시그너처 해싱 기반 고성능 침입방지 알고리즘 설계 및 구현

왕 정 석[†] · 곽 후 근[††] · 정 윤 재[†††] · 권 희 웅[††††] · 정 규 식[†††††]

요 약

침입방지 시스템(IPS, Intrusion Prevention System)은 인라인모드(in-line mode)로 네트워크에 설치되어, 네트워크를 지나는 패킷 또는 세션을 검사하여 만일 그 패킷에서 공격이 감지되면 해당 패킷을 폐기하거나 세션을 종료시킴으로서 외부의 침입으로부터 네트워크를 보호하는 시스템을 의미한다. IPS에서 주로 사용되는 시그너처 기반 필터링에서는 침입방지시스템을 통과하는 패킷의 페이로드와 시그너처라고 불리는 공격패턴들과 비교하여 같으면 그 패킷을 폐기한다. 시그너처의 개수가 증가함에 따라 하나의 들어온 패킷에 대하여 요구되는 패턴 매칭 시간은 증가하게 되어 패킷지연 없이 동작하는 고성능 침입탐지시스템을 개발하는 것이 어렵게 되었다. 본 논문에서는 패턴 매칭 시간을 시그너처의 개수와 무관하게 하기 위하여 시그너처 해싱 기반에 기반한 고성능 침입방지시스템을 제안한다. 제안한 방식을 리눅스 커널 모듈 형태로 PC에서 구현하였고 웜 발생기, 패킷발생기, 스마트비트라는 네트워크 성능 측정기를 이용하여 시험하였다. 실험결과에 의하면 기존 방식에서는 시그너처 개수가 증가함에 따라 성능이 저하되었지만 본 논문에서 제안한 방식은 성능이 저하되지 않았다.

키워드 : 침입방지 시스템, 시그너처 기반 필터링, 시그너처 해싱

The Design and Implementation of High Performance Intrusion Prevention Algorithm based on Signature Hashing

Jeongseok Wang[†] · Hukeun Kwak[††] · Yunjae Jung[†††] · Huing Kwon[††††] · Kyusik Chung[†††††]

ABSTRACT

IPS(Intrusion Prevention Systems), which is installed in inline mode in a network, protects network from outside attacks by inspecting the incoming/outgoing packets and sessions, and dropping the packet or closing the sessions if an attack is detected in the packet. In the signature based filtering, the payload of a packet passing through IPS is matched with some attack patterns called signatures and dropped if matched. As the number of signatures increases, the time required for the pattern matching for a packet increases accordingly so that it becomes difficult to develop a high performance IPS working without packet delay. In this paper, we propose a high performance IPS based on signature hashing to make the pattern matching time independent of the number of signatures. We implemented the proposed scheme in a Linux kernel module in a PC and tested it using worm generator, packet generator and network performance measure instrument called smart bit. Experimental results show that the performance of existing method is degraded as the number of signatures increases whereas the performance of the proposed scheme is not degraded.

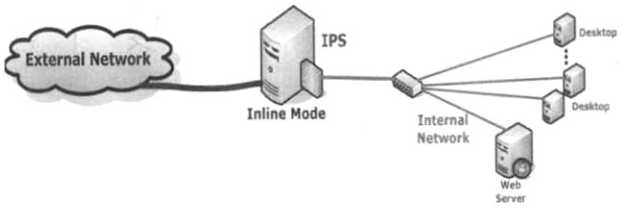
Key Words : IPS, Signature based Filtering, Signature Hashing

1. 서 론

인터넷을 비롯한 네트워크의 의존도가 높아지고, 점점 더 중요하고 복잡한 작업들이 컴퓨터와 이들의 연결 집합체인 네트워크를 통해 이루어 지고 있다. 이로 인해 네트워크 트래픽은 급증하여 이미 수 기가비트 급으로 확장되었고, 이러한 네트워크 속도의 증가로 인해 네트워크를 통한 서비스 거부 공격(DoS, DDoS), 인터넷 웜(internet worm), 이메일 바이러스, 피싱(phishing)등의 악의적인 공격도 빠른 속도로 증가하고 있으며, 이로 인한 피해도 급격히 증가되는 추세이다. 네트워크 및 내부 시스템에 대한 악의적인 공격에 대한 방어로 기존에는 방화벽(firewall), 침입탐지 시스템(IDS: Intrusion Detection Systems)등이 사용되었으나, 방화벽은 네트워크 트래픽에 대해 IP 주소, 프로토콜 종류, 서비스 포트 등의 정보만을 이용하여 이상 여부를 판단하기 때문에 공개된 포트를 사용하는 공격에는 취약점을 가지고 있어 최근에 많은 문제를 일으키는 웜, 이메일 바이러스 등의 공격 등을 차단하지 못하는 취약점을 가지고 있다.

또한 침입탐지 시스템은 패킷의 헤더 뿐만 아니라 데이터

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어 졌음.
† 정 회 원 : 숭실대학교 전자공학과 박사과정
†† 준 회 원 : 숭실대학교 전자공학과 대학원(postdoc)(교신저자)
††† 정 회 원 : 숭실대학교 전자공학과 대학원 졸업
†††† 준 회 원 : 숭실대학교 전자공학과 박사과정
††††† 정 회 원 : 숭실대학교 정보통신전자공학부 교수
논문접수 : 2007년 2월 28일, 심사완료 : 2007년 5월 9일



(그림 1) 침입 방지 시스템

(payload) 부분까지 조사하여 공격 여부를 판단하고, 이에 대한 판단 결과를 보고함으로써 네트워크 보안에 도움을 줄 수 있지만, 이는 이미 공격이 진행되어 내부 네트워크가 그 공격으로 인한 피해를 입은 후에 감지 할 수 있어 효과적이고 빠른 대응을 기대하기 어렵다. 침입방지 시스템(IPS)은 기존에 네트워크 방어를 위해 사용된 방화벽, 네트워크 침입탐지 시스템이 처리하지 못한 부분까지 빠르고 효율적으로 처리함으로써, 외부의 공격을 근본적으로 차단하여 네트워크 및 내부 컴퓨터를 안전하게 보호하고, 효율적이고 깨끗한 네트워크 유지할 수 있도록 한다[1, 2]. (그림 1)은 침입방지 시스템이 네트워크에서 어떻게 구성되는지를 나타내고 있다.

그림에서 나타나듯이 침입방지 시스템은 외부 네트워크와 내부 네트워크의 종단에 인라인으로 위치하며, 오가는 모든 패킷을 시그너처(signature)라고 불리는 공격 패턴을 표현한 룰들과 비교 검사하여 해당 패킷의 감염 및 이상 여부를 판단하고, 패킷을 폐기하거나 세션을 종료함으로써 네트워크를 공격으로부터 보호한다. 이를 위해서 침입방지 시스템은 높은 안정성과 빠른 성능을 확보해야만 하고, 정상적인 트래픽에 대해 최대한 지연 없는 서비스를 제공하고, 오탐지를 최소화하여 네트워크의 통신에 지장을 주어서는 안 된다. 이를 위해서 그동안 많은 노력을 통해 침입탐지시스템을 위한 효율적인 패턴 매칭 알고리즘이 고안되었고 현재까지 많은 발전이 이루어졌다. 하지만 보다 효율적인 침입방지 시스템 개발을 위해서 다음과 같은 사항이 고려되어야 한다.

- 성능의 안정성 : 단위시간당 들어오는 패킷이 증가함에 따라, 또는 적용되는 공격 패턴 룰의 개수가 증가에 따라 시스템의 성능이 크게 저하되지는 않는가?
- 룰 상관관계와 성능 : 룰의 상관관계에 따라 시스템의 성능이 영향을 받지는 않는가?
- 예측 가능한 성능 : 룰의 내용, 개수, 적용시기 등에 따라 항상 안정적이고, 예측 가능한 성능을 보이는가?
- 정확한 탐지 능력 : 적용되는 룰을 이용하여 항상 정확한 탐지를 해 낼 수 있는가?

본 논문에서는 침입탐지시스템을 위한 기존의 패턴 매칭 알고리즘에 대해 소개하고, 해당 알고리즘이 가지고 있는 한계와 문제에 대해 분석하고, 이를 해결할 새로운 패턴 매칭 알고리즘을 제안한다. 제안된 방식에서는 시그너처라고 불리는 공격 패턴 룰의 개수에 무관하게 동작하는 패턴 매칭 성능, 예측 가능한 성능, 기존 알고리즘보다 빠른 처리능력 등을 보임으로써 기존 알고리즘의 한계를 극복하였다. 본 논문의 구성은 다음과 같다. 2장에서는 기존의 패턴 매칭 알고리즘과 그 문제점에 대해 소개하고, 3창에서는 이를 해결하는 새로운 패턴 매칭 알고리즘을 제안한다. 4장에서는 기존 방식과 제안된 방식에 대해 실험을 통해 비교 및 토론 하며, 5장에서는 결론을 맺는다.

2. 기존 연구

침입방지 시스템(IPS)은 침입탐지 시스템(IDS)과 같이 심도 있는 패킷 검사를 수행하여 각 패킷 및 세션을 검사하며, 인라인모드(in-line mode)로 네트워크에 설치되어 네트워크를 지나는 모든 패킷을 살펴보며, 패킷 단위의 혹은 그이상의 연관성을 검토, 분석하여 침입의 여부를 판단하고, 해당 패킷을 드롭하거나 커넥션을 종료시킴으로서 외부의 침입으로부터 네트워크를 보호하는 시스템을 의미한다. 침입방지 시스템은 네트워크를 보호하기 위해 크게 두 가지로 구분되는 공격에 대한 방어를 수행하는데, 이미 알려진 공격의 유형과 내용에 대해 방어를 수행하는 시그너처 기반 필터링(signature based filtering)과 스스로 학습과 탐지로 패킷 및 세션의 이상 유무를 판단하여 알려지지 않은 공격이나 이상 세션 등에 대해 방어하는 자동 학습(self-learning)에 의한 변칙(anomaly) 탐지 및 방지의 기법이다.

