DOI: 10.3745/KIPSTC.2009.16-C.4.449

고성능 네트워크 침입방지시스템을 위한 개선된 시그니처 해싱 알고리즘

고 중 식[†] · 곽 후 근[††] · 왕 정 석[†††] · 권 희 웅[†††] · 정 규 식[††††]

요 약

시그니처 해싱 알고리즘[9]은 해시 테이블을 사용하여 네트워크 침입방지시스템(Intrusion Prevention System)을 위한 빠른 패턴 매칭 속도를 제공한다. 시그니처 해싱 알고리즘은 모든 규칙에서 2 바이트를 선택하여 해쉬 값을 구한 후 해쉬 테이블에 링크시킨다. 이렇게 하여 패턴 매칭 시에 실제 검사하는 규칙의 개수를 줄임으로써 성능이 향상되는 장점을 가진다. 그러나 규칙의 개수와 상관관계가 증가할 경우 같은 해쉬 값을 가지는 규칙의 개수가 증가하여 성능이 떨어지는 단점이 있다.

본 논문에서는 시그니처 해싱 알고리즘의 단점을 보완하기 위해 규칙의 개수와 상관관계에 무관하게 모든 규칙을 해쉬 테이블 상에 고르게 분포시키는 방법을 제안한다. 제안된 방법에서는 해쉬 테이블에 규칙을 링크하기 전에 해당 해쉬 값에 링크된 규칙이 있는지 검사한다. 만약 링크된 규칙이 없으면 해당 해쉬 값에 규칙을 링크하고, 링크된 규칙이 있으면 다른 위치에서 해쉬 값을 다시 계산한다. 제안한 방법은 리눅스 커널 모듈 형태로 PC에서 구현하였고, 네트워크 성능 측정 툴인 Iperf를 이용하여 실험하였다. 실험 결과에 의하면 기존 방식에서는 시그니처 개수 및 규칙의 상관관계가 증가함에 따라 성능이 저하되었지만, 본 논문에서 제안한 방식은 시그니처 개수와 규칙의 상관관계에 무관하게 일정한 성능을 유지하였다

키워드 : 침입 차단 시스템, 시그니처 해싱, 패턴 매칭 알고리즘, 해시 테이블

An Improved Signature Hashing Algorithm for High Performance Network Intrusion Prevention System

Joongsik Ko[†] · Hukeun Kwak[††] · Jeongseok Wang[†††] · Huiung Kwon[†††] · Kyusik Chung[††††]

ABSTRACT

The signature hashing algorithm[9] provides the fast pattern matching speed for network IPS(Intrusion Prevention System) using the hash table. It selects 2 bytes from all signature rules and links to the hash table by the hash value. It has an advantage of performance improvement because it reduces the number of inspecting rules in the pattern matching. However it has a disadvantage of performance drop if the number of rules with the same hash value increases when the number of rules are large and the corelation among rules is strong.

In this paper, we propose a method to make all rules distributed evenly to the hash table independent of the number of rules and corelation among rules for overcoming the disadvantage of the signature hashing algorithm. In the proposed method, it checks whether or not there is an already assigned rule linked to the same hash value before a new rule is linked to a hash value in the hash table. If there is no assigned rule, the new rule is linked to the hash value. Otherwise, the proposed method recalculate a hash value to put it in other position. We implemented the proposed method in a PC with a Linux module and performed experiments using Iperf as a network performance measurement tool. The signature hashing method shows performance drop if the number of rules with the same hash value increases when the number of rules are large and the corelation among rules is strong, but the proposed method shows no performance drop independent of the number of rules and corelation among rules.

Keywords : IPS(Intrusion Prevention System), Signature hashing, Pattern matching Algorithm, Hash table

1. 서 론

최근 수년간 컴퓨터 처리능력의 비약적인 발전으로 더욱 다양하고 많은 양의 서비스와 정보를 빠른 시간에 이용할 수 있게 되었다. 하지만 이런 발전과 함께 인터넷을 통한 악의적인 공격도 점점 다양해지고 있다. 이러한 인터넷을

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
† 정 회 원 : 숭실대학교 정보통신전자공학부 석사과정
†† 정 회 원 : 숭실대학교 정보통신전자공학부 postdoc(교신저자)
††† 정 회 원 : 숭실대학교 정보통신전자공학부 박사과정
†††† 정 회 원 : 숭실대학교 정보통신전자공학부 교수
논문접수 : 2009년 1월 22일
수 정 일 : 1차 2009년 3월 16일
심사완료 : 2009년 3월 19일

통한 공격에 의한 피해가 늘어나면서 보안에 대한 중요성이 커지고 있으며, 이에 따라 여러 가지 보안 기술과 장비들이 개발되어 사용되고 있다. 대표적인 보안 장비로 방화벽(Firewall), 침입 탐지 시스템(IDS: Intrusion Detection System), 침입 방지 시스템(IPS: Intrusion Prevention System), 그리고 최근 중요성이 커지고 있는 웹 방화벽(Web Application Firewall)이 있다. 보안 장비는 인터넷을 통해 전송되는 패킷들 중 자신을 통과하는 패킷을 캡처하여 패킷 내부에 악의적인 공격과 관련된 패턴이 있는지 검사하여 공격 패턴이 발견될 경우 로그를 남기거나 실시간으로 차단함으로써 외부로부터의 공격이나 침입으로부터 자신의 시스템을 보호하는 기능을 수행한다[1, 2]. 이때 패턴 매칭에 사용되는 알고리즘으로 KMP(Knuth-Morris-Pratt)[3], BM (Boyer-Moore)[4]와 같은 싱글 패턴 매칭 방식과 Wu-Manber[5], Aho-Corasick[6] 등과 같은 멀티 패턴 매칭 방식이 있다. 이들 장비들은 서로 검사를 수행하는 Layer나 방법, 그리고 공격 및 침입이 탐지되었을 때 수행하는 동작이 다르기 때문에 여러 단계에 거쳐 중복되게 설치하거나 하나의 장비에 방화벽, IDS/IPS, 웹 방화벽의 기능을 포함시키는 통합보안장비로 이어지고 있다[7].

대표적인 침입탐지 프로그램인 Snort[8]에서는 패킷의 페이로드에 공격패턴이 있는지 검사하는 방법으로 Aho-Corasick 알고리즘을 사용하여 패킷의 검사를 수행한다. Snort에서 패턴 매칭에 사용하고 있는 Aho-Corasick 알고리즘은 패킷 검사에 사용할 규칙에 포함된 패턴 정보들을 트리 구조로 구성하여 패킷의 페이로드를 처음부터 끝까지 한번만 검사하는 것으로 해당 패킷이 악의적인 패킷인지 정상 패킷인지 확인할 수 있다. 하지만 Aho-Corasick의 경우 Snort에서 사용하는 규칙의 개수가 증가함에 따라 패킷의 검사 속도가 느려지는 단점이 있다. 특히 각 규칙에 포함된 패턴들 사이의 상관관계에 따라 패킷 검사 속도에 큰 차이를 보인다. Aho-Corasick 알고리즘의 이런 특징은 실시간으로 검사 및 차단을 통해 패킷의 통과 여부를 결정해야하는 침입방지시스템(IPS) 모드로 동작하는 경우에 커다란 단점으로 작용될 수 있다[6, 8].

위와 같은 Aho-Corasick 알고리즘의 단점을 보완하여 제안된 알고리즘이 시그니처 해싱 알고리즘이다[9]. 시그니처 해싱 알고리즘은 모든 규칙에서 해시 값을 구해 모든 규칙을 해시 테이블을 사용하여 분산시킨 후 패킷을 검사할 때 페이로드에서 해시 값을 구해 해당 해시 값에 링크된 규칙만 검사하는 것으로 전체적으로 실제 비교되는 규칙의 개수를 줄여 패턴 매칭 성능을 향상시키는 방법이다. 이는 규칙의 개수가 늘어나더라도 해시 테이블에 의해 규칙이 분산되기 때문에 규칙의 개수 증가에 따른 성능저하가 적고, 규칙의 상관관계가 적을 경우에도 빠른 패턴 매칭 속도를 보인다[9]. 하지만 규칙의 상관관계가 높을 경우 하나의 해시 값에 많은 수의 규칙이 링크되어 해당 해시 값에 대한 순차 매칭를 하는 시간이 길어져 성능 저하가 발생하는 단점을 가지고 있다.

