임계값(TH_{hash}) 이상이면 전이
3	counter >= TH_{string}, where counter = counter + 1 if(string length > TH_{length})	비교한 문자열 길이가 일정길이(TH_{length})를 초과하면 카운터(counter)를 증가시키고 그 값이 임계값(TH_{string}) 이상이면 전이

하며 페이로드의 첫 번째 바이트부터 마지막 바이트까지 패턴을 검색한다.

해결원칙 2:

시프트테이블 기반의 패턴매칭 기법의 패턴매칭 속도를 저하시키는 조건이 발생되면 유한 오토마타 기반의 패턴매칭 기법으로 알고리즘을 전이하여 최악의 경우를 회피한다.

이제 혼합형 다중패턴매칭 기법에 필요한 전이조건을 도출하는 것이 중요한 문제가 된다. 본 논문에서는 실제 사례에 대한 포괄적인 분석을 토대로 시프트테이블 기반의 패턴매칭 기법에서 유한 오토마타 기반의 패턴매칭 기법으로 전이하기 위한 세 가지 조건을 제안한다.

첫 번째 조건은 패턴의 개수가 많거나 동일한 문자로 구성된 페이로드로 인하여 시프트 값이 연속적으로 0이 되어 1 바이트 이상 시프트하지 못하는 현상이 반복되는 경우이다. 즉, 시프트 값이 0인 경우가 연속적으로 나타나는지 판단하기 위하여 시프트테이블에서 시프트 값이 0인 블록이 나타나면 카운터를 증가시키고, 그 값이 임계값(TH_{shift}) 이상인지 확인한다 (이하 조건 1).

두 번째 조건은 패턴의 해시 값과 일치하는 문자열이 연속적으로 나타나는 경우이다. 페이로드에서 m만큼의 문자열을 해시함수를 이용하여 해시한 값이 해시테이블에 존재할 경우 카운터를 증가시켜 그 값이 임계값(TH_{hash}) 이상이 되었을 때를 전이조건으로 본다 (조건 2).

세 번째 조건은 일정길이(TH_{length}) 이상인 문자열 비교가 연속적으로 나타나는지 판단하는 것이다. 시프트테이블 기반의 패턴매칭 기법에서 문자열 비교가 패턴매칭 속도에 가장 큰 영향을 미친다. 따라서 일정길이 이상 문자열 비교를 할 때마다 카운터를 증가시키고 그 횟수가 임계값(TH_{string}) 이상이 되었을 때를 전이조건으로 간주한다 (조건 3).

시프트테이블 기반의 패턴매칭 기법에서 유한 오토마타 기반의 패턴매칭 기법으로 전이하기 위한 조건을 정리하면 〔표 1〕과 같다.

제안한 혼합형 다중패턴매칭 기법은 〔표 1〕에서의 전이조건을 만족시키면 시프트테이블 기반의 패턴매칭 기법에서 유한 오토마타 기반의 패턴매칭 기법으로 전이하고, 일정개수의 패킷을 처리한 이후에 다시 시프트테이블 기반의 패턴매칭 기법으로 복귀한다. 이때 전이조건에 대한 연산이 단순하기 때문에 전이에 수반되는 오버헤드는 무시할 수 있는 수준일 것으로 예측된다.

IV. 성능시험

제안된 혼합형 다중패턴매칭 기법의 패턴매칭 성능을 측정하기 위해서는 유한 오토마타 기반의 패턴매칭 기법과 시프트테이블 기반의 패턴매칭 기법이 각각 필요하다. 본 논문에서는 2절에서의 분석에 바탕을 두되 객관적인 비교가 가능하고 최신 검색 기법을 포용할 수 있도록 snort에 구현된 알고리즘들을 채택하여 유한 오토마타 기반의 패턴매칭 기법으로는 DFA-AC를, 시프트테이블 기반의 패턴매칭 기법으로는 MWM을 활용하기로 한다.