이 중 알려진 공격 및 외부의 침입으로부터 네트워크를 보호하기 위해 시그너처 기반 필터링을 수행하는 경우 효과적인 패킷 검사 방법이 필요한데 이 과정을 Deep packet inspection이라 하며, 이를 통해 인터넷 웜 바이러스, 컴퓨터 바이러스, 스팸 메일, 불법적 저작물들의 흐름을 파악하고 차단할 수 있다. 또한 이 과정을 통해 각 패킷에 있는 모든 정보를 확인함으로써 컨텐츠 정보의 정확성 및 컨텐츠 스위칭 등에 활용할 수 있다. 이와 같은 패킷 데이터를 조사하는 패턴 매칭(pattern matching) 과정은 침입을 탐지하는 과정 중에서도 가장 많은 계산을 요구한다. 또한, 공격의 범위와 알려진 침입 및 바이러스의 시그너처는 시간이 갈수록 계속 증가하게 되고, 네트워크의 대역폭 및 요구 속도가 기하급수적으로 빨라짐에 따라 시그너처 기반의 필터링을 수행함에 있어 현재 네트워크를 지나려는 패킷이 알고 있는 많은 시그너처 중 일치하는 지를 확인하는 알고리즘의 성능이 침입방지 시스템의 네트워크 성능과 처리속도, 보장 대역폭 등에 매우 중요한 척도가 되고 있다[3-6].

본 논문에서는 시그너처 해싱 기법을 이용하여 패킷을 효율적으로 검사하여 알려진 공격에 대한 감염 여부 및 이상 패킷을 신속하고 정확하게 검출할 수 있는 고성능 침입방지 시스템을 구현하는 방법을 제안한다. 이를 위해 본 절에서는 그동안 제안되고 활용되었던 패킷 검사 방법인 순차 매칭[7], Aho-Corasick[8], Jump Aho-Corassick[9] 등의 알고리즘에 대해 소개하고, 각 알고리즘이 가지는 문제점을 지적하고 이를 해결하는 새로운 알고리즘을 제안한다.

2.1 시그너처와 룰

알려진 공격에 대해 차단하는 시그너처 기반 필터링의 경우 해당 공격에 대한 패턴을 이용하여 패킷 데이터를 조사한다. 이러한 패턴 매칭을 위해서는 공격이 나타나는 패턴에 대해 정확하게 기술된 내용이 존재하여야 하며, 이 전체적인 내용을 룰이라고 한다. 이 룰에는 직접적으로 패킷의 데이터와 비교될 실제 시그너처를 포함해, 시그너처가 적용될 서비스 포트, 룰의 이름, 시그너처가 나타나는 위치, 비교 방식 등에 대한 상세한 정보가 담겨져 있으며, 이 룰에 기술된 정보와 일치하는 패킷이 들어올 경우 침입방지 시스템은 해당 패킷을 공격 또는 침입 패킷으로 판단하고 그에 따른 적절한 조치를 수행할 수 있다. 일반적으로 룰은 다양한 형태를 통해 기술될 수 있으나 본 논문에서는 침입탐지 시스템에서부터 현재에 이르기까지 많은 사용 층을 확보하고 있는 SNORT[10]의 룰 기술 방식을 따르도록 한다.

(그림 2)는 시그너처와 룰의 상관관계 및 간단한 룰의 예제를 나타낸다. 이 그림에서 예시된 룰은 25번 포트를 통해 접근하는 패킷에 대해 ".zip\"0d0a0d0a\""라는 시그너처가 존재하는 패킷에 대해 "Win32/Mytob.worm.66932-zip-MIME"이라는 웜 패킷으로 판단하라는 것을 의미한다. 시그너처 기반 필터링 기법을 응용하면 악성코드를 통해 전파되는 웜 바이러스 뿐 만 아니라 네트워크상의 상당부분을 패턴화 하여 적용할 수 있다. 예를 들면 스팸 메일 필터링, 도메인 필터링, 애드/스파이 웨어 필터링, P2P 서비스 제어 등 다방면에 적용이 가능하다.

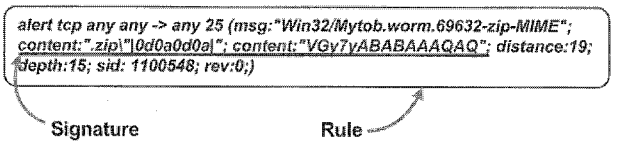
2.2 순차 매칭(Sequential matching)

순차 매칭은 들어오는 패킷에 대해 각각의 룰 내용을 반복적으로 조사하며 해당 시그너처가 패킷 데이터에 존재하는지를 확인하는 가장 기본적인 매칭 방법이다. 순차 매칭은 각 룰들에 대해 가장 원시적이고 확실한 매칭을 수행할 수 있는 반면, 적용되는 룰의 개수가 많아질수록 기하급수적으로 처리 시간 지연 및 성능 저하가 발생하게 되어 확장성과 성능에 치명적인 결함이 있다. 이러한 문제로 인해 네트워크 트래픽이 적고, 비교를 수행하여야 하는 룰의 개수가 한정적일 경우에 사용할 수 있다. 하지만 연일 새로운 공격과 취약점이 발표되고, 네트워크 트래픽이 계속 증가하고 있는 현재의 네트워크를 비추어 볼 때 순차 매칭의 적용은 현실적인 어려움을 안고 있다.

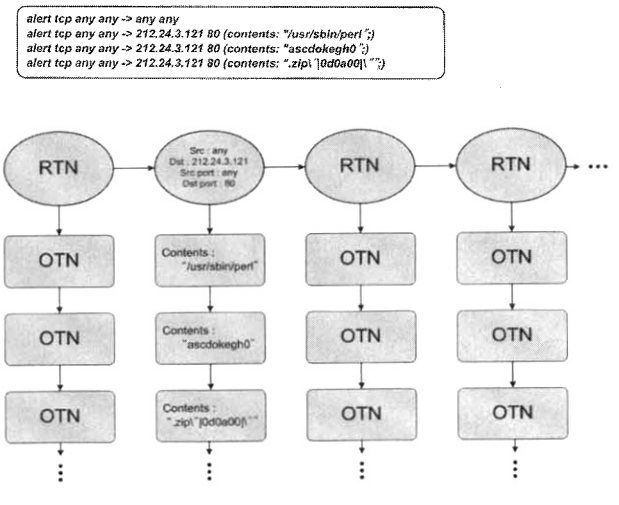
2.3 Aho-Corasick 알고리즘

Aho-Corasick 알고리즘[8]은 과거 침입방지 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. 이를 위해 SNORT는 룰에 대해 (그림 3)에서 보이는 바와 같은 2차원 링크드 리스트를 구성한다. 이 링크드 리스트는 Rule Tree Node(RTN)와 OTN (Option Tree Node)로 구성되며, 룰의 시그너처와 매칭이 확인되었을 때의 행동에 대한 부분이 각각 RTN과 OTN에 맵핑된다[11]. RTN은 여러 룰이 공통적으로 가질 수 있는 조건들, 예를 들면 패킷의 발신지와 목적지 주소 및 포트, 패킷의 전송 방향, 프로토콜 타입 등을 담고 있고, OTN은 각 룰에 더해질 수 있는 여러 옵션들, 예를 들면 TCP flag, ICMP 코드 타입, 시그너처 등의 내용을 담고 있다[11]. Aho-Corasick 알고리즘을 이용하는 SNORT는 먼저 패킷의 프로토콜과 발신지, 목적지, 포트번호 등을 기준으로 적용할 룰 집단(ruleset)을 선택하고, 그 집단 내에 있는 모든 시그너처에 대해 Aho-Corasick 알고리즘을 사용하여 검사를 시도하고, 매칭 되는 패턴이 발견되면 해당 룰이 가리키는 행동을 취하게 된다.

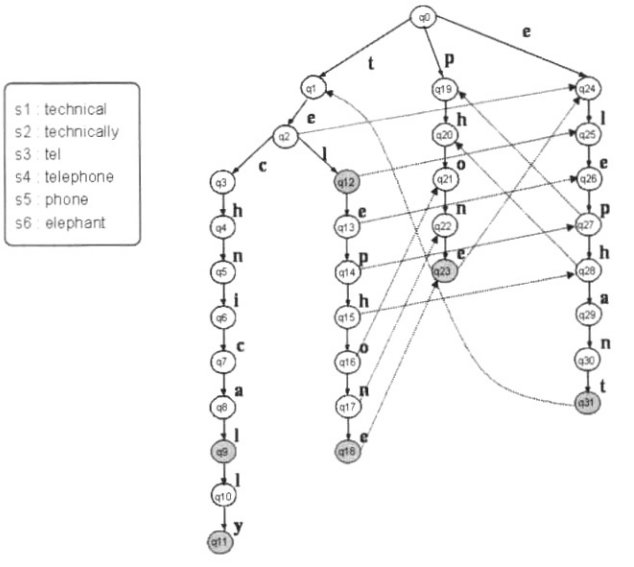
(그림 4)는 예시된 시그너처들을 검사하고자 할 경우 Aho-Corasick 알고리즘이 이루는 트리 구조 및 매칭 구조를 나타낸다. (그림 4)에서도 알 수 있듯이 Aho-Corasick 알고리즘은 트리구조를 통해 매칭 속도와 적용 가능한 룰의 개수를 비약적으로 증진시키는 결과를 보였지만 이는 기본적으로 각 시그너처가 상당한 연관관계를 가져야만 제 효과를 낼 수 있다. 즉 시그너처 간의 상관관계에 굉장히 종속적인 성능을 나타내는 알고리즘으로 불특정 시그너처를 사용하게 되는 실제 네트워크에서 이는 시스템의 성능에 대한 예측을 어렵게 하고, 확장성을 가로막는 요인으로 작용한다. 또한 룰 내의 시그너처간 상관관계가 우수하여 Aho-Corasick 알고리즘이 제 성능을 발휘하는데 적합하다 하더라도, 매칭 알고리즘 자체의 특성상 한 바이트의 캐릭터(character) 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 없게 된다[12].



