논문 2006-43TC-7-10

버퍼 오버플로우 웜 고속 필터링을 위한 네트워크 프로세서의 Bloom Filter 활용

(A Bloom Filter Application of Network Processor for High-Speed Filtering Buffer-Overflow Worm)

김 익 균*, 오 진 태*, 장 종 수*, 손 승 원*, 한 기 준**

(Ikkyun Kim, Jintae Oh, Jongsoo Jang, Sungwon Sohn, and Kijun Han)

요 약

컨텐츠를 기반으로 인터넷 웜등의 유해 패킷을 네트워크에서 차단하는 기술은 탐지의 정확도와 네트워크 성능의 한계 극복이라는 두 가지 문제에 초점이 맞추어져 있다. 특히 멀티 기가비트 성능을 기본으로 하는 현재의 전달 네트워크에서 고속으로 웜 트래픽을 차단하는 능력이 주요 이슈로 대두되고 있다. 본 논문은 라우터 혹은 방화벽과 같은 통신 및 보안 장비에 주요 기술로 사용되는 네트워크 프로세서 환경에서 멀티 기가비트 수준으로 고속 웜 필터링이 가능한 구현 구조를 제안한다. 고속 웜 필터링을 위한 설계의 특징으로는 네트워크 프로세서가 가지는 내부 레지스터와 메모리의 자원 한계점을 극복하기 위하여 Bloom Filter를 활용하였고, 특히 버퍼 오버플로우 기법을 이용하는 웜들에 대해 단순 패턴매칭 뿐만 아니라, 유해 코드의 길이 검사를 수용할 수 있는 구조로 시그너처 관리가 확장 가능하도록 설계되었다. 설계된 고속 웜-필터링 구조를 기가비트 이더넷 인터페이스를 가진 Intel IXP 네트워크 프로세서 플랫폼에서 마이크로 코드형태로 구현하였고, 알려진 웜들이 포함된 트래픽을 사용하여 그 성능을 분석하였다

Abstract

Network solutions for protecting against worm attacks that complement partial end system patch deployment is a pressing problem. In the content-based worm filtering, the challenges focus on the detection accuracy and its performance enhancement problem. We present a worm filter architecture using the bloom filter for deployment at high-speed transit points on the Internet, including firewalls and gateways. Content-based packet filtering at multi-gigabit line rates, in general, is a challenging problem due to the signature explosion problem that curtails performance. We show that for worm malware, in particular, buffer overflow worms which comprise a large segment of recent outbreaks, scalable -- accurate, cut-through, and extensible -- filtering performance is feasible. We demonstrate the efficacy of the design by implementing it on an Intel IXP network processor platform with gigabit interfaces. We benchmark the worm filter network appliance on a suite of current/past worms, showing multi-gigabit line speed filtering prowess with minimal footprint on end-to-end network performance.

Keywords : 버퍼 오버플로우, 웜, 방화벽, Bloom Filter, 네트워크 프로세서

I. 서 론

네트워크 웜 공격에 대처하는 방법들로 호스트 시스템의 취약점 패치를 통하여 공격에 대비하는 해결점을 보완하기 위해 네트워크 인입점에서 웜 패킷을 차단하는 기술들이 라우터나 네트워크 방화벽과 같은 통신 및 보안 장비에 적용되고 있다. 네트워크에서 컨텐츠를 기반으로 인터넷 웜등의 유해 패킷을 차단하는 기술은 변형 공격에 대한 탐지의 정확도와 네트워크 성능의 한계 극복이라는 두 가지 문제에 초점이 맞추어져 있다. 즉, 멀티 기가비트 성능을 기본으로 하는 현재의 전달 네트

* 정회원, 한국전자통신연구원, 정보보호연구단
(ETRI, Information Security Division)
** 정회원, 경북대학교 컴퓨터공학과
(Kyungpook National University, Computer Engineering Department)
접수일자: 2006년6월15일, 수정완료일: 2006년7월14일

워크에서 선로 속도로 변형된 웜 트래픽을 탐지 및 차단하는 능력이 주요 이슈로 대두되고 있다.