본 논문에서는 이러한 시그니처 해싱 알고리즘의 단점을 개선하여 해시 테이블에 규칙을 링크시킬 때 해당 해시 값에 링크된 규칙이 존재할 경우에 패턴에서 해시 값을 계산하는 위치를 바꿔서 링크된 규칙이 없는 해시 값이나 가장 적은 수의 규칙이 링크된 해시 값을 찾아 해당 규칙을 링크함으로써 해시 테이블에 규칙을 전체적으로 고르게 분산시켜서 성능을 개선하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 연구를 소개하고 기존 연구의 단점 및 접근 방식을 설명한다. 3장에서는 제안된 방식을, 4장에서는 실험 및 토론을 마지막으로 5장에서는 결론 및 향후 연구 방향을 설명한다.

2. 기존 연구

2.1 패턴 매칭

2.1.1 규칙과 시그니처

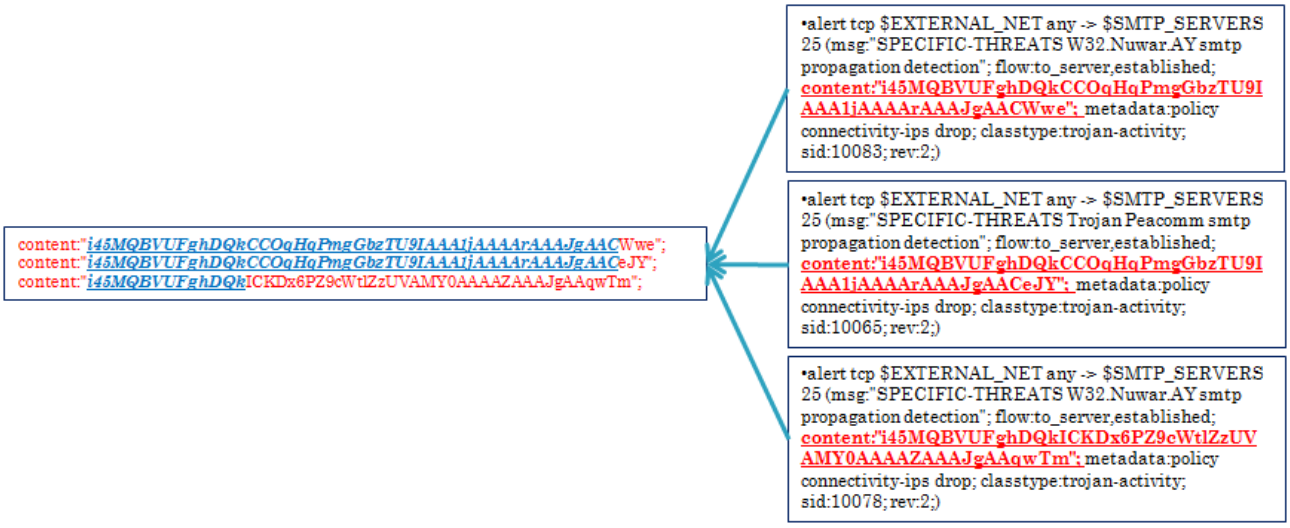
패턴 기반 IDS/IPS는 해당 공격에 대한 패턴을 이용하여 패킷을 검사한다. 정확한 패턴 매칭을 위해서는 공격에 해당하는 패턴에 대한 정확한 내용이 기술되어야 하고, 이를 포함하는 전체적인 내용을 규칙이라고 한다. 규칙에는 시그니처라고 불리는 패킷의 페이로드와 직접 비교할 패턴 정보와 시그니처가 적용될 서비스 포트, 규칙의 이름, 시그니처가 나타나는 위치, 비교 방식 등에 대한 상세한 정보가 포함되어 있다. 만약 이 규칙에 기술된 정보와 일치하는 패킷이 들어올 경우 침입방지 시스템은 해당 패킷을 공격 또는 침입 패킷으로 판단하고 그에 따른 조치를 취할 수 있다[9].

2.1.2 규칙의 상관관계

패턴 매칭에 사용되는 규칙들의 시그니처가 일치하는 부분의 정도에 따라 규칙의 상관관계를 정의한다. 시그니처가 일치하는 부분이 많을수록 규칙의 상관관계가 높다고 하고, 일치하는 부분이 적을수록 상관관계가 낮다고 한다. 규칙의 상관관계는 변종 웜, 비슷한 유형의 공격에 대한 시그니처를 정의할 경우에 많이 발생하며, 멀티 패턴 매칭을 수행할 경우 패턴 매칭 성능에 큰 영향을 미친다. (그림 1)은 규칙에 대한 상관관계의 예를 보여주는 그림이다. 이는 Snort에서 사용되고 있는 규칙의 일부로써 오른쪽은 각 규칙의 전체 내용을 포함하고, 왼쪽은 각 규칙의 시그니처 부분을 표시한 것이다. 각 규칙의 시그니처를 살펴보면 왼쪽의 밑줄이 있는 부분과 같이 시그니처 사이에 일치하는 내용이 존재한다. 이렇게 각 시그니처 사이에 일치하는 부분으로 규칙의 상관관계를 정의한다.

2.2 Aho-Corasick 알고리즘

대표적인 open source IDS 프로그램인 Snort 에서는 여러 가지 방법을 통해 해당 패킷이 정상적인 패킷인지 아니면 악의적인 패킷인지를 검사한다. 그 중 패킷의 페이로드에 공격패턴이 있는지 검사하는 방법으로 Aho-Corasick 알

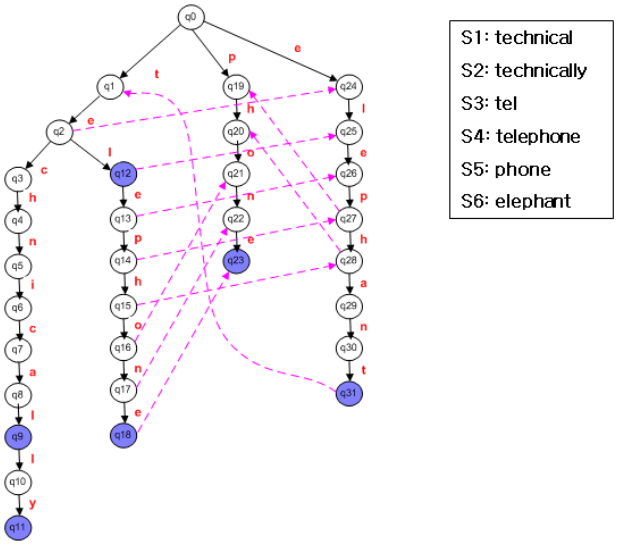


(그림 1) 규칙의 상관관계의 예

고리즘을 사용하여 패킷의 검사를 수행한다[6].

Snort에서 패턴 매칭에 사용하고 있는 Aho-Corasick 알고리즘은 패킷 검사에 사용할 규칙에 포함된 패턴 정보들을 (그림 2)와 같이 트리 구조로 구성하여 패킷의 페이로드를 처음부터 끝까지 문자단위로 트리의 각 단계별로 비교하면서 전체 페이로드를 한번만 검사하는 것으로 해당 패킷이 악의적인 패킷인지 정상 패킷인지 확인할 수 있다[10].

(그림 3)은 Jump Aho-Corasick 알고리즘으로 Aho-Corasick 알고리즘의 순차매칭에 의해 발생하는 성능 저하를 개선하기 위해 k 바이트 단위로 각 노드를 구성하여 매칭을 수행한다. Jump Aho-Corasick 알고리즘은 k 바이트 단위로 점프하면서 매칭을 수행하기 때문에 Aho-Corasick 알고리즘의 메모리 접근 증가 및 1 바이트 단위의 매칭으로 인한 병목 현상을 완화시킴으로써 매칭 속도를 증가시키는 방식이다[11-13].