(그림 2) 시그너처와 룰



(그림 3) RTN OTN 링크드 리스트 구조와 SNORT 룰



(그림 4) Aho-Corasick 알고리즘

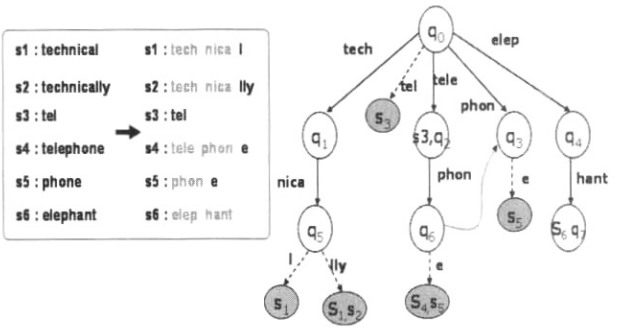
2.4 Jump Aho-Corasick 알고리즘

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 Junp Aho-Corasick 알고리즘이다. (그림 5)과 <표 1>에서 보는 바와 같이 Jump Aho-Corasick 알고리즘은 같은 시그너처 그룹에 대해 기존의 Aho-Corasick 알고리즘이 구성하던 트리와 사뭇 다른 형태의 트리를 구성한다. 이를 위해 먼저 연관성 있는 부분들의 묶음으로 각 시그너처를 조사 및 분해하여 이 분해된 묶음 단위의 블록을 이용하여 트리를 구성한다.

Jump Aho-Corasick 알고리즘은 비록 Aho-Corasick 알고리즘이 가지는 메모리 접근의 증가 및 철자 단위의 매칭으로 인한 병목 현상 등의 문제를 어느 정도 완화시키지만, 실제 존재하는 시그너처의 길이가 매우 다양하고, 그에 따라 많은 메모리를 필요로 하는 단점을 가지고 있다. (그림 5)에서 볼 수 있듯이 이는 시그너처의 범위 및 상태 수를 획기적으로 축소하고, 실제 매칭 진행시 건너뛰며 진행될 확률을 높임으로써 성능 개선을 거두었지만, 오히려 이전에 비해 더욱 시그너처 간의 상관관계에 의존적이 되었다. 이는 앞서 지적하였던 Aho-Corasick 알고리즘이 가지는 한계를 그대로 유지하는 것으로 역시 시그너처 간의 관계에 많은 영향을 받으며 성능 수치에 대한 예측이 어려운 알고리즘이다[13].

2.5 제안된 방법

본 논문에서 제안하는 시그너처 해싱 기반의 알고리즘은 기존의 알고리즘이 가지고 있는 확장성과 룰 적용 안정성을 보장하기 위해 시그너처 해싱 기법을 이용하여 불필요한 매칭 동작을 줄이고 효율적으로 패킷의 데이터를 검사하는 방



(그림 5) Jump Aho-Corasick 알고리즘

<표 1> Jump Aho-Corasick 알고리즘의 시그너처 분류

[state, substr]	Next State	Matching String	Failure Chain
[q_0, tech]	q_1	-	q_0
[q_0, tele]	q_2	S_3	q_0
[q_0, phon]	q_3	-	q_0
[q_0, elep]	q_4	-	q_0
[q_1, nica]	q_5	-	q_0
[q_2, phon]	q_6	-	q_3, q_0
[q_4, hant]	q_7	S_6	q_0
[q_0, tel]	-	S_3	-
[q_5, lly]	-	S_1, S_2	-
[q_3, e]	-	S_5	-
[q_5, l]	-	S_1	-
[q_6, e]	-	S_4, S_5	-

법을 제안한다. 이러한 방법은 패킷의 데이터 부분에 대해 검사할 때 실제로 매칭 되는 시그너처의 수 자체를 줄임으로써 항상 매칭 되는 룰이 극도로 적은 수를 유지하도록 하여 룰의 개수에 영향을 받지 않아 확장성이 용이하고, 그 성능 수치를 예측 가능토록 하여 안정적인 침입방지 시스템을 운용할 수 있도록 한다.

3. 시그너처 해싱에 기반한 고성능 침입방지 시스템

패턴 매칭의 수행 성능을 향상시키기 위해서 필요한 가장 중요한 부분은 처리해야 하는 패킷이 도착했을 때, 해당 패킷의 데이터를 룰의 시그너처와 비교하는 횟수를 줄이는데 있다. 즉, 비교해야 하는 룰의 개수를 줄이거나, 비교해야하는 경우의 수를 줄임으로써 매칭 수행 성능을 향상시킬 수 있다. 하지만 단순히 분류를 잘 해서 매칭을 수행해야 하는 룰의 수를 줄이는 것은 룰의 개수가 점점 많아지고 있는 현실을 감안하면 그다지 효과적이라고 할 수 없는 방법이다.

이를 보장하기 위해서는 항상 예측 가능하고, 성능이 보장되는 새로운 알고리즘이 필요하고, 또한 앞서 소개한 다른 알고리즘이 가진 한계인 시그너처 간의 상관관계에서도 자유로워야 한다. 본 절에서는 시그너처 해싱이라는 기법을 기존의 알고리즘이 가지고 있는 한계를 극복하는 새로운 알

고리즘을 제안한다. 이를 이용하여 룰의 개수가 증가하더라도 그에 무관하게 시스템이 동작하도록 룰의 확장성(scalability)과 성능(performance)을 보장하는 보다 효율적이고 정확한 침입방지 시스템을 구성할 수 있다.

3.1 시그너처 해싱

시그너처 해싱이란 적용되는 모든 룰의 내부에 존재하는 시그너처에 대해 이 중 일부(2 bytes)를 정해 해싱 값을 만들고, 이 해싱 값을 이용하여 검색하고자 하는 패킷 데이터와의 비교를 통해 실제 확인해볼 필요가 있는 시그너처 만을 비교 하도록 경우의 수를 현저히 줄이는 방식이다. 시그너처의 특정 2byte를 정하여 이를 해싱의 기준 값으로 삼고, 이 해싱 값에 각 룰의 실제 시그너처를 연결하여 매칭에 사용될 불필요한 동작을 획기적으로 줄임으로써 매칭 속도를 향상시키고, 항상 안정된 성능을 낼 수 있도록 하기위한 알고리즘이다. 기준치를 2bytes로 삼고 이를 이용하여 해싱 값을 구하는 이유는 일단 해싱을 통해 얻을 수 있는 경우의 수가 충분히 많고, 데이터를 실제 스캔할 때 소요되는 비용이 최소화되기 때문이다.

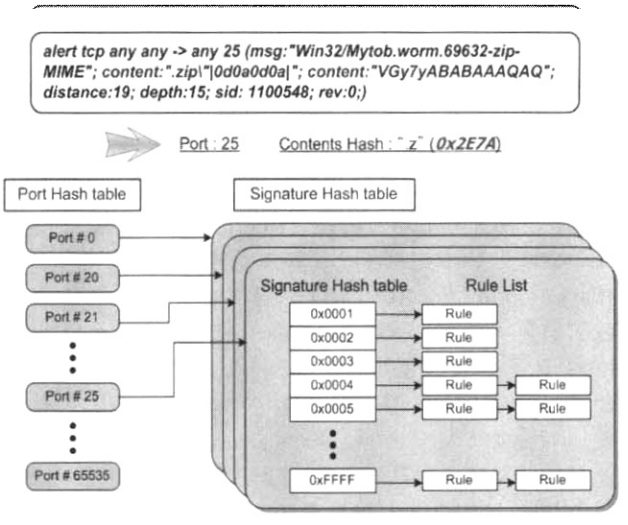
2bytes를 이용하여 해싱 값을 구할 때 가능한 경우의 수는 이론적으로 모두 65535개 이다. 이는 현재 존재하는 룰의 개수보다 훨씬 많은 양이고, 프로토콜과 포트로 나누어진 상태의 룰에 있어서는 충분히 크기 때문이다. 제안된 방식은 패턴 매칭을 수행하기 전에 적용할 룰을 정하여 해당 룰을 해싱 기법을 이용하여 포트 및 시그너처 해시 테이블로 정리하는 준비 단계와 실제 패킷이 들어왔을 때, 이 패킷 데이터를 조사하여 시그너처가 정확히 존재하는지를 검사하는 부분으로 나누어진다.