우선 성능시험에 필요한 패킷데이터를 준비하였다. Snort 2.3.3 규칙 집합에서 'content' 옵션을 갖는 규칙을 패턴으로 이용하고, DEFCON 9의 "Capture The Flag" 대회를 진행하며 덤프한 패킷〔14〕중 일부를 패킷데이터로 활용하였다. 〔표 2〕는 성능시험

〔표 2〕 성능시험 패킷데이터

	파일명	파일크기(바이트)	패킷수	탐지율
패킷데이터 1	defcon_eth4.dump	15,627,152	153,480	≈0 ~ 12%
	defcon_eth4.dump2	21,879,534	26,252	≈0 ~ 16%
	defcon_eth4.dump3	21,374,117	143,899	≈0 ~ 9%
	defcon_eth5.dump	21,081,854	46,926	≈0 ~ 16%
	defcon_eth5.dump2	21,147,417	92,334	≈0 ~ 10%
패킷데이터 2	defcon_eth0.dump4	21,046,409	14,222	0, ≈100%
	defcon_eth0.dump5	21,307,560	14,397	0, 100%
	defcon_eth0_dump6	21,465,295	41,350	0, 100%
	defcon_eth0_dump7	21,317,207	19,460	0, ≈100%
	defcon_eth0_dump8	21,341,484	19,489	0, ≈100%

에 사용된 덤프 패킷의 특징을 간략히 정리한 것이다.

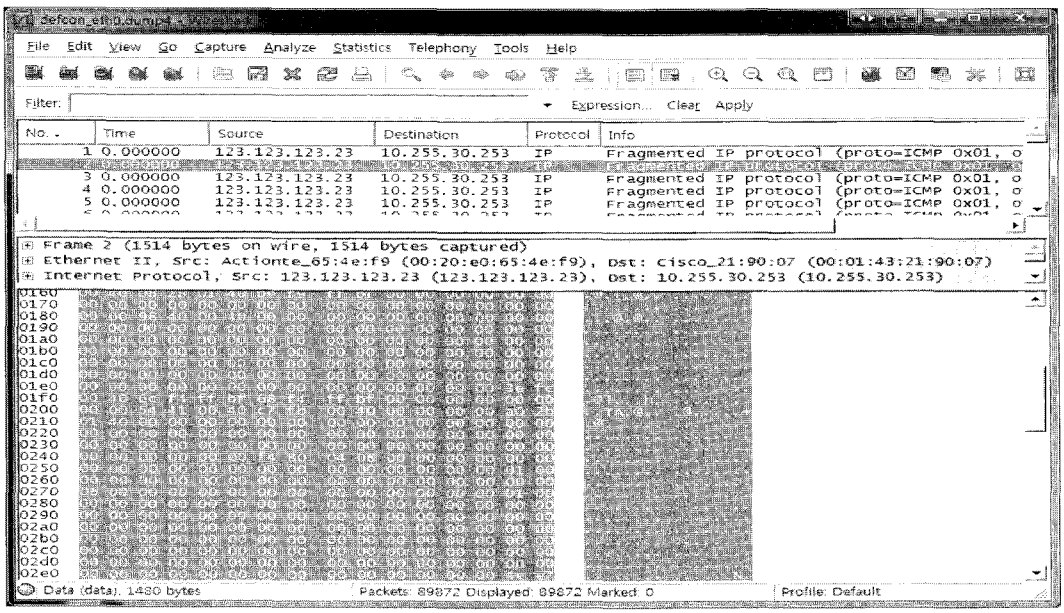
탐지율은 전체 패킷 중에서 1개 이상 패턴이 일치하는 패킷의 수를 비율로 나타낸 것이다. 패킷데이터 1은 일반적인 경우로 다양한 종류의 페이로드가 존재하는 경우인 반면, 패킷데이터 2는 같은 문자로 이루어진 페이로드를 갖는 패킷데이터로 최악의 경우를 시험하기 위한 것이다. 패킷데이터 1에 대해서는 사용하는 패턴집합의 크기를 증가시키면 최대 10% 전후까지 탐지율이 점진적으로 증가하지만, 패킷데이터 2는 대부분 특정 패턴을 포함하고 있기 때문에 패턴집합의 크기가 일정 수준이 넘기 전에 0%였던 탐지율이 특정 패턴이 포함되면 바로 100%로 증가한다.

〔그림 8〕은 프리웨어 프로그램인 Wireshark〔15〕를 이용하여 패킷데이터 2의 일부를 보인 것이다. 동일한 패킷이 연속적으로 나타나고 동일한 문자 '00'이 연속적으로 반복되는 것을 확인할 수 있다.