(그림 2) Aho-Corasick trie 구조

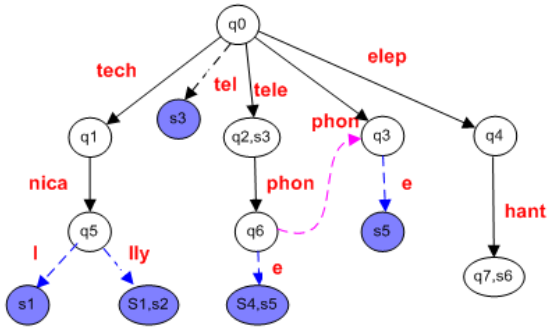
Aho-Corasick 알고리즘은 Snort와 Apache 웹 서버와 연동하여 사용할 수 있는 웹 방화벽인 mod-security[14, 15]에서 패턴 매치에 사용하는 방법이다. 멀티 패턴 매칭 방법의 대표적인 알고리즘으로써 빠른 검사 속도와 패킷의 페이로드를 한번만 검사하는 것으로 모든 패턴과 비교를 할 수 있다는 장점을 가지고 있다. 하지만 Aho-Corasick의 경우 사용하는 규칙의 개수가 증가함에 따라 트리를 구성하는 노드의 수가 증가하여 패킷의 검사 속도가 느려지는 단점이 있다. 특히 패킷의 페이로드 내용에 따라서 같은 개수의 규칙에서도 큰 성능의 차이를 보인다. 그리고 각 규칙에 포함된 패턴들 사이의 상관관계에 따라 트리를 구성하는 가지의 수가 달라져서 패킷 검사 속도에 큰 차이를 보는 것이 단점으로 지적되고 있다[16].

2.3 시그니처 해싱 알고리즘

시그니처 해싱 알고리즘은 적용되는 모든 규칙의 내부에 존재하는 시그니처에 대해 이 중 2 바이트를 정해 해시 값을 만들고, 이 해시 값을 이용하여 검색하고자 하는 패킷 데이터와 비교하여 실제 확인해 볼 필요가 있는 시그니처만 비교하도록 하여 실제 비교하는 규칙의 개수를 줄이는 방법으로 패턴 매칭 성능을 향상시키는 방법이다. 각 포트별로 해시 테이블을 만들고 각 포트별로 해당하는 규칙에서 해시 값을 구하여 해당 해쉬 값에 규칙을 링크 시킨다. 패턴 매칭을 할 때 패킷 데이터의 페이로드에서 2 바이트단위로 1 바이트 윈도잉(Windowing)을 하면서 해쉬 값을 계산하여 해당 해쉬 값에 링크된 규칙만 실제로 비교를 하는 방식이다[9].

2.3.1 일반적인 해싱

(그림 4)는 시그니처 해쉬 테이블을 구성하는 예와 각 해쉬 테이블에 어떤 방식으로 규칙들이 맵핑되는지 보여준다. 패턴 매칭에 적용할 모든 규칙을 포트별로 분류하고, 각 포

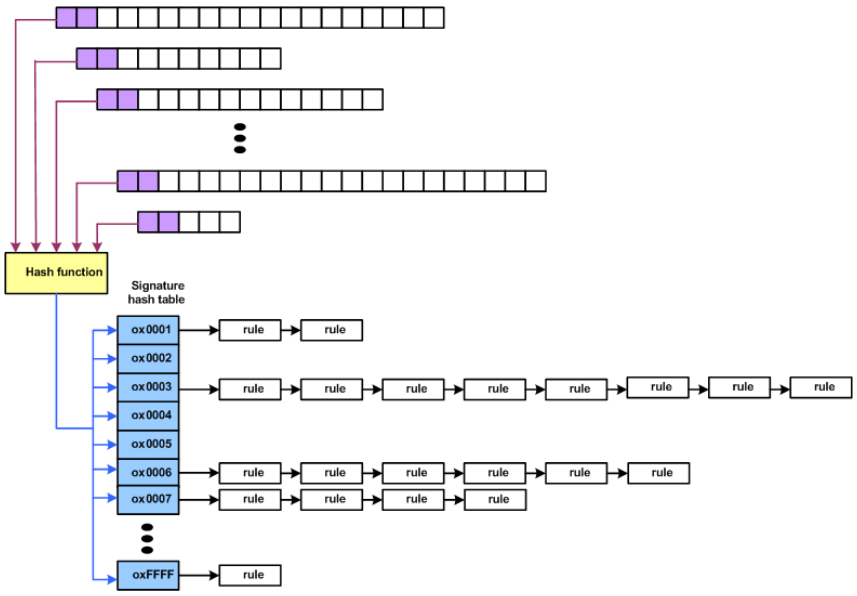


S1: technical
S2: technically
S3: tel
S4: telephone
S5: phone
S6: elephant

S1: technical
S2: technically
S3: tel
S4: telephone
S5: phone
S6: elephant

$\langle state, substring \rangle$	Next State	Matching Strings	failure chain
$\langle q_0, tech \rangle$	q_1	NULL	q_0
$\langle q_0, tele \rangle$	q_2	s_3	q_0
$\langle q_0, phon \rangle$	q_3	NULL	q_0
$\langle q_0, elep \rangle$	q_4	NULL	q_0
$\langle q_1, nica \rangle$	q_5	NULL	q_0
$\langle q_2, phon \rangle$	q_6	NULL	q_3, q_0
$\langle q_4, hant \rangle$	q_7	s_6	q_0
$\langle q_0, tel \rangle$	NULL	s_3	NULL
$\langle q_3, e \rangle$	NULL	s_5	NULL
$\langle q_5, l \rangle$	NULL	s_1	NULL
$\langle q_5, ly \rangle$	NULL	s_2, s_1	NULL
$\langle q_6, e \rangle$	NULL	s_4, s_5	NULL

(그림 3) Jump Aho-Corasick trie 구조



(그림 4) 시그니처 해쉬 테이블 구조

트에 해당하는 각 규칙들의 시그니처에서 첫 2 바이트를 선택하여 해쉬 값을 계산하고 해당 해쉬 값에 규칙의 상세정보를 연결 리스트(Linked List)를 이용하여 연결한다. 연결 리스트를 이용하여 등록되는 규칙의 상세 비교 정보는 시그니처의 전체 내용, 패킷에서 해당하는 시그니처가 존재하는 위치 등 직접 비교에 사용되는 내용을 포함한다.

이렇게 구성된 시그니처 해쉬 테이블을 이용하여 실제 패킷이 들어왔을 경우 패킷의 데이터를 비교, 검사하는 기준이 된다. 해쉬 값을 사용하여 검사를 하는 경우 해당 해쉬 값이 일치하는지 매칭하는데 걸리는 소요시간이 일반적인 텍스트 매칭에 비해 짧고, 각 해쉬 값에 연결되는 규칙의 개수가 줄어듦에 따라 패킷 데이터에 대해 규칙과 매치되는 경우의 수는 줄어들어 매칭 속도를 향상시킬 수 있다.

2.3.2 컨텐츠 교정 해싱

시그니처 해싱 알고리즘은 패킷에 매칭을 수행하는데 있어서 실제 매칭되는 규칙의 수를 줄임으로써 패턴 매칭 성능을 향상시키고 적용되는 규칙의 개수와 관계없이 빠른 패턴 매칭 성능을 보장한다. 하지만 최근 급증하고 있는 웜의 경우 그에 대한 변종이 계속 나타남에 따라 시그니처의 변

화가 크지 않은 시그니처가 많아지고 있다. 이 때 일반적인 해싱 알고리즘을 사용하여 해쉬 테이블을 구성할 경우 하나의 해쉬 값에 많은 수의 규칙이 연결되게 된다. 결국 많은 수의 규칙이 연결된 해쉬 값이 나올 경우 패턴 매칭을 수행함에 있어서 시그니처 전체를 비교해야 하는 규칙의 수가 많아진다는 것을 나타낸다. 이는 패턴 매칭 성능을 저하시키고 알고리즘의 성능 예측을 어렵게 만든다.