3.1.1 포트/시그너처 해시 테이블

패턴 매칭 수행에 앞서 적용할 룰에 대해 포트 및 시그너처를 기준으로 정리하는 단계를 거치게 되는데, 이를 위해 (그림 2)에서 예시하였던 룰에서 해당되는 서비스 포트와 시그너처 중 일부(2 bytes)를 추출하여 포트 해시 테이블과 시그너처 해시 테이블을 구성하는 과정이 필요하다. 이를 위해 먼저 각 프로토콜과 서비스 포트 번호 등에 대해 먼저 분류를 하고, 해당 포트를 해싱 한 값을 이용하여 포트 해시 테이블을 구성한다. 이는 적용되는 룰에 대해 같은 포트 단위로 해시 테이블을 작성하여, 패턴 매칭을 진행할 때 경우의 수를 일차적으로 줄여주는 역할을 한다. 포트 해시 테이블을 구성하면, 해당하는 포트 해시 테이블이 갖고 있는 시그너처 해시 테이블에 시그너처의 해싱 값을 등록한다. 여기서 작성한 시그너처 해시 테이블에 실제 매칭에 사용되는 룰의 정보를 연결 리스트를 이용하여 등록하는 과정을 마치면 패턴 매칭을 수행하기 위한 사전 준비 작업을 마치게 된다.

<u>일반적인 해싱을 적용한 시그너처 해싱</u>

(그림 6)은 룰에서 포트 해시 테이블과 시그너처 해시 테이블을 구성하는 예와 각 해시 테이블에 어떤 식으로 룰들



(그림 6) 포트 및 시그너처 해시 테이블

이 맵핑되는지를 보여준다. (그림 6)에서와 같이 예시된 룰의 경우 25번 포트(SMTP)를 이용하여 전파되는 이메일 웜 바이러스에 대한 룰로써 일단 포트 정보가 25번이며, 시그너처 해싱 값으로 사용할 부분을 시작 하는 스트링에서 첫 2 bytes를 이용하는 것으로 여기서는 ".z"가 선택되었다는 것을 볼 수 있다. 이 경우 해당 포트별로 룰이 존재할 경우 포트 해시 테이블에 포트의 해싱 값을 등록하고, 해당 포트 해시 테이블에 시그너처 해시 테이블을 연결한다.

시그너처 해시 테이블에 해싱 값을 등록한 후, 그 해싱 값에 룰의 상세 비교 정보를 연결 리스트(linked list)를 이용하여 연결해 놓는다. 연결 리스트를 이용하여 등록되는 룰의 상세 비교 정보는 시그너처 전체 내용, 패킷에서 해당하는 시그너처가 존재하는 위치 등 직접 비교에 사용되는 내용들을 포함하고 있다. 이러한 일련의 방법을 통해 적용하고자 하는 모든 룰에 대해 해싱 값을 구하고 연결 리스트를 등록하는 과정을 거치고 나면 포트 해시 테이블과 시그너처 해시 테이블이 완성되고, 이는 실제 패킷이 들어올 경우 패킷의 데이터를 비교, 검사하는 기준이 된다. 해싱 값을 사용하여 검사를 하는 경우 해당 해싱 값이 일치하는지 매칭 하는데 걸리는 소요시간이 일반적인 텍스트 매칭에 비해 극히 짧고, 각 해싱 값에 연결되는 룰의 개수가 줄어듦에 따라 패킷 데이터에 대해 룰과 매칭 되는 경우의 수는 극히 줄어들게 된다.

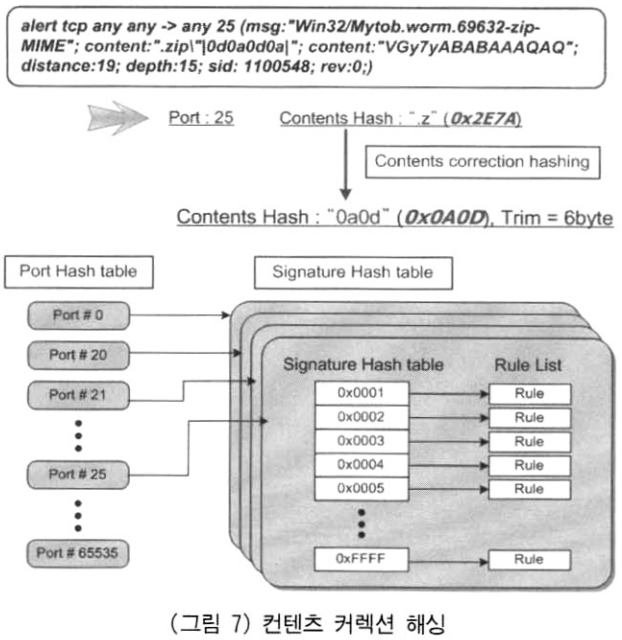
<u>컨텐츠 커렉션 해싱(Contents-correction hashing)을 적용한 시그너처 해싱</u>

제안된 시그너처 해싱을 이용한 방법은 패킷에 대해 매칭을 수행함에 있어 실제로 매칭 되는 룰의 수를 획기적으로 줄임으로써 매칭 수행 성능을 비약적으로 향상시키고, 적용되는 룰의 개수와 관계없는 처리를 보장한다. 이는 해싱 기법을 이용하여 매칭 시 소요되는 시스템 자원을 줄이고, 실제 매칭에 수행되는 경우의 수를 줄임으로써 얻을 수 있는 효과이다. 하지만 최근 급증하고 있는 웜의 경우 그에 대한 변종이 계속해서 나타나게 되고 이들 변종 웜은 원래의 웜과

비교하여 시그너처의 변화가 크지 않게 된다. 이런 경우 유사한 시그너처의 증가로 인해, 시그너처 간의 연관성이 높아지게 되고, 이 때 일반적인 해싱을 이용하기 위해 시그너처의 내용 중 시작 부분의 2bytes를 이용하여 해싱 값을 구하게 되면 같은 해싱 값이 존재하게 될 가능성 또한 높아진다.

해싱을 이용하여 검색을 할 경우 구해진 해싱 값에 실제 비교해야 할 룰이 연결 리스트로 연결되게 되는데, 2bytes를 이용하여 구한 해싱 값이 같다는 것은 결국 해당 해싱 값에 많은 룰들이 연결 리스트를 이용하여 연결되어 있음을 의미한다. 결국 이는 직접적으로 시그너처 전체를 비교해야 하는 항목이 많아지는 것을 뜻한다. 이는 전반적인 탐지 성능을 저하시킬 뿐만 아니라 알고리즘의 성능 예측을 어렵게 만들고, 비교되는 룰의 개수와 경우의 수를 줄여 성능을 향상시킨다는 알고리즘의 기본 취지에 맞지 않는다.

이러한 경우를 줄이기 위해 해싱 값을 구하기 위한 2bytes 부분을 시그너처의 특정 부분(예, 첫 2bytes)이 아닌 임의의 부분을 취하여 해싱 값을 구하도록 보완하여 제안하는 기법이 컨텐츠 커렉션 해싱(Contents - correction hashing)이다. 제안하는 방법에서는 랜덤하게 생성된 위치 값을 이용하여 컨텐츠 시작에서부터 그 위치 값 만큼 떨어진 곳에서 찾은 2bytes에 대하여 계산한 해싱 값과 그 위치 값을 사용한다. 이 방법은 시그너처간 유사성이 높은 경우에도 각각 해싱 값이 다르게 나오도록 조정하는 역할을 수행하며, 이로 인해 시그너처 해시 테이블이 더 균일한 분포를 가질 수 있도록 한다. 따라서 시그너처 해싱 값을 결정하는 데 있어, 컨텐츠 커렉션 해싱 기법을 이용하면 상관관계가 높은 룰을 적용하거나, 룰의 개수가 많아짐에 따라 같은 해싱 값에 룰이 집중되는 것을 막을 수 있으며, 보다 효율적인 분포를 통해 매칭 스피드를 높일수 있는 효과를 얻을 수 있다. (그림 7)은 컨텐츠 커렉션 해싱 값을 구하고, 사용하는 방법



(그림 7) 컨텐츠 커렉션 해싱

에 대해 보여준다. 그림에서 보이듯이 일반적인 해싱을 적용한 방식의 경우 해당되는 시그너처의 첫 2bytes 만을 이용하여 해싱 값을 구하게 되면, ".z"라는 부분을 이용하여 해싱 값을 구하게 되고 이는 중복될 가능성이 대단히 높게 된다. 따라서 이를 극복하기 위해 시그너처 내의 특정 위치가 아닌 임의의 위치-위 예제에서는 6bytes 이후-의 2bytes 부분을 택하여 해싱 값을 구하게 됨에 따라, 다른 시그너처와 중복될 가능성을 낮출 수 있게 되며, 결과적으로 시그너처 해시 테이블에 연결되는 룰의 연결 리스트도 일반적인 해싱 방법을 사용하였을 때에 비해 훨씬 적어지게 된다. 이는 매칭을 수행하는데 있어 실제 비교해야할 룰의 수가 줄어듦을 의미하고, 따라서 매칭 수행 성능 및 시간을 예측 가능하게 해주는 역할을 한다.