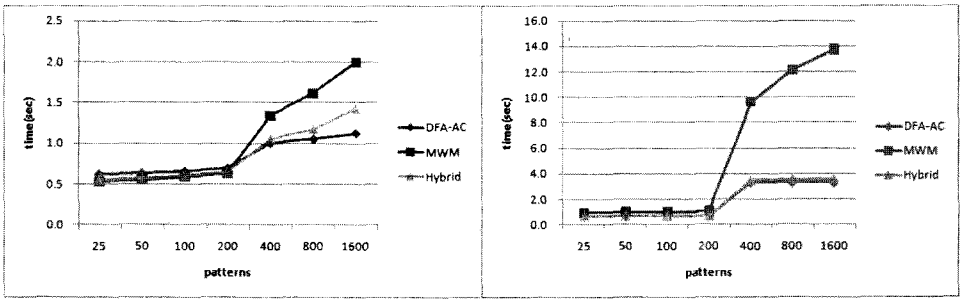
성능시험에 사용된 시스템의 CPU는 Intel Core 2 Quad (2.33 GHz)이고, 메모리는 3.25 GB이다. 패턴매칭에 소요되는 시간만을 측정하기 위하여 본 논문에서는 패킷에서 페이로드를 미리 추출하여 패턴들과 비교했다.

4.1 전이조건에 따른 성능시험

패턴의 개수가 다양한 그룹들이 존재할 때 전이조건에 따른 제안한 기법의 패턴매칭 속도를 측정하기 위하여 〔표 3〕과 같이 각 전이조건의 임계값을 정하였다.

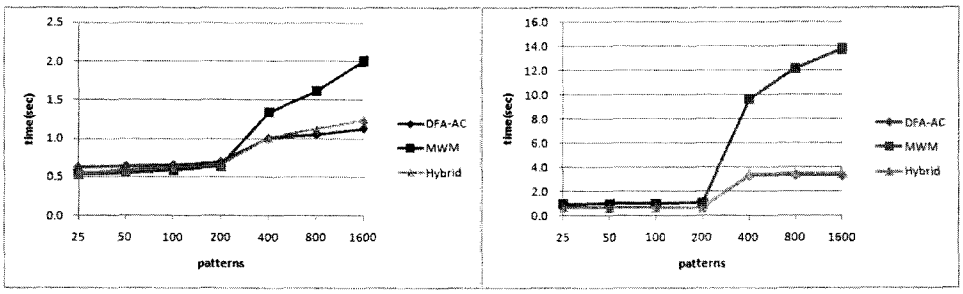


〔그림 8〕 패킷데이터 예제



(a) 패킷데이터 1 (b) 패킷데이터 2

〔그림 9〕 조건 1을 적용할 때의 성능 측정 결과



(a) 패킷데이터 1 (b) 패킷데이터 2

〔그림 10〕 조건 2를 적용할 때의 성능 측정 결과

〔표 3〕 전이조건에 필요한 임계값

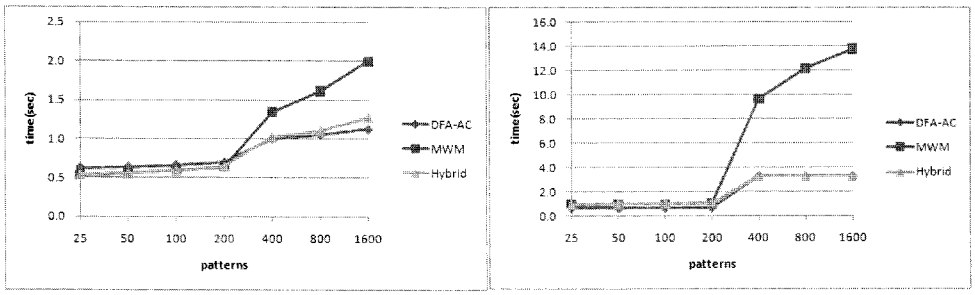
변수	임계값
TH_{shift}	10,000
TH_{hash}	10,000
TH_{length}	10
TH_{string}	1,000

TH_{shift}와 TH_{hash}는 시프트 연산과 해쉬 연산을 반복하는 정도를 나타내므로 10,000회 정도 반복되면 유한 오토마타 기반 패턴매칭기법으로 전이하는 것이 바람직하다고 가정했고, TH_{length}과 TH_{string}는 길이 10 이상인 문자열이 1,000 회 정도 반복되면 3.1 절에서의 문제점이 발생된다고 본다.