이러한 단점을 보완하기 위해 해싱값을 구하기 위한 2 바이트 부분을 시그니처의 첫 2 바이트가 아닌 임의의 부분(Random)을 선택하여 해싱값을 구하도록 제안하는 방법이 컨텐츠 교정 해싱(Contents-Correction Hashing)이다. 컨텐츠 교정 해싱은 랜덤하게 생성된 위치 값을 이용하여 컨텐츠 시작에서부터 그 위치 값만큼 떨어진 곳에서 선택한 2 바이트에 대하여 계산한 해싱값과 그 위치 값을 사용한다. 이 방법은 시그니처 간 유사성이 높은 경우에도 각각 해싱값이 다르게 나오도록 조정하는 역할을 수행하며, 이로 인해 시그니처 해쉬 테이블이 더 균일한 분포를 가질 수 있도록 한다. (그림 5)(a)는 컨텐츠 교정 해싱을 이용하여 구성한 해쉬 테이블의 구성을 나타내고, (그림 5)(b)는 컨텐츠 교정 해싱의 해쉬 테이블 구성 순서도를 보여준다. 컨텐츠 교정 해싱을 이용함으로써 규칙의 상관관계가 높더라도 실제 전체적으로 비교하는 규칙의 개수를 줄여 성능 저하를 줄일 수 있다.

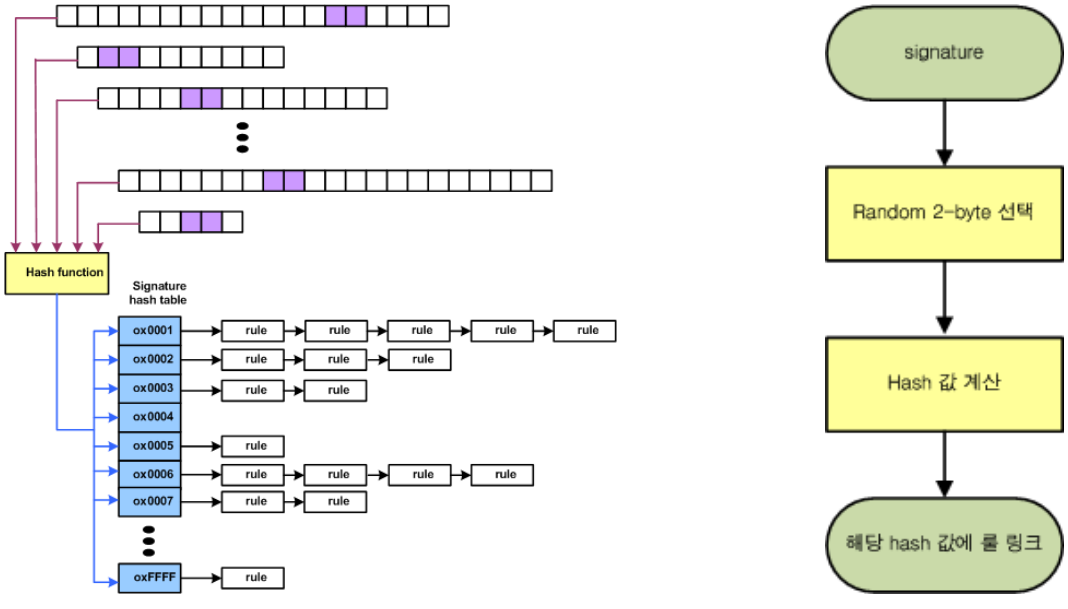
2.4 접근 방식

2.4.1 시그니처 해싱 알고리즘의 문제점

현재 보안장비에서 사용되고 있는 규칙은 변종 웜과 비슷한 유형의 공격을 하나의 그룹으로 구성하여 패턴 매칭을 위한 동작을 실행한다. 이에 따라 상관관계가 높은 규칙의 수가 많아지고 패턴 매칭 성능에 영향을 주게 된다. 시그니처 해싱 알고리즘은 규칙의 상관관계가 높을 경우 같은 해쉬 값을 가지는 규칙의 수가 증가하게 되어 패킷의 페이로드(Payload)에서 해쉬 값을 계산하여 패턴 매칭을 수행할 때 성능저하를 보이게 된다. (그림 6)은 4.1절의 실험 환경을 사용하여 나온 컨텐츠 교정을 사용하는 시그니처 해싱 알고리즘의 특성을 보여주는 그래프이다.

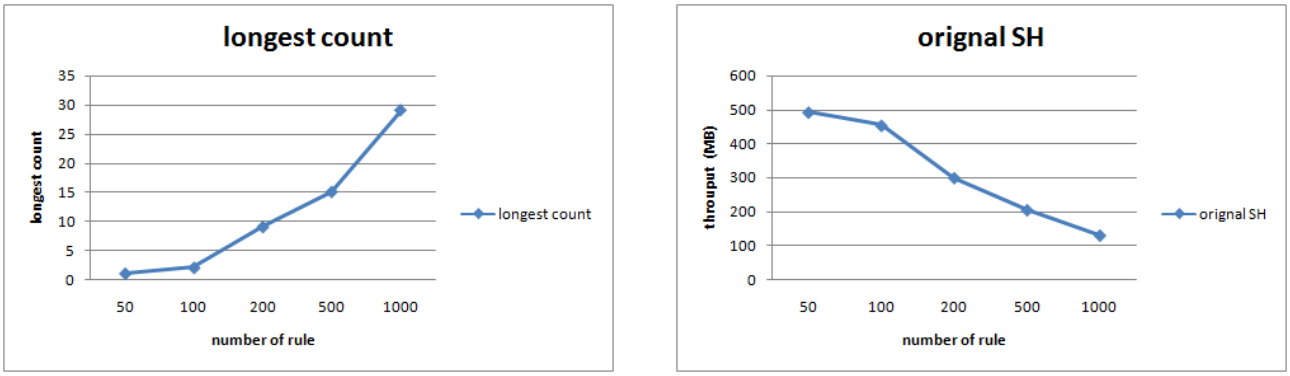
(그림 6)(a)에서 x 축은 규칙의 개수이고, y 축은 그에 따른 규칙의 longest count(동일한 해쉬 값을 가지는 규칙의 최대 개수)를 보여주는 그래프이다. 이 경우 전체 규칙의 10%에 해당하는 규칙이 동일한 포트를 가지고 있는 상황을 가정하여, 규칙의 개수를 50개, 100개, 200개, 500개, 1000개로 증가시키면서 규칙의 일부에 동일한 패턴을 넣었을 때 규칙의 longest count의 변화를 보여주는 그림이다. 규칙의 분포도를 높이기 위해 컨텐츠 교정 방식을 사용한다 하더라도 상관관계가 큰 규칙을 사용할 경우, 각 컨텐츠의 일부에서 얻는 해시 값이 같아질 확률도 높아지기 때문에 적용 규칙의 개수가 많아질수록 longest count 역시 증가하는 것을 볼 수 있다. 이는 규칙의 특정 해시 값으로 몰림으로 인해서 발생하게 되는데, 결과적으로 패턴 매칭 수행 시 전체 성능에 큰 영향을 주는 요인으로 작용하게 된다.

(그림 6)(b)는 위와 같이 해쉬 테이블이 구성된 환경에서 패턴 매칭을 수행했을 때 규칙의 개수에 따른 성능의 변화를 보여주는 그래프이다. x 축은 규칙의 개수를 나타내고, y 축은 규칙의 개수에 따른 패턴 매칭 성능을 나타낸다. 그림을 보면 규칙의 개수가 늘어날수록 패턴 매칭 성능이 떨어지는데 (그림 6)(a)의 longest count의 증가에 비례하여 성능의 저하가 크게 나타나는 것을 확인할 수 있다.