컨텐츠 커렉션 해싱을 사용할 경우 일반적인 해싱을 적용한 경우와 달리 byte windowing을 통해 해싱 값 비교를 진행하는 과정에 변화가 필요하다. 일반적인 해싱을 적용한 경우에는 실제 시그너처와 전체 텍스트 비교 (full-text matching)시 조사하던 패킷의 데이터 부분과 비교할 시그너처에 대해 매칭을 수행하면 되지만, 컨텐츠 커렉션 해싱이 적용된 경우 해싱의 기준 값이 첫 2bytes가 아니기 때문에 이를 비교하기 위해서는 시그너처의 어느 부분을 이용하여 해싱 값을 구했는지에 대한 정보가 필요하다. 이를 위해 컨텐츠 커렉션 해싱을 적용하는 경우에는 시그너처 해시 테이블에 연결되는 룰의 정보에 시그너처의 내용 중 시작부분에서 얼마만큼 떨어진 곳의 2bytes 값을 이용하였는지에 대한 정보(trim)를 기록한다. 본 논문에서는 매칭 효율을 높이고, 적용되는 룰의 개수에 대한 확장성을 보장하기 위해 컨텐츠 커렉션 해싱을 적용한 방법을 사용한다.

3.2 패턴 매칭 방법

적용되는 모든 룰에 대해 포트 해시 테이블과 시그너처 해시 테이블이 준비되면 들어오는 패킷에 대해 검사를 시작할 준비가 완료된 상태이다. 패킷이 들어오면 일단 해당 패킷을 프로토콜 및 포트에 따라 분리하게 된다. TCP 패킷인지 UDP 패킷인지 다른 프로토콜의 패킷인지, 그리고 TCP나 UDP 패킷의 경우 접속하고자 하는 포트가 무엇인지에 대해 분류를 한 후 해당되는 포트 해시 테이블만 비교함으로써 먼저 매칭이 진행될 룰의 수를 줄인다.

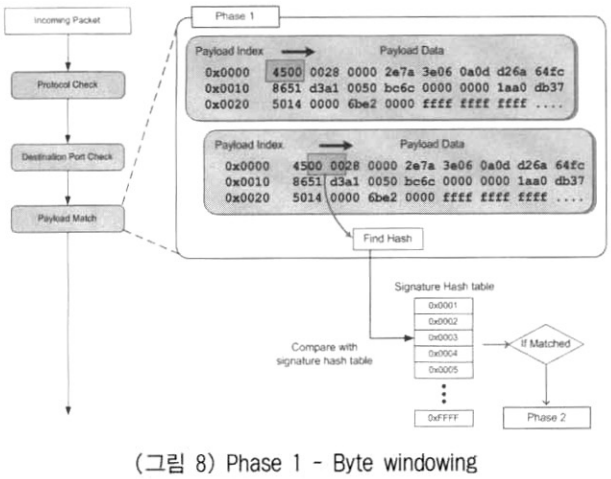
제안한 방법은 실제 패킷 데이터에 대해 매칭을 수행함에 있어 두 가지 절차를 거치게 된다. 먼저 패킷의 데이터 부분을 기준치(2bytes) 만큼씩 옮겨가며, 해싱 값을 구해 그 해싱 값과 일치하는 값이 시그너처 해시 테이블에 존재하는지를 확인하는 과정이다. 이 부분은 해싱 기법을 사용하여 매칭을 수행하므로 원하는 해싱 값이 존재하는지를 한 번에 확인할 수 있다. 해싱 값을 얻어 비교하는 데이터 부분을 패킷의 끝부분으로 옮겨가며 계속 검사를 하다가 시그너처 해시 테이블에 존재하는 해싱 값이 발견되면, 이는 해당 패킷이 공격 패킷일 가능성이 있다는 것을 의미하고, 세부 조사를 통해 실제 공격 패킷인지 여부를 정밀하게 검사하여

판단하게 된다. 첫 번째 비교 과정을 통해 실제로 정밀 조사를 해야 하는 해당 해싱 값에 분포되는 룰의 수가 대폭 감소하기 때문에 - 이상적일 경우에 1개 - 이를 통해 비약적인 속도 향상을 얻을 수 있다.

3.2.1 Phase 1 - Byte windowing

먼저 byte windowing을 통해 패킷의 데이터 부분을 windowing하며 검사한다. 이 경우 시그너처 해싱 값에 사용되었던 기준치(2bytes) 만큼씩 windowing하며 해싱 값을 구한다. 여기서 구해진 해싱 값을 가지고 이미 작성되어 있는 시그너처 해시 테이블에 같은 값이 있는지를 비교하고, 같은 값이 없다면 해당 부분은 이상이 없다고 판단하고 계속 windowing을 진행한다. 이를 통해 해당하는 부분에 대해 실제로 모든 룰에 존재하는 시그너처와 시그너처를 수행하지 않아도 해당 부분과 같은 부분이 존재하지 않는 다는 것을 알게 되고 이로 인해 시그너처를 수행하는 횟수를 줄일 수 있다. 이미 프로토콜과 포트 번호를 통해 해당되는 룰의 가지 수가 줄어든 상태에서 해싱 값을 이용하여 다시 한 번 분류를 해놓은 상태이므로, 실제 비교해야하는 대상의 룰은 전체 룰에서 극히 일부이다. 만약 시그너처 해시 테이블과 비교하여 같은 값이 존재하면 해당 두 번째 단계를 적용하여 최종적으로 해당 패킷이 실제 룰과 일치하는지를 정밀 검사한다.

(그림 8)은 byte windowing을 하며 해싱 값을 찾는 과정을 보여준다. 그림에서와 같이 패킷의 데이터 부분을 byte windowing 하며 2bytes씩 뽑아내어 해싱 값을 구한 후, 이를 해당 시그너처 해시 테이블과 비교하는 작업을 수행하는 과정을 진행한다. 구한 해싱 값이 시그너처 해시 테이블에 존재하지 않는다면 의심되는 부분이 아니므로 다음 byte로 windowing 하여 다시 해싱 값을 구하고 해시 테이블과 비교하는 작업을 반복한다. 만일 byte windowing을 진행하며 구한 해싱 값이 시그너처 해시 테이블과 일치한 경우가 발생한다면, 이는 공격 패킷으로 의심해 볼 수 있는 상황이 되므로 실제 룰 매칭을 수행하여 세부 검사를 진행한다.



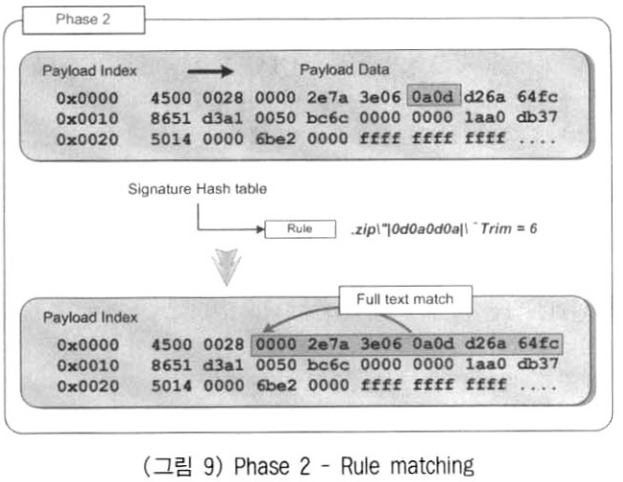
(그림 8) Phase 1 - Byte windowing

3.2.2 Phase 2 - Rule matching

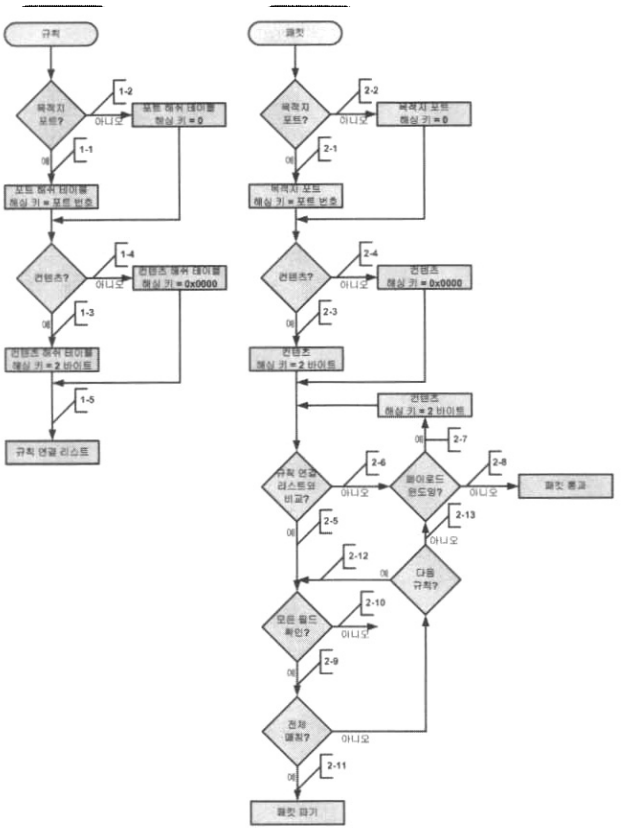
Byte windowing 하며 얻은 해싱 값과 시그너처 해시 테이블에 존재하는 해싱 값이 같을 경우 해당 패킷이 룰과 일치하는지를 의심하고 확인해 보아야 한다. 이 경우 시그너처 해시 테이블의 해당 해싱 값에 연결된 룰에 대한 연결 리스트를 따라가며 시그너처의 전체 값과 패킷의 해당 데이터를 비교하여 룰에 정확히 일치하는지를 확인한다. 이 작업은 오탐을 막기 위해 굉장히 중요한 행동이며, 룰에 기술된 다른 많은 정보와도 함께 비교하여 정확히 일치하는지를 확인하는 과정을 거쳐 최종적으로 패킷이 룰에 적용되어 적절한 조치를 취해도 되는지를 확인하는 단계이다. 이 경우에 매칭 되는 룰의 실질적인 수를 줄이기 위해 앞서 포트 및 시그너처 해시 테이블을 구성하고, 패킷의 데이터와 매칭을 수행할 때 byte windowing을 시행한 것이다.