〔그림 9〕는 혼합형 다중패턴매칭 기법에 조건 1을 적용했을 때 DFA-AC 기법과 MWM 기법, 그리고 본 논문에서 제시된 기법의 성능을 측정한 결과를 보인다. 그래프에서의 시간은 각 샘플데이터 파일에 대하여 패턴매칭 연산을 수행할 때 소요되는 시간의 합이다. 일반적인 경우에는 〔그림 9〕(a)와 같이 패턴의 개수가 적은 그룹의 패턴매칭 속도는 MWM 알고리즘이 빠른 반면, 패턴의 개수가 많은 그룹의 경우에는 MWM 알고리즘에서 시프트 값이 0이 되는 블록이 상대적으로 많아지기 때문에 오히려 DFA-AC 알고리즘의 처리속도가 빠르다. 그러나 제안한 기법을 적용하면 패턴의 개수에 상관없이 성능이 좋은 쪽으로 수렴하는 것을 확인할 수 있다. 〔그림 9〕(b)를 보면 패킷데이터 2가 시프트 값이 0이 되는 블록이 연속적으로 나오는 최악의 경우를 유발하므로 MWM 알고리즘의 처리속도는 저하됨에도 불구하고 제안한 혼합형 다중패턴매칭 기법은 DFA-AC 알고리즘의 처리속도와 비슷한 것을 볼 수 있다.

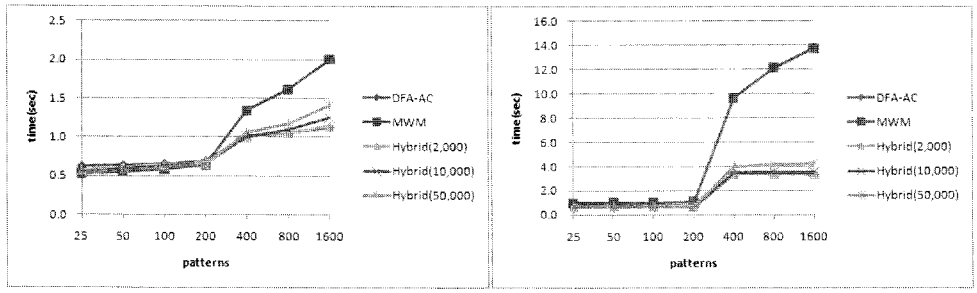
즉 제시된 방법은 기본적으로 시프트테이블 기반 다중패턴매칭 기법의 장점을 그대로 활용하고 있으며, 패턴집합의 크기가 커질 때는 유한 오토마타 기반 다중패턴매칭 기법과 유사하도록 전이하는데, 이는 제시된 기법의 전이 모델이 다양한 패턴집합의 크기에 대하여 능동적으로 대응할 수 있음을 입증한다고 볼 수 있다.

〔그림 10〕은 조건 2를 적용하여 시험한 결과이다.



(a) 패킷데이터 1 (b) 패킷데이터 2

〔그림 11〕 조건 3을 적용할 때의 성능 측정 결과



(a) 패킷데이터 1 (b) 패킷데이터 2

〔그림 12〕 임계값 변경에 따른 성능 측정

조건 2를 적용한 전이방법은 패킷데이터 1의 경우에 DFA-AC 알고리즘에서 처리하는 패킷이 조금 줄어들지만 처리속도는 조건 1을 사용하는 경우와 비슷한 추이를 보인다.

〔그림 11〕은 조건 3을 적용할 때의 시험결과이다. 조건 3을 이용하면 패턴의 개수가 적을 때는 임계값 TH_{length} 이상 문자열 비교를 하는 패턴이 적기 때문에 DFA-AC 알고리즘으로 전이가 일어나지 않아 MWM 알고리즘의 수행시간과 비슷하다. 그러나 패턴의 개수가 많을 때는 상대적으로 일치하는 문자열이 많아지고 이에 따라 조건에 의하여 DFA-AC 알고리즘으로의 전이가 많아지므로 DFA-AC 알고리즘의 패턴매칭 수행시간과 비슷해진다. 패킷데이터 2의 경우 패턴의 개수가 200개 이하인 그룹에서는 비교한 문자열길이가 10 이상인 패턴이 존재하지 않기 때문에 DFA-AC 알고리즘으로 전이가 일어나지 않기 때문에 혼합형 다중패턴매칭 기법의 성능이 MWM 알고리즘의 성능과 같은 것을 확인할 수 있다.