(a) (b)

(그림 5) 컨텐츠 교정 해싱을 이용한 해쉬 테이블 구성(a) 및 순서도(b)



(a) (b)

(그림 6) 규칙의 개수에 따른 규칙의 longest count(a) 및 성능 변화(b) (SH : Signature Hashing)

2.4.2 본 논문의 접근 방식

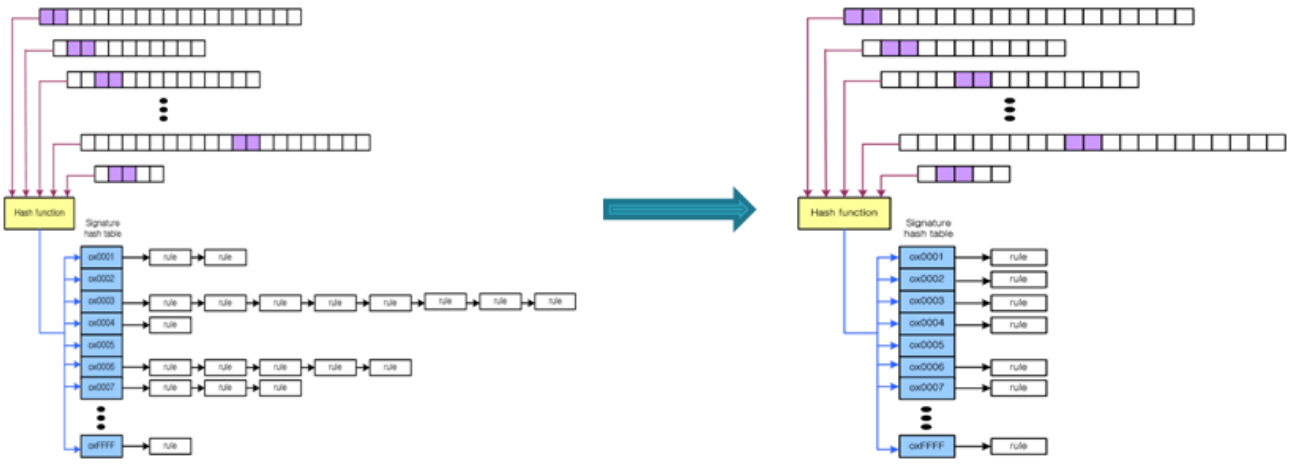
본 논문에서는 규칙에서 해쉬 값을 계산하여 해쉬 테이블을 구성할 때 해당 해쉬 값에 링크된 규칙의 개수를 검사한 후 링크된 규칙이 없거나 가장 적은 수의 규칙이 링크된 해쉬 값에 규칙을 링크시켜 모든 규칙이 해쉬 테이블 전체에 고르게 분포되도록 함으로써 규칙의 개수가 늘어나더라도 longest count의 수를 일정하게 유지시키는 방식을 제안한다. 이러한 방식을 사용하면 기존 시그니처 해싱에서 발생하는 하나의 해쉬 값에 많은 수의 규칙이 링크되는 문제를 해결할 수 있다.

3. 제안된 방식

3.1 전체 구조

제안된 방식에서는 해쉬 테이블을 구성할 때 계산된 해쉬 값에 링크된 규칙이 없는 해쉬 값이 나올 때까지 해쉬 값의 계산 위치를 바꾸어가며 반복하여 해쉬 값을 계산한다. 이러한 방법은 모든 규칙을 해쉬 테이블 전체에 고르게 분포시킴으로써 규칙의 개수에 상관없이 longest count를 일정하게 유지시켜 패턴 매칭 시 성능을 향상시킨다. (그림 7)은 기존 방식의 시그니처 해싱 알고리즘과 제안한 방식의 해쉬 테이블의 구성을 비교하여 보여주는 전체적인 그림이다.

제안된 방식에서는 시그니처의 처음부터 2 바이트 단위로 1 바이트 윈도잉하면서 해쉬 값을 계산하는 과정을 규칙이 링크되어 있지 않은 해쉬 값이 나올 때까지 반복하여 해당 해쉬 값에 규칙을 링크시키는 방식을 사용한다. 만약 시그니처의 맨 끝까지 윈도잉이 끝날 때까지 규칙이 링크되지 않은 해쉬 값을 구하지 못했을 경우에는 지금까지 계산된 해쉬 값 중에서 가장 적은 수를 가지는 규칙의 링크된 해쉬 값에 규칙을 링크시킨다. 이렇게 함으로써 (그림 7)(a)에서 보이는 것과 같이 해쉬 테이블을 구성할 때 규칙의 시그니처에서 2 바이트를 선택하여 계산된 해쉬 값에 무조건 해당 규칙을 링크시켜 규칙의 상관관계가 있을 경우 하나의 해쉬 값에 많은 수의 규칙이 링크되는 기존 시그니처 해싱 알고리즘의 단점을 보완하는 방식이다. 이와 같은 방법을 통해 상관관계가 있는 규칙에서도 서로 다른 해쉬 값을 구할 수 있기 때문에 하나의 해쉬 값에 많은 수의 규칙이 링크되는 것을 막을 수 있다. 이는 (그림 7)(b)와 같이 보안 장비에



(a) (b)

(그림 7) 해쉬 테이블 구조 - 기존 시그니처 해싱 알고리즘(a)과 제안된 방식(b)

적용할 모든 규칙을 해쉬 테이블 전체에 고르게 분포시킴으로써 longest count를 줄이고, 이를 통해 규칙의 개수가 늘어나고 상관관계가 있는 규칙이 많더라도 높은 성능의 패턴 매칭을 수행할 수 있도록 한다.

3.2 동작 과정

제안된 방식의 동작 과정은 (그림 8)과 같다.

(그림 8)(a)는 기존 시그니처 해싱 알고리즘의 동작 과정으로 해쉬 테이블에 추가할 규칙의 시그니처에서 랜덤하게 2 바이트를 선택하여 해쉬 값을 계산한 후 해쉬 테이블의 해당 해쉬 값에 규칙을 링크시킨다. 하지만 이렇게 해쉬 테이블을 구성할 경우 하나의 해쉬 값에 많은 수의 규칙이 링크되어 성능 저하가 발생하는 원인이 되기 때문에 본 논문에서는 (그림 8)(b)와 같은 방식을 제안하였다.