룰의 내부에는 오탐을 막기 위해 출발지와 목적지 정보, 시그너처 이외에도 해당 시그너처가 패킷에서 존재하는 위치 등의 상세 정보가 들어있으며, 이를 통해 더욱 세밀한 검사가 가능하게 한다. 실제 룰 매칭을 수행할 때에는 이러한 부가 정보들의 내용을 먼저 비교하면, 패킷 데이터 부분과 시그너처 전체를 매칭하지 않아도 공격 패킷 여부를 쉽게 파악할 수 있어 더욱 실제 매칭 수행의 가능성을 낮출 수 있다. 본 논문에서 사용한 컨텐츠 커렉션 해싱을 적용하기 위해 룰 정보에는 시그너처 해시 테이블에 기록된 해싱 값이 실제 시그너처에서 얼마만큼의 위치에 존재하는 지를 기록하고 있으며(trim), 룰 매칭을 진행하는 과정에서는 이 거리만큼 앞으로 이동하여 전체 시그너처와 매칭을 진행하여야 정확히 패킷의 데이터와 시그너처가 일치하는 지를 확인할 수 있다. (그림 9)은 룰 매칭을 진행하는 세부 과정에 대해 보여준다.

(그림 10)은 제안된 알고리즘의 전체적인 순서도를 나타낸다. 왼쪽 그림은 기존 규칙을 이용하여 포트 및 시그너처 해시 테이블을 만드는 과정이고, 오른쪽 그림은 패킷이 IPS로 들어왔을 때 제안된 알고리즘에 의해 처리되는 과정을 나타낸다.



(그림 9) Phase 2 - Rule matching



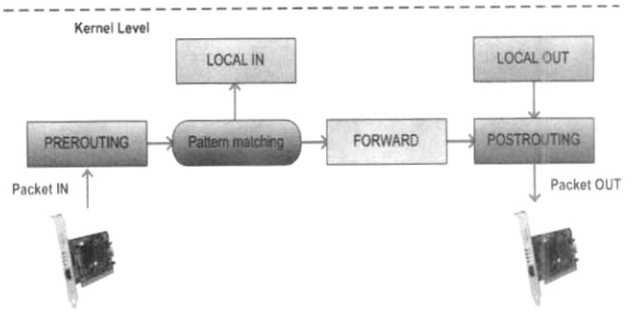
(그림 10) 제안된 알고리즘의 순서도

4. 구현 및 실험

4.1 구현 및 실험 환경

제안한 시그너처 해싱 기법을 적용한 알고리즘을 평가하기 위해 실제 침입방지 시스템에서와 같이 리눅스 커널 모듈 형태로 구현하여 실험에 사용하였다. 사용한 커널 버전은 2.6.16.18을 사용하였으며, 리눅스 넷필터(Netfilter)의 PREROUTING hook point를 이용하여 패킷을 처리하는 형태로 구현되었다. 넷필터는 리눅스에서 제공하는 커널 레벨에서 네트워크 패킷을 처리할 수 있도록 다양한 기능과 인터페이스를 제공하는 오픈소스 커널 모듈로써, 본 논문에서 제안하는 방법에 대한 구현은 패킷이 시스템에 도착하여 라우팅 로직을 수행하기 이전 단계인 PREROUTING 부분에서 패킷을 검사하여 공격 패킷 여부를 판단할 수 있도록 구현되었다. (그림 11)은 리눅스에서 제안된 알고리즘이 구현되어 적용된 부분에 대해 나타낸다.

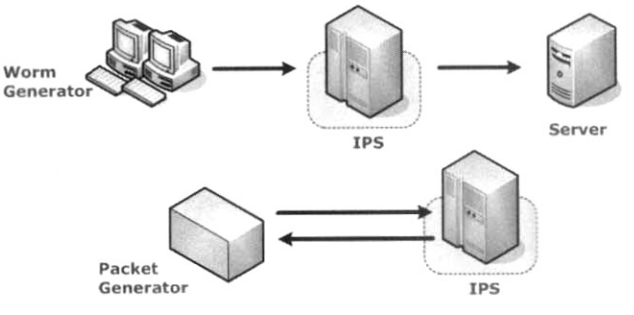
<표 2>는 실험에 사용된 하드웨어와 소프트웨어를 나타낸다. 성능과 효율성 측면 뿐만 아니라 네트워크 보안 장비로서 가장 중요한 척도는 역시 공격을 얼마만큼 정확하게 차단하느냐는 부분일 것이다. 따라서 적용되는 룰의 개수에 따른 알고리즘 별 네트워크 트래픽 성능 수치와 알고리즘 별 공격 패킷 검출 성능 수치를 구분하여 실험을 진행하였다[14]. 실험을 위해서 웜 패킷을 생산해낼 수 있는 웜 발생기와 서버, 그리고 일반적인 네트워크 구성 환경을 시뮬레이션 할 수 있도록 패킷 발생기를 이용하여 각각의 환경에 대해 테스트를 수행하였다. 사용자가 웜 발생기와 패킷 발생기를 이용하여 네트워크 성능에 대한 테스트를 수행하였을 때 기존에까지 연구되어 사용되고 있는 Jump Aho-Corasick 알고리즘을 이용하는 경우와 새로 제안된 시그너처 해싱 기법을 사용한 경우에 대해 룰의 개수에 따른 성능 변화를 측정함으로써 제안된 방법의 성능을 검증하는 방식으로 실험을 수행하였다[15].



(그림 11) 패턴 매칭 엔진 구현 위치

〈표 2〉 실험용 하드웨어 및 장비

	하드웨어		수 량
	CPU (GHz)	RAM	
웜 발생기	P-Ⅳ 2.6	512 MB	3
침입방지 시스템	Xeon 3.4	2 GB	1
서버	P-Ⅳ 2.6	2 GB	1
패킷 발생기	-	-	Smartbit[16] 1EA



(그림 12) 실험 환경

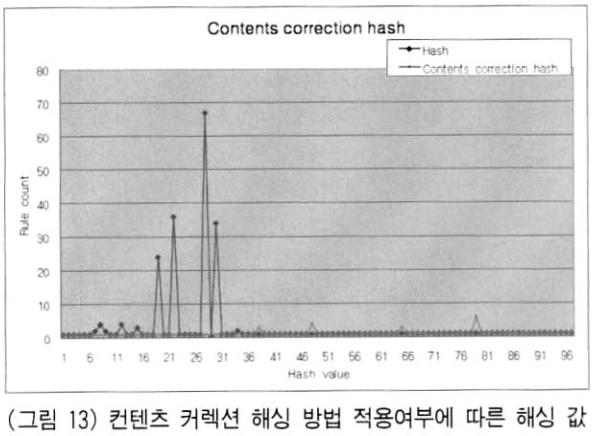
〈표 3〉 실험용 소프트웨어 항목

	버 전	수량	비 고
운영체제	2.6.16.18	1	리눅스
SNORT	2.4.5	1	-
룰(Rules)	2006. 8 현재	1000개	국내 H사가 제공한 공격 패턴

(그림 12)는 실험 환경을 나타낸다. 기존의 알고리즘과의 비교를 수행함에 있어, 오픈소스 침입탐지 및 방지 시스템으로 널리 사용되고 있는 SNORT와의 비교를 통하여 각 실험 상황에 대해 비교, 평가 하였으며, 사용된 소프트웨어의 항목 및 버전은 <표 3>과 같다.

4.2 해싱 방법에 대한 비교

(그림 13)은 <표 3>에서 설명한 1000개의 룰에 대하여 컨텐츠 커렉션 해싱을 적용하지 않았을 때와 적용하였을 때의 해싱 값 분포도를 나타낸다. 일반적인 해싱 기법을 적용하였을 경우에는 해싱 값의 분포가 특정 부분에 상당히 집중되는 모습을 보여주고 있다. 이는 앞서 설명했듯이 성능 저하의 원인이 되며, 알고리즘의 성능예측에 중대한 악영향을 미치는 요인이 된다. 하지만 컨텐츠 커렉션 해싱을 적용한 후의 모습은 해싱 값이 집중되지 않고 고른 분포를 보이고 있으며, 이는 제안된 알고리즘이 원하는 성능을 낼 수 있도록 하는 역할을 한다. 또한 이는 시그너처간의 상관관계가 아주 강한 경우에도 성능을 보장할 수 있음을 의미하며, 룰의 개수가 많아질 경우에도 최적의 분포를 나타낼 수 있도록 하여 룰의 확장성 및 안정성을 유지할 수 있도록 해준다.