조건 1과 2에 의해 전이되는 혼합형 다중패턴매칭 기법의 패턴매칭 속도는 형태가 비슷하다. 그러나 조건 3을 적용한 경우에는 패킷데이터 1과 같이 최악의 경우가 아닐 때에는 성능이 좋은 쪽으로 수렴하지만 〔그림 11〕(b)의 패턴의 개수가 200 이하일 때처럼 MWM 알고리즘에서 DFA-AC 알고리즘으로 전이가 전혀 일어나지 않아서 성능이 개선되지 않는 그룹이 존재한다. 따라서 일반적으로 보았을 때 조건 3보다는 조건 1이나 조건 2가 패턴의 개수가 다양한 그룹들로 이루어진 규칙집합의 패킷검사 속도를 향상시킬 것으로 기대할 수 있다.

4.2 임계값 변경에 따른 성능시험

제안한 기법은 전이조건의 임계값에 의하여 패턴매칭 알고리즘이 전이하므로 임계값이 성능에 영향을 미칠 수 있다. 즉, 임계값을 결정하는 방법 혹은 그 영향에 대한 평가가 본 논문에서의 전이조건이 유효함을 보이는데 반드시 필요하다. 본 절에서는 임계값이 성능에 미치는 영향을 살펴보기 위하여 임계값을 변경했을 때의 수행시간을 측정하였다. 여기서는 조건 2를 적용하고 임계값을 2000, 10000, 50000으로 변경하면서 패턴매칭 수행시간의 변화를 확인하도록 한다.

〔그림 12〕에서 보는 것과 같이 임계값 TH_{hash}를

작게 하면 MWM 알고리즘에서 DFA-AC 알고리즘으로 전이가 많아져 DFA-AC 알고리즘의 성능에 비슷해지고, 반대로 임계값 TH_{hash}를 크게 하면 MWM 알고리즘에서 처리하는 패킷이 많아져 MWM 알고리즘의 패턴매칭 속도와 비슷해진다. 이러한 현상은 패킷데이터 1을 사용한 경우와 패킷데이터 2를 사용한 경우에 모두 공통적으로 나타났다.

그러나 대체적으로 보았을 때 임계값을 변화시켰을 때의 영향이 크지 않았다. 즉, 적당한 임계값을 알 수 없더라도 두 패턴매칭 기법 사이의 전이 형태는 유사하게 나타날 것으로 기대할 수 있다. 이것은 임계값을 결정하는 방법에 의한 영향을 최소화할 수 있다는 점에서 고무적이다. 조건 1과 조건 3의 경우도 임계값을 변경할 때의 시험 결과가 이와 비슷했다.

V. 결 론

본 논문에서는 유한 오토마타를 이용하는 다중패턴매칭 기법과 시프트테이블을 이용하는 다중패턴매칭 기법을 결합할 경우 각 기법에서의 취약점을 효과적으로 회피할 수 있음을 보였다. 시프트테이블을 이용하여 패턴매칭 기법을 설계하면 평균적인 성능이 우수하다는 점에서 바람직하지만, 동일한 문자가 반복적으로 나타나는 문자열에 대하여, 즉 최악의 입력에 대하여 패턴매칭 속도의 하락 정도가 매우 크다.

본 논문에서 제시된 혼합형 다중패턴매칭 기법은 이러한 문제점을 극복하기 위하여 시프트테이블 기반의 패턴매칭 기법을 기본 패턴매칭 알고리즘으로 취하고 1 바이트 이상 시프트하며 페이로드에서 패턴을 검색하되, 성능 저하 요인을 예측하도록 구성된 전이 조건을 검사하여 이 조건에 맞는 경우에는 유한 오토마타 기반의 패턴매칭 기법으로 전이하도록 설계되었다. 이러한 목적을 만족시킬 수 있도록 본 논문에서는 두 기법 사이의 전이에 필요한 조건으로 세 가지 대안을 제시하고 이러한 전이조건들이 두 기법 사이에서 최적의 조합을 효과적으로 찾아낼 수 있음을 보였다.

실제 패킷데이터를 적용한 실험 결과 제안된 혼합형 다중패턴매칭 기법이 최악의 경우를 효과적으로 회피하고 두 기법의 장점을 잘 결합할 수 있음을 알 수 있었다. 즉 검사해야 할 다양한 종류의 입력 형태에 대하여 제안된 혼합형 기법이 무난한 성능을 보이는 것으로 나타났다. 한편, 전이조건을 구성하는 임계값의 선택 문제가 제안된 기법의 성능에 중요한 변수가 아님도 확인할 수 있었는데, 이는 제안된 기법이 특별히 제한된 경우에만 유효한 것이 아님을 입증한다.