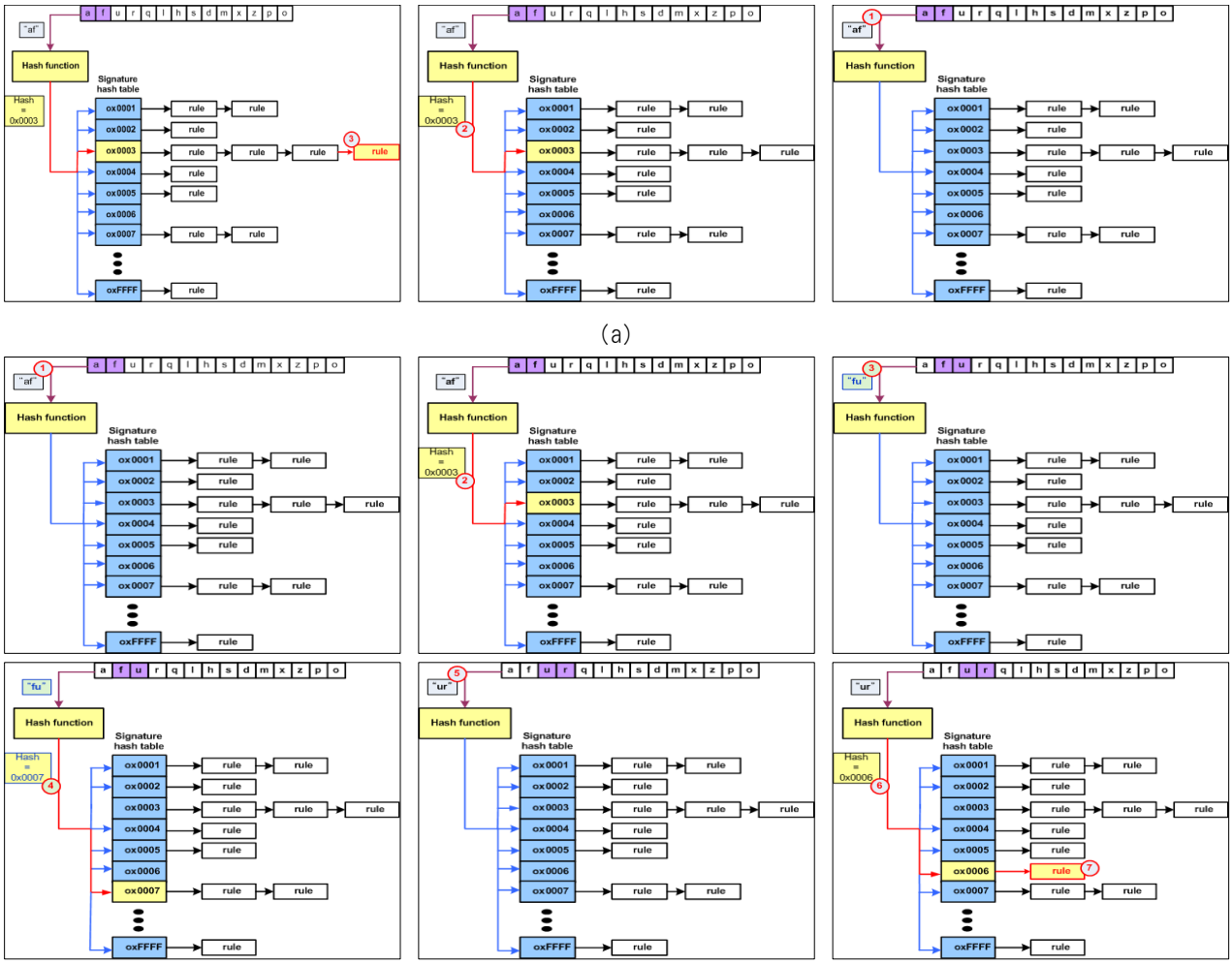
(그림 8)(b)는 제안된 방식의 동작 과정을 보여주고 있다. 해쉬 테이블에 추가할 규칙의 시그니처의 처음부터 2 바이트 단위로 선택하여 해쉬 값을 계산한 후 해당 해쉬 값에 링크된 규칙이 있는지 확인한다. 만약 계산된 해쉬 값에 링크된 규칙이 없을 경우 해당 해쉬 값에 규칙을 링크시키고, 링크된 규칙이 있을 경우에는 시그니처에서 해쉬 값의 계산 위치를 1 바이트 윈도잉을 하면서 해쉬 값을 계산하는 과정을 규칙이 링크되어 있지 않은 해쉬 값이 나올 때까지 반복하여 해당 해쉬 값에 규칙을 링크시키는 방식을 사용한다. 만약 시그니처의 맨 끝까지 윈도잉을 하면서 규칙이 링크되지 않은 해쉬 값을 구하지 못했을 경우에는 지금까지 계산된 해쉬 값 중에서 가장 적은 수의 규칙의 링크된 해쉬 값에 규칙을 링크시킴으로써 하나의 해쉬 값에 최소한의 규칙을 링크시키는 방식이다.

제안된 방식의 해쉬 테이블 구성 과정을 정리하면 다음과 같다.

단계 1. 해쉬 테이블에 추가할 규칙에서 시그니처의 처음 2 바이트를 선택한다.

단계 2. 선택한 2 바이트의 해쉬 값을 계산하여 해쉬 테이블에서 해당 해쉬 값에 링크된 규칙이 있는지 검사한다.

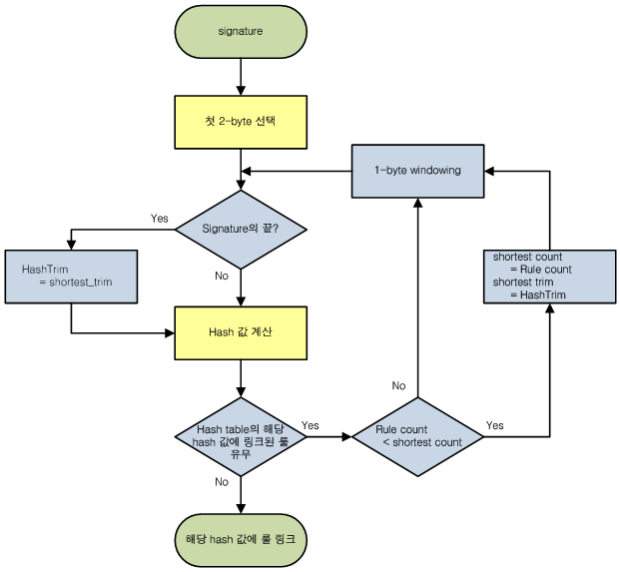
단계 3.



(그림 8) 해쉬 테이블 구성 방식 - 기존 시그니처 해싱 알고리즘(a)과 제안된 방식(b)

3-1. 링크된 규칙이 없을 경우 해당 해쉬 값에 규칙을 링크한다.
3-2. 링크된 규칙이 있을 경우 링크된 규칙이 없는 해쉬 값이 나올 때까지 시그니처에서 해쉬 값을 구하는 위치를 1 바이트씩 이동하며 2 바이트를 선택하여 단계 2부터 반복한다.
단계 4. 시그니처의 끝까지 링크된 규칙이 없는 해쉬 값이 존재하지 않을 경우 가장 적은 수의 규칙이 링크된 해쉬 값에 해당 규칙을 링크한다.

(그림 9)는 제안된 방식의 순서도를 나타낸다. 이와 같은 과정을 통해 모든 규칙을 해쉬 테이블에 전체적으로 고르게 분포시켜 longest count를 줄일 수 있다. 제안된 방식은 규칙의 개수가 늘어나고 상관관계가 존재하더라도 항상 해쉬 테이블 전체에 규칙을 고르게 분포시킴으로써 패턴 매칭 시 성능에 영향을 적게 받도록 기존 시그니처 해싱 알고리즘을 보완하였다.



(그림 9) 제안된 방식의 해쉬 테이블 구성 순서도

3.3 정성적 비교

<표 1>은 기존 방식과 제안된 방식을 해쉬 값을 계산하는 위치의 선택 방식, 해쉬 테이블에 규칙을 링크시키는 방식 및 해쉬 테이블에서 규칙의 분포도의 관점에서 정리한 것이다. 제안된 방식은 해쉬 값을 계산할 때 시그니처의 앞에서부터 2 바이트 단위로 해쉬 값을 계산하여 링크된 규칙이 없는 해쉬 값이 나올 때까지 1 바이트 윈도잉하면서 해쉬 값의 계산을 반복하여 해당 해쉬 값에 규칙을 링크시킨다. 기존 방식은 시그니처에서 랜덤하게 2 바이트를 선택하여 해쉬 값을 계산한 후 해당 해쉬 값에 무조건 규칙을 링크시킨다. 기존 방식에서는 계산된 해쉬 값에 무조건 규칙을 링크시키기 때문에 규칙에 상관관계가 있을 경우 같은 해쉬 값이 나올 가능성이 높아져 규칙이 하나의 해쉬 값에 몰려 분포도가 낮아진다. 하지만 제안된 방법은 시그니처의 처음부터 해쉬 값을 계산하여 계산된 해쉬 값에 링크된 규칙이 있는지 검사한 후 링크된 규칙이 없으면 해당 해쉬 값에 규칙을 링크시킨다. 만약 링크된 규칙이 있으면 시그니처에서 해쉬 값을 계산하는 위치를 바꾸며 링크된 규칙이 없는 해쉬 값이 나올 때까지 위의 과정을 반복하고, 시그니처의 끝까지 링크된 규칙이 없는 해쉬 값이 나오지 않을 경우에는 가장 적은 수의 규칙이 링크되어 있는 해쉬 값에 규칙을 링크시킨다. 이와 같은 방법으로 제안된 방법에서는 해쉬 테이블에 전체적으로 규칙이 고르게 분포되어 규칙의 분포도를 높임으로써 성능 향상을 가져온다.