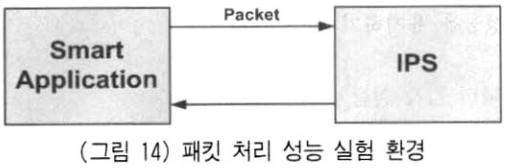


(그림 13) 컨텐츠 커렉션 해싱 방법 적용여부에 따른 해싱 값 분포

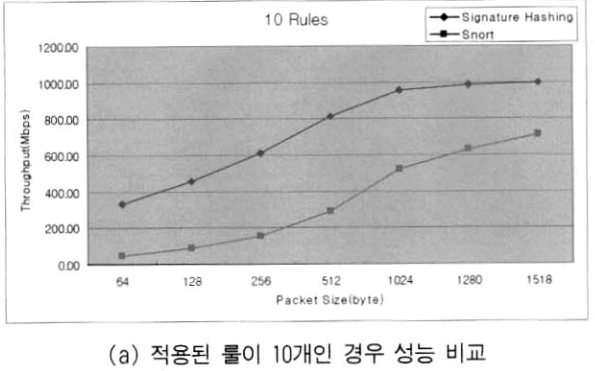
4.3 패킷 처리 성능 실험

(그림 14)는 패킷 처리 성능을 실험하기 위한 구성이다. 제안된 알고리즘과 기존 알고리즘에 대해 각 패킷 사이즈별 및 적용 룰 개수 별 패킷 처리 능력에 대한 비교 테스트를 진행하였다. 이를 통해 룰의 확장성과 네트워크 성능 운용에 대한 안정성에 대해 평가 한다. 실험 방법은 (그림 12)의 실험환경에서 적용 룰의 개수를 각각 10, 100, 500, 1000개로 조정해 가며, 각 상황 별로 64, 128, 256, 512, 1024, 1280, 1518 byte의 사이즈를 가진 패킷을 1Gbps의 속도로 전송하여 해당 트래픽에서 처리 여부를 평가한다.

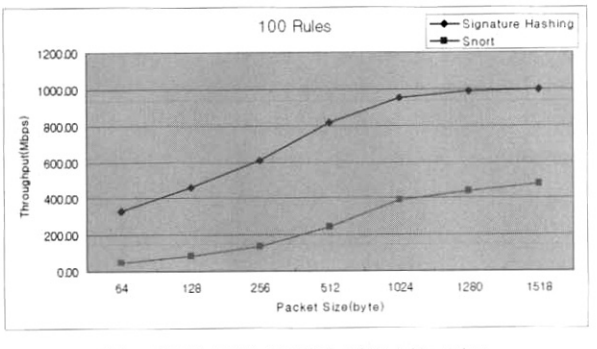
(그림 15)는 패킷 처리 성능 실험에 대한 결과를 나타낸다. 결과 그림에서 알 수 있듯이 기존의 알고리즘과 제안된 시그너처 해싱 기법을 사용하였을 경우, 룰의 개수가 각각 10, 100, 500, 1000개가 적용되었을 경우에 각 상황별로 패킷



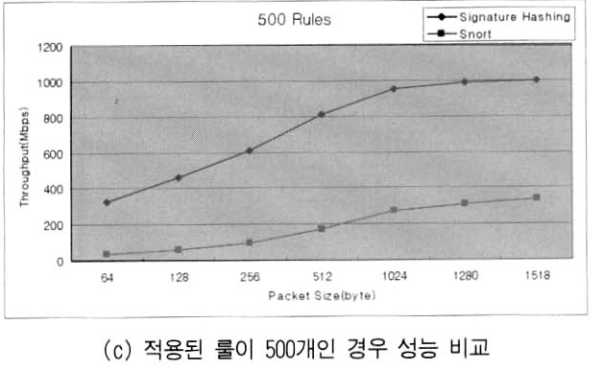
(그림 14) 패킷 처리 성능 실험 환경



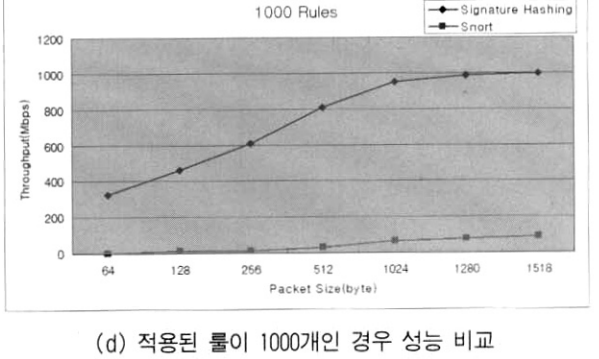
(a) 적용된 룰이 10개인 경우 성능 비교



(b) 적용된 룰이 100개인 경우 성능 비교



(c) 적용된 룰이 500개인 경우 성능 비교



(d) 적용된 룰이 1000개인 경우 성능 비교

(그림 15) 룰 개수 및 패킷 사이즈 변화에 대한 패킷 처리 성능 비교 그래프

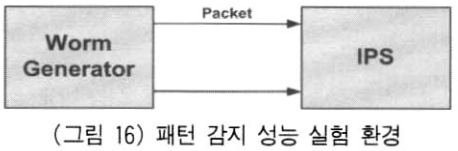
을 처리하는 성능을 비교해 보면, 기존 알고리즘의 경우 룰의 적용 개수가 전체적인 패킷 처리 성능에 영향을 미치고 있음을 알 수 있다. 제안된 알고리즘의 경우 기존 알고리즘에 비해 룰의 개수가 많아지더라도 전체 네트워크 성능에 큰 차이 없이 고르게 향상된 성능을 보이는 것을 확인할 수 있으며, 기존의 알고리즘의 경우 룰의 개수가 늘어날수록 네트워크 성능이 낮아지며, 전반적으로 제안된 알고리즘에 비해 낮은 성능을 보이는 것을 확인할 수 있다.

또한 패킷 사이즈의 변화, 즉 초당 처리 패킷 개수의 변화도 룰의 개수에 관계없이 항상 일정한 형태의 추이를 나타냄으로써 트래픽의 변화에 따른 시스템의 성능에 대해 예측 가능한 모습을 나타낸다. 각 패킷 사이즈 별 차이점은 존재하나 평균적으로 기존 알고리즘에 비해 제안된 알고리즘은 룰의 개수에 대해 10개일 때 약 280%, 100개일 때 약 330%, 500개일 때 약 480%, 2,500%의 성능 향상이 나타남을 확인할 수 있으며, 이를 통해 룰의 개수가 많아짐에 따라 그 성능의 차이는 더욱 더 크게 나타남을 알 수 있다. 이는 기존의 알고리즘이 룰의 개수에 영향을 받아 적용되는 룰의 개수가 많아짐에 따라 기하급수적으로 성능이 감소하는 반면, 제안된 알고리즘은 룰의 개수에 영향을 받지 않아 항상 같은 성능을 유지하기 때문에 나타나는 차이이다.

4.4 패턴 감지 실험

침입방지 시스템으로서 가장 중요하고 기본적인 역할은 공격에 대해 정확하고 효율적인 판단과 그 검출이라고 할 수 있다. 효과적인 패턴 매칭 알고리즘은 그 처리 속도나 룰 적용 개수와 상관없이 어떠한 경우에도 정확히 패턴에 매칭 되는 패킷을 찾아내어 요구하는 동작을 즉시 수행할 수 있어야 하며, 이를 충족시키지 못할 경우 아무리 성능이 좋은 알고리즘이라 할지라도 침입방지 시스템에 적합한 알고리즘이라 할 수 없기 때문에 트래픽 중 일부를 패턴에 매칭 되는 형태로 제작하여 테스트 하였을 경우 정상 패킷의 탐지(false-positive)나 탐지 대상 패킷을 탐지하지 못하는 상황이 발생하는지에 대한 여부가 중요한 평가 기준이 된다. 이를 실험하기 위해 웜 패킷을 생성해내는 웜 발생기를 이용하여, 실제 트래픽 중 웜의 비율을 10%~100%까지 변화를 시켜가며 패킷을 보냈을 때, 제안된 알고리즘과 기존 알고리즘이 정확히 패턴을 감지해 낼 수 있는 가를 평가한다. (그림 16)은 패턴 감지 테스트의 실험환경을 나타낸다.

<표 4>는 패턴 감지 테스트의 실험 결과를 나타낸다. 표에 나타난 바와 같이 10%에서 100%에 이르기까지 10%씩 매칭 되는 패턴을 가진 패킷의 비율을 증가시켜가며 패킷을 생성하여 시스템에 보낸 결과 기존의 알고리즘과 제안된 알고리즘 모두 동일하게 패턴을 감지해 냄을 알 수 있다. 이는



(그림 16) 패턴 감지 성능 실험 환경

〈표 4〉 시그너처 패킷 분포도 및 검출 비율

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Signature Hashing	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Snort	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%

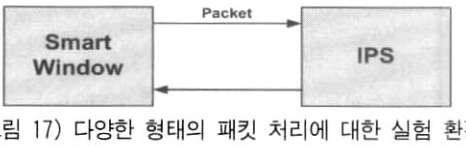
제안된 알고리즘이 패턴 감지 능력에 손상 없이 패킷 처리 성능을 향상시켜 신뢰도를 유지하며, 성능을 개선한 알고리즘임을 알 수 있다.

4.5 다양한 형태의 트래픽 처리 실험

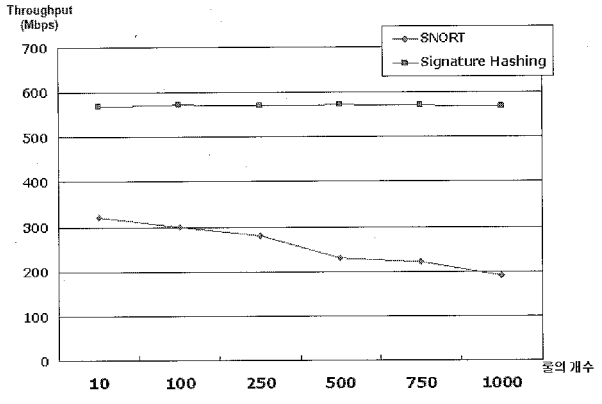
실제 네트워크 환경에서의 동작을 고려해 볼 때, 일반적인 네트워크의 트래픽 형태는 다양한 프로토콜 및 포트를 가지고 있으며, 이러한 경우에 다양한 종류와 사이즈의 패킷들 가운데 정확히 원하는 패킷을 찾아내는 것은 실제 네트워크에서 굉장히 중요하다. 제안된 알고리즘과 기존 알고리즘에 대해 적용되는 룰의 개수를 변화시켜 가며, 패킷 발생기를 이용하여 다양한 형태의 패킷을 통과시켜 보며 룰의 개수에 따른 네트워크 처리 능력을 실험한다. 다양한 형태를 가진 패킷 분포를 만들기 위해 <표 5>와 같은 분포를 가진 패킷을 생성하여 침입방지 시스템을 통과하도록 하였을 때, 해당 패킷에 대해 처리하는 성능을 비교하여 각 알고리즘에 대한 성능을 평가한다. (그림 17)은 일반 네트워크 트래픽에 대한 실험 환경을 나타낸다.