Snort와 같이 패킷검사 목적으로 다중패턴매칭 기법을 활용하는 사례들을 살펴보면, 최근에는 시프트테이블을 사용하는 대신 이전처럼 유한 오토마타를 사용하는 기법으로 회귀하는 추세를 볼 수 있는데 이는 시프트테이블 방식이 갖는 최악 성능에서 비롯된 측면이 크다고 볼 수 있다. 본 논문에서 제시된 방법은 이러한 최악 성능 문제를 효과적으로 회피할 수 있는 대안으로 활용될 수 있다.

참고문헌

[1] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," Proceedings of the 13th USENIX conference on system administration, Nov. 1999.

[2] S. Antonatos, K.G. Anagnostakis, and E.P. Markatos, "Generating realistic workloads for network intrusion detection systems," ACM Workshop on Software and Performance, 2004.

[3] R.S. Boyer and J.S. Moore, "A Fast String Searching Algorithm," Communications of the ACM, 1977.

[4] A.V. Aho and M.J. Corasick, "Efficient String Matching: An Aid to Bibliographic search," Communications of the ACM, 18 June 1975.

[5] Y. Jianming, X. Yibo and L. Jun, "Memory Efficient String Matching Algorithm for Network Intrusion Management System," Proceedings of Global Telecommunications Conference, pp. 1-5, 2006.

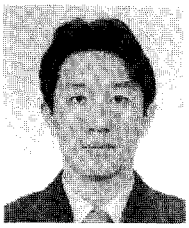
[6] Marc Norton, "Optimizing Pattern Matching for Intrusion Detection," Technical Paper, (http://docs.ids research.org/OptimizingPatternMatchingForIDS.pdf, 2004).

[7] Andrew R. Baker and Joel Esler, Snort IDS and IPS Toolkit, Syngress, Mar. 2007.

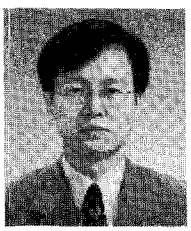
[8] Toshino Nishimura, Shuichi Fukamachi and Rakesi Shinohara, "Speed-up of Aho-Corasick Pattern Matching Machines

by Rearrangin States," String Processing and Information Retrieval, 2001.
[9] Nathan Tuck, Timothy Sherwood, Brad Calder and George Varghese, "Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection," IEEE INFOCOM 2004, 2004.
[10] S. Wu and U. Manber, "A Fast Algorithm for Multi-Pattern Searching", Technical Report TR 94-17, University of Arizona at Tuscan, May. 1994.
[11] Yang Dong Hong, Xu Ke and Cui Yong, "An Improved Wu-Manber Multiple Patterns Matching Algorithm," IPCCC 2006, 2006.
[12] Sunday DM, "A Very Fast Substing Search Algorithm," in Communication of the ACM, 1990.
[13] S. Wu, U. Manber, "AGREP - a fast approximate pattern-matching tool," Proceedings of USENIX Technical Conference, pp. 153-162, 1992.
[14] DEFCON 9 CTF Data file. http://www.cs.ucsb.edu/~seclab/datasets/defcon/ctf9/.
[15] G. Combs, et al., Wireshark, 2007 (http://www.wireshark.org).

〈著者紹介〉



이 재 국 (Jae-Kook Lee) 학생회원
2002년 2월: 충남대학교 컴퓨터과학과 졸업
2004년 2월: 충남대학교 컴퓨터과학과 석사
2004년 3월~현재: 충남대학교 컴퓨터공학과 박사과정
〈관심분야〉 인터넷보안



김 형 식 (Hyong-Shik Kim) 종신회원
1988년 2월: 서울대학교 컴퓨터공학과 졸업
1990년 2월: 서울대학교 컴퓨터공학과 석사
1997년 8월: 서울대학교 컴퓨터공학과 박사
1997년 12월~1999년 8월: 미국 UAH, Faculty Research Associate
1999년 9월~현재: 충남대학교 컴퓨터공학과 교수
〈관심분야〉 인터넷보안, 컴퓨터시스템구조