4. 실험 및 토론

4.1 실험 환경

실험은 시그니처 해싱 알고리즘을 사용하는 침입방지시스템 장비를 사용하여 실험하였다. 기존에 사용되고 있던 시그니처 해싱 알고리즘을 수정하여 침입방지시스템에 모듈로 설치한 후 기존 방식과 제안된 방식을 통해 longest count와 성능을 측정하였다. 성능 측정은 대표적인 네트워크 성능 측정 툴인 iperf를 사용하여 측정하였다. (그림 10)은 실험 환경을 보여주고 <표 2>는 실험에 사용한 장비와 각 장비의 사양을 보여주고 있다.

<표 3>은 실험에 사용한 규칙과 패킷의 페이로드에 대한 것을 보여준다. 본 실험에서는 규칙의 개수를 50개부터 1000개까지 증가시키면서 longest count와 성능의 변화를 측정하였다. 그리고 시그니처는 규칙의 상관관계의 형성을 위해 “aaaaa”를 인위적으로 포함시킨 규칙을 전체 규칙의 10%로

〈표 1〉 기존 방식과 제안된 방식의 차이 (정성적 비교)

	해쉬 값 계산 위치 선택 방식	해쉬 테이블에 규칙 링크 방식	해쉬 테이블의 규칙 분포도
기존 방식	랜덤 위치 선택	무조건 해당 해쉬 값에 링크	낮음 (하나의 해쉬 값에 많은 수의 규칙이 링크)
제안된 방식	첫 2 바이트부터 1 바이트 윈도잉	- 링크된 규칙이 없는 해쉬 값에 링크 - 모든 해쉬 값에 규칙이 링크되어 있을 경우 가장 적은 수의 규칙이 링크된 해쉬 값에 링크	높음 (하나의 해쉬 값에 적은 수의 규칙이 링크. 해쉬 테이블 전체에 고르게 분포)

(그림 10) 실험 환경

〈표 2〉 실험에 사용된 장비 및 사양

	client	server	침입방지시스템
OS	Linux-2.6.20.21	Linux-2.6.16.18	Linux-2.6.20.21
CPU	Dual Core 1.6GHz	Pentium4 3.2GHz	Dual Core 2.2GHz
소프트웨어	iperf-2.0.2	iperf-2.0.2	시그니처 해싱 적용

〈표 3〉 실험에 사용된 시그니처 및 패킷 페이로드 구성

규칙 개수	50	100	200	500	1000
시그니처	"aaaaa"로 상관관계가 있는 규칙 적용(각 규칙의 10%)				
패킷 페이로드	"aaaaa"가 포함된 패킷을 생성하여 전송				

구성하였다. 패킷의 페이로드는 규칙이 상관관계를 가지는 부분인 "aaaaa"를 포함한 패킷을 전송하여 성능을 측정하였다.

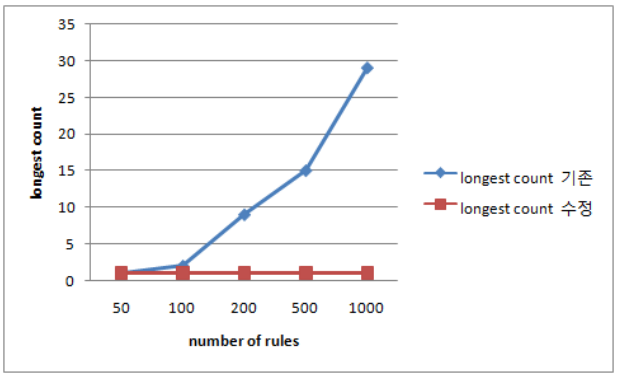
4.2 실험 결과

4.2.1 Longest Count

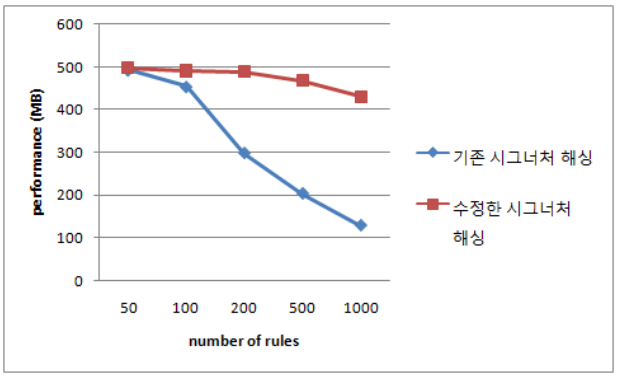
(그림 11)은 규칙의 개수에 따른 longest count의 변화를 보여준다. 기존 방법은 랜덤하게 선택한 2 바이트에서 해쉬 값을 계산한 후 무조건 계산된 해쉬 값에 규칙을 링크시키기 때문에 상관관계가 있는 규칙의 개수가 증가할수록 longest count 역시 증가하는 것을 볼 수 있다. 하지만 제안된 방법을 사용하면 규칙을 링크시키기 전에 먼저 해당 해쉬 값에 링크된 규칙이 있는지 확인한 후 링크된 규칙이 없는 해쉬 값이 나올 때까지 1 바이트 윈도잉을 하면서 해쉬 값을 다시 계산하여 해당 해쉬 값에 규칙을 링크하고, 만약 링크된 규칙이 없는 해쉬 값이 없을 경우에는 가장 적은 수의 규칙이 링크된 해쉬 값에 규칙을 링크시키기 때문에 상관관계가 있는 규칙의 개수가 늘어나더라도 longest count가 1로써 일정한 것을 볼 수 있다.

4.2.2 성능(Throughput)

(그림 12)는 규칙의 개수에 따른 기존 방식과 제안된 방식의 성능을 보여준다. 기존 방식의 경우 규칙의 개수가 증가함에 따라 (그림 11)에서와 같이 longest count가 증가하면서 패턴 매칭 성능이 큰 폭으로 떨어지는 것을 볼 수 있다. 반면에 제안된 방식은 규칙의 개수가 늘어나더라도 (그림 11)에서와 같이 longest count가 1로 일정하기 때문에 성



(그림 11) 규칙의 개수에 따른 longest count의 변화



(그림 12) 규칙의 개수에 따른 성능 비교

능이 떨어지는 폭이 적은 것을 확인할 수 있다. 이는 시그니처 해싱 알고리즘이 링크된 규칙을 순차 매칭을 통해 검사하는 방법을 사용하기 때문에 규칙의 개수가 늘어나면서 longest count가 증가하면 실제 일대일 매칭을 수행하는 규칙의 개수가 늘어나기 때문에 50개와 100개의 규칙을 사용할 때에는 차이가 많이 나지 않지만 100개 이상의 규칙을 사용할 경우에는 longest count의 차이가 크게 나타남으로써 성능의 차이도 크게 나타나는 것을 알 수 있다.

4.2.3 정량적 비교

(그림 13)은 규칙의 개수가 증가함에 따라 제안된 방식이 기존의 방식에 비해 성능이 좋아지는 정도를 보여주는 그래프이다. 규칙의 개수가 100개까지는 성능 향상의 폭이 크지 않지만 200개 이상이 되면 성능 향상 폭이 점점 커져 규칙이 1000개일 경우 제안된 방식이 기존 방식보다 333%의 성능 향상을 보인다. 이는 전체적으로 평균 약 186%의 성능 향상을 보이는 것으로 제안된 방식이 기존 방식에 비해 상관관계가 있는 규칙의 개수가 늘어나더라도 좋은 성능을 내는 것을 나타낸다.