〈표 5〉 패킷 발생기의 패킷 분포도

패킷 길이	프로토콜	비 율
90	TCP	7% (25:3%, 80:4%)
90	UDP	7%
90	Otherwise IP	4%
128	TCP	5% (25:2%, 80:3%)
128	UDP	3%
128	Otherwise IP	1%
256	TCP	5% (25:2%, 80:3%)
256	UDP	3%
256	Otherwise IP	1%
512	TCP	12%(25:6%, 80:6%)
512	UDP	6%
512	Otherwise IP	3%
1024	TCP	13%(25:6%, 80:7%)
1024	UDP	5%
1024	Otherwise IP	3%
1500	TCP	13%(25:6%, 80:7%)
1500	UDP	6%
1500	Otherwise IP	3%



(그림 17) 다양한 형태의 패킷 처리에 대한 실험 환경



(그림 18) 다양한 형태의 트래픽에서 패킷 처리 능력 비교

다양한 형태의 트래픽에 대한 실험 결과 (그림 18)과 같은 결과를 얻을 수 있었다. 그림에서 볼 수 있듯이 기존의 알고리즘은 적용되는 룰의 개수가 증가함에 따라 처리 성능 역시 지속적으로 감소하고 있는 모습을 보인 반면, 제안된 알고리즘은 적용되는 룰의 개수에 관계없이 일정한 성능을 보이고 있다. 이는 제안된 알고리즘이 다양한 형태의 트래픽을 처리함에 있어 적용되는 룰의 개수에 대한 확장성을 가지고 있다는 것을 보여준다.

5. 결론 및 향후 연구 방향

본 논문에서는 향후 보다 빠르고 안정적으로 패킷을 검사할 수 있도록 하기 위해 시그너처 해싱 기법을 적용한 알고리즘을 제안하였고, 기존에 사용되던 알고리즘과의 비교를 통해 예측 가능한 높은 성능을 낼 수 있음을 확인하였다. 기존에 사용되던 알고리즘은 트리구조를 사용하고, 이전의 시그너처 결과를 기억함으로써 시그너처 경우의 수를 줄이는 등의 시도로 성능을 향상시키는 효과를 가져왔지만, 시그너처의 상관관계에 종속성이 강하고, 룰의 개수가 많아질수록 성능 수치가 낮아지는 단점이 있다. 이는 향후 계속적으로 알려진 공격과 패턴이 늘어날 것을 비추어 볼 때 확장성에 대해 심각한 문제로 인식될 수 있는 상황이다. 제안된 시그너처 해싱 기법을 이용하면 룰의 개수에 영향을 받지 않고 실제 시그너처에 수행되는 룰의 경우의 수를 줄임으로써 룰이 적게 걸린 것과 같은 효과를 내서 결국 룰의 개수와 관계없는 예측 가능하고 높은 성능을 내는 것을 확인하였다.

그러나 해싱 기법을 사용하면 룰의 상관관계가 깊을수록 같은 시그너처 해싱 값을 가질 가능성이 높아지고, 하나의 해싱 값에 많은 룰이 연결되면 이는 곧 성능저하와 연결될 가능성이 있다. 이를 해결하기위해 컨텐츠 커렉션 해싱 기법을 사용하였지만 룰의 상관관계가 매우 높은 상황이라면 랜덤하게 해싱 값을 구하는 방식은 항상 같은 성능을 낼 수 있도록 보장되기 어려운 것이 현실이다. 또한 기존의 알고리즘에 비해 포트 및 시그너처 해시 테이블을 유지하고 관리함에 따라 기존의 방법에 비해 상대적으로 큰 메모리를 차지하게 되는 측면도 존재한다. 고성능의 침입방지 시스템을 비롯하여 패킷을 깊은 수준에서 분석하고 확인하는데 있어 매칭 알고리즘은 네트워크 성능이 증가하고 그 처리속도가 중요시되는 현실에 비추어 볼 때 매우 중요한 부분임에 분명하다. 이러한 새로운 기법을 이용하면 기존에 비해 훨씬 효율적이고 안정적인 네트워크 보안 구축이 가능하리라 사료된다.

향후 연구 방향을 요약하면 다음과 같다.

- 룰의 상관관계와 각 알고리즘의 성능 분석 : 기존의 알고리즘은 룰의 상관관계가 높을수록 좋은 성능을 보였고, 시그너처 해싱 기법의 경우 크지는 않지만 룰의 상관관계가 낮을수록 좋은 성능을 보였다.
- 높은 룰의 상관관계에 대한 성능 개선 : 룰의 상관관계가 극도로 높은 경우에도 해싱 값의 분포를 조절하여 최고의 성능을 낼 수 있도록 보완하는 연구가 필요하다.
- 정규 표현식에 대한 지원 : 시그너처 해싱의 경우 정규표현식으로 구성된 시그너처에 대해 처리하는데 상당한 난점이 존재한다. 향후 연구를 통해 이를 해결할 수 있는 방안을 모색한다.

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] M. Gokhale, D. Dubois, A. Dubois, M. Boorman, S. Poole, and V. Hogsett, "Granidt: Towards Gigabit Rate Network Intrusion Detection Technology", The 12th International Conference on Field-Programmable Logic and Applications, Sep., 2002.

[4] Y.H Cho, S. Navab, and W. H. Mangione-Smith, "Specialized Hardware for Deep Network Packet Filtering", The International Conferene on Field Programmable Logic and Applications, Sep., 2002.

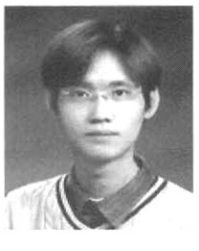
[5] I. Sourdis and D.Pnevmatikatos, "Fast, Lage-Scale String Match for a 10Gbps FPGA-based Network Intrusion Detection System", The 13th International Conference on Field Programmable Logic and Application, Sep., 2003.

[6] I. Sourdis and D.Pnevmatikatos, "Pre-decoded CAMs for Efficient and High-Speed NIDS Pattern Matching", tThe 12th Annual IEEE Symposium on Field Programmable Custom Computing Machines, Apr., 2004.

[7] Netfilter, http://www.netfilter.org.

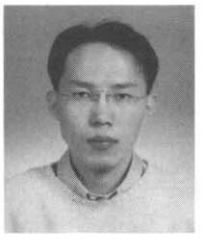
[8] A. Aho, M. Corasick, "Efficient string matching: an aid to biliographic search", Comm. ACM. 18:333-40, 1975.

[9] 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.
[10] Snort. http://www.snort.org/
[11] 김선일, "네트워크 침입방지 시스템을 위한 고속 패턴 매칭 가속 시스템", 정보처리학회논문지 A, 제12-A권 제2호, Apr., 2005.
[12] J. Lockwood, "Fast and Scalable Pattern Matching for Content Filtering", Architectures for Networking and Communication System(ANCS), Oct., 2005.
[13] J. Moscola, J. Lockwood, R. P. Loui, and M. Pachos, "Implementation of a Content-Scanning Module for an Internet Firewall", The 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Apr., 2003.
[14] 전용희, "침입방지시스템(IPS)의 기술 분석 및 성능평가 방안", 정보보호학회지, 제15권, 제2호, Apr., 2005.
[15] An NSS Group Report V 1.0, "Intrusion Prevention Systems(IPS), Group Test", NSS, Jan., 2004.
[16] Smartbits, http://www.spirentcom.com/



왕 정 석

e-mail : wang@q.ssu.ac.kr
2004년 숭실대학교 정보통신전자공학부 졸업(학사)
2007년 숭실대학교 전자공학과 졸업(석사)
2004년~현 재 (주) 펌킨넷코리아 중앙연구소 팀장
2007년~현 재 숭실대학교 전자공학과 박사과정
관심분야 : 네트워크 보안, 이중화 및 부하 분산

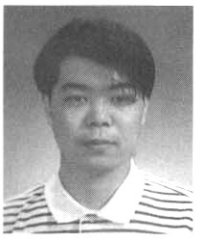


곽 후 근

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

정 윤 재

e-mail : jgyver@pumpkinnet.co.kr
2002년 숭실대학교 정보통신전자공학부 졸업(학사)
2004년 숭실대학교 전자공학과 대학원 졸업(석사)
2003년~현 재 (주) 펌킨넷 코리아 중앙 연구소 팀장
관심분야 : 클러스터링, 네트워크 컴퓨팅, 네트워크 어플리케이션 가속 및 스위칭



권 희 웅

e-mail : hukwon@pumpkinnet.co.kr
1997년 숭실대학교 정보통신전자공학부 졸업(학사)
1999년 숭실대학교 전자공학과 졸업(석사)
2007년 숭실대학교 전자공학과 박사 수료
2000년~현 재 (주) 펌킨넷 코리아 중앙 연구소 실장
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