(그림 13) 규칙의 개수에 따른 성능 향상도

4.3 토론

제안된 방식의 장점은 규칙의 개수와 상관관계에 관계없이 빠른 속도의 패턴 매칭을 수행함으로써 높은 성능을 제공할 수 있는 것이다. 패턴 매칭 시 기존 방식보다 모든 규칙을 해쉬 테이블에 고르게 분포시켜서 하나의 해쉬 값에 링크된 규칙의 수를 줄임으로써 패턴 매칭 시 실제 매칭을 수행하는 규칙의 수를 줄여 빠른 패턴 매칭 속도를 제공한다. 기존 시그니처 해싱 방식이 해쉬 테이블을 사용하여 이전에 사용되던 패턴 매칭 알고리즘(Aho-Corasick)에 비해 실제 매칭되는 규칙의 수를 줄여 빠른 성능을 제공했다. 하지만 규칙의 상관관계가 높을 경우 longest count가 증가하면서 성능이 저하되는 단점이 있다.

제안된 방식은 상관관계가 높은 규칙이 존재하더라도 서로 다른 해쉬 값을 구하여 모든 규칙을 해쉬 테이블에 전체적으로 고르게 분포시켜 longest count의 수를 줄이고 규칙의 개수가 증가하더라도 높은 성능을 제공할 수 있게 한다. 하지만 모든 규칙이 해쉬 테이블 전체에 분포되기 때문에 해쉬 테이블의 크기가 성능에 영향을 줄 수 있다. 해쉬 테이블의 크기가 규칙의 개수와 거의 같거나 작을 경우 해쉬 테이블 전체에 규칙이 링크되어 패턴 매칭 시 페이로드의 모든 위치에서 매칭되는 해쉬 값이 존재하게 되기 때문이다. 하지만 이 문제는 충분한 크기의 해쉬 테이블을 사용하면 해결할 수 있을 것이라고 생각한다.

5. 결론 및 향후 연구 방향

본 논문에서는 기존 시그니처 해싱 알고리즘에서 해쉬 테이블을 구성할 때 한 번 구해진 해쉬 값에 무조건 규칙을 링크시킴으로써 하나의 해쉬 값에 많은 수의 규칙이 링크되어 성능이 저하되는 문제를 해결하기 위하여 규칙을 링크시킬 때 링크된 규칙이 없는 해쉬 값이 나올 때까지 해쉬 값 계산을 반복하여 규칙을 링크시키는 방식을 제안하였다. 제안된 방식은 규칙의 상관관계가 높은 경우에 모든 규칙을 해쉬 테이블에 고르게 분포시킴으로써 규칙의 개수와 상관관계에 영향을 받지 않고 기존 방식에 비해 약 186% 향상된 높은 성능을 제공할 수 있다는 것을 확인하였다. 이는 대용량의 트래픽을 빠른 속도로 처리해야하는 환경에서 더욱 큰 효과를 가져올 수 있다.

향후 연구 방향으로는 해쉬 테이블의 크기와 관련된 연구이다. 제안된 방법은 해쉬 테이블 전체에 모든 규칙을 고르게 분포시키기 때문에 해쉬 테이블의 크기에 크게 영향을 받게 된다. 하지만 무조건 큰 해쉬 테이블을 사용하게 되면 메모리 낭비의 문제가 발생한다. 이런 단점을 보완하기 위해 해쉬 테이블 크기에 따른 성능의 변화를 분석하여 규칙의 개수에 따른 최적의 해쉬 테이블의 크기를 찾는 연구를 진행할 것이다.

참 고 문 헌

[1] 정보흠, 김정녀, 손승원, “침입방지시스템 기술 현황 및 전망”, 주간기술동향 통권 1098호, June, 2003.

[2] X. Zhang, C.Li, and W.Zheng, “Intrusion Prevention System Design”, Proceedings of the Fourth International Conference on Computer and Information Technology, Sep., 2004.

[3] Knuth DE, Morris JH and Pratt VB, “Fast pattern matching in strings”, SIAM Journal of Computing 1977.

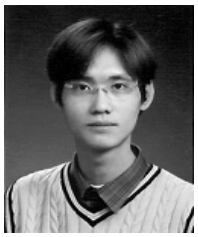
[4] Boyer RS and Moore JS, “A Fast String Searching Algorithm”, Communications of the ACM 1977.

[5] S.Wu and U. Manber. “A fast algorithm for multi-pattern searching.”, Technical Report TR-94-17, Department of Computer Science, University of Arizona, 1994.

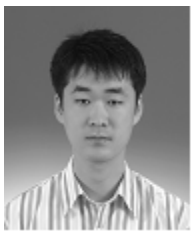
[6] A. Aho, M. Corasick, “Efficient string matching: an aid to biliographic search”, Comm. ACM. 18:333-40, 1975.

[7] 전용희, “침입방지시스템(IPS)의 기술 분석 및 성능평가 방안”, 정보보호학회지, 제15권, 제2호, Apr., 2005.
[8] Snort. http://www.snort.org/
[9] 왕정석, 곽후근, 정윤재, 권희웅, 정규식, “시그니처 해싱 기반 고성능 침입방지 알고리즘 설계 및 구현”, 정보처리학회논문지, June, 2007.
[10] Christoph M. Hoffmann and Michael J. O′Donnel, “Pattern Matching in Trees”, Journalo f the Assoclallonf or ComputingM achinery,Vol.29, No.I, January, 1982.
[11] S. Dharmapurikar, P.Krishnamurthy, T.Sproull, and J.W.Lockwood, “Deep Packet Inspection Using Parallel Bloom Filters”, The International Symposium on High Performance Interconnects (HotI), Aug., 2003.
[12] J. Lockwood, “Fast and Scalable Pattern Matching for Content Filtering”, Architectures for Networking and Communication System(ANCS), Oct., 2005.
[13] Sarang Dharmapurikar and John W. Lockwood, “Fast and Scalable Pattern Matching for Network Intrusion Detection Systems”, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL.24, NO.10, OCTOBER, 2006.
[14] ModsSecurity, http://www.modsecurity.org
[15] Ristic and Ivan, “Apache Security”, Oreilly & Associates.
[16] 고중식, 곽후근, 김정길, 정규식, “규칙과 페이로드에 따른 Snort의 성능 분석”, 한국정보보호학회 춘계학술대회, pp. 325-328, 2008.

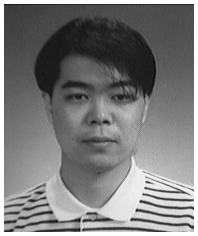
곽 후 근
E-mail : gobarian@q.ssu.ac.kr
1996년 호서대학교 전자공학과(학사)
1998년 숭실대학교 전자공학과(석사)
1998년~2006 숭실대학교 전자공학과(박사)
1998년 8월~2000년 7월 (주) 3R 부설 연구소 주임 연구원
2006년 3월~현 재 숭실대학교 정보통신전자공학부 대학원 postdoc
관심분야 : 네트워크 컴퓨팅 및 보안



왕 정 석
E-mail : wang@q.ssu.ac.kr
2004년 숭실대학교 정보통신전자공학부(학사)
2007년 숭실대학교 전자공학과(석사)
2007년~현 재 숭실대학교 정보통신전자공학부 박사과정
관심분야 : 네트워크 보안, 이중화 및 부하 분산



고 중 식
E-mail : kojs@q.ssu.ac.kr
2006년 숭실대학교 정보통신전자공학부(학사)
2006년 3월~현 재 숭실대학교 정보통신전자공학부 석사과정
관심분야 : 네트워크 보안



권 희 웅
E-mail : didorito@q.ssu.ac.kr
1997년 숭실대학교 정보통신전자공학부(학사)
1999년 숭실대학교 전자공학과(석사)
1999년~현 재 숭실대학교 정보통신전자공학부 박사과정
관심분야 : 네트워크 및 어플리케이션에 대한 부하 분산, 가속, 보안

정 규 식

E-mail : kchung@q.ssu.ac.kr

1979년 서울대학교 전자공학과(공학사)

1981년 한국과학기술원 전산학과(이학석사)

1986년 미국 University of Southern California(컴퓨터공학석사)

1990년 미국 University of Southern California(컴퓨터공학박사)

1998년 2월~1999년 2월 미국 IBM Almaden 연구소 방문 연구원

1990년 9월~현 재 숭실대학교 정보통신전자공학부 교수

관심분야 : 네트워크 컴퓨팅 및 보안