변형된 공격에 대한 탐지정확도 측면을 고려해 보면, 시그너처 기반의 필터링 시스템들은 알려진 패턴이나 그 시나리오를 인입 트래픽과 비교하기 때문에 오류율이 비교적 낮다고 볼 수 있지만, 변형된 형태의 웜에 대해서는 대처하기 힘들뿐 만 아니라 각 경우의 시그너처를 사용할 때는 시그너처 급증(Signature Explosion) 문제에 당면하게 된다. 과다한 시그너처의 사용은 필터의 성능에도 악영향을 미치게 되고 현실적으로 적용이 힘든 상태에 까지 이르게 될 수 있다. 최근 연구[9,10]들에 따르면, 버퍼 오버플로우 웜이 지닌 불변의 성질들을 이용하여, 보다 단순한 시그너처를 생성할 수 있으며 이를 이용하여 고속 웜 필터에 쉽게 적용가능 한 구조를 제시할 수 있다. 그 대표적인 예로서, APE(Abstract Payload Execution)[9]에 의하면 버퍼 오버플로우 웜은 패킷 내에 유해한 실행 가능 코드(Malicious Executable Code)가 항상 존재하며, APE 방법론에 의해 유해 코드 길이를 탐지하고 MEL (Maximum Execution Length)로 정의한 후, 응용 서비스 별 평균 MEL과 비교하여 이상 유무를 판별할 수 있음을 보여주었다. 또한 APE 방법론을 확장한 styx[10]에서는 버퍼 오버플로우 웜 방어를 위한 시그너처까지 자동 생성하는 방안이 제시되고 있다.

반면에 Snort[12], Bro[16]와 같은 기존의 시그너처 기반 웜 필터들은 시그너처 생성 자체가 웜의 발생에 기반한 것 아라기보다는 각종 OS와 그 응용 프로그램의 취약성을 기준으로 시그너처가 생성되고 있다. 예를 들어 SNORT 2.0 기준으로 시그너처의 수를 보면 2500여 개가 운용되고 있고, 이들 대부분은 각 종 취약점을 기준으로 패이로드 내의 특정 패턴을 기준으로 시그너처가 관리되고 있다. 향후 증가하는 시그너처의 수와 그 변형 시그너처들은 시그너처 급증문제를 야기 시킬 것이고, 이는 날로 고속화되는 백본 네트워크에 적용하기에 걸림돌이 되고 있다.

본 논문에서는 버퍼 오버플로우 웜의 특성을 고려하여 고속 처리가 가능한 시그너처 표현방법을 제시하고, 이를 네트워크 솔루션으로 널리 사용되는 네트워크 프로세서의 웜 필터 응용으로 활용하기 위해 마이크로 코드로 구현하였고, Bloom Filter[2]를 사용하여 그 성능을 향상시켰으며, 두 가지의 네트워크 프로세서 환경에서 그 성능을 벤치마킹하였다. 본 논문에서 제안된 시그너처 표현 구조와 처리 알고리즘은 버퍼 오버플로우 기법의 웜에 최적화된 구조로서 시그너처 급증 문제를 해결할 수 있고 변형된 웜에 대응이 가능하며, 네트워크 프로세서에서 성능 문제점으로 지적된 패이로드 패턴 매칭으로 인한 처리 속도 저하 문제를 피할 수 있다.

이 후 논문의 구성은 제 II장에서 버퍼 오버플로우 웜과 고속 필터링에 관한 연구에 대해 기술하고, Bloom Filter의 소개와 그 활용 예를 기술하며, 제 III장에서는 본 논문에서 제안한 고속 웜 필터링을 위한 시그너처 표현과 네트워크 프로세서 기반 필터링 구조에 대해 기술하며, 성능 향상을 위해 사용된 Bloom Filter의 활용에 대해 설명하고, 제 IV장에서는 NP 기반으로 구현된 웜 필터링의 성능 벤치마킹에 대해 기술하며, 마지막으로 결론 및 향후 계획에 대해 다룬다.

II. 버퍼 오버플로우 웜과 관련 연구

1. 버퍼 오버플로우 웜

네트워크 보안 영역에서 흔히 회자되는 인터넷 웜에 대하여 명확하면서 전체적인 동감을 이끌어내는 "정의"가 아직까지도 없다고 볼 수 있지만, 웜이 가지는 주요 특징으로는 네트워크를 기반으로 전파되는 점과 자기 전파력을 가지고 있다는 점이다. 네트워크에서 콘텐츠를 기반으로 인터넷 웜을 차단하려는 시도는 바이러스에 비해 웜이 더욱 빠르고 효과적으로 차단될 수 있다고 인식되기 때문이다. 컴퓨터 바이러스는 특정 프로그램에 추가된 기생 코드로서, 프로그램 실행 제어를 바이러스 코드를 먼저 실행하여 컴퓨터를 감염시키고 자기 복제를 하는 특성이 있다. 이 바이러스 코드의 추가는 아주 다양한 형태를 취할 수 있으므로 정확하고 고속으로 탐지하기 어렵다. 즉 하드디스크와 메모리 상주 프로세스들을 전체적으로 탐색하여 바이러스 시그너처 패턴을 찾아내는 것이 유일한 방법으로 사용되고 있다. 또한 이 방법은 감염 후 제거되는 형태로서 사후 대응적인 해법으로 볼 수 있다. 이에 반해 인터넷 웜은 네트워크 프로토콜의 동작의 일부로 이동성을 가진 기생 코드로 활동한다. TCP/UDP 상위의 HTTP, RPC 등과 같은 상위 프로토콜의 취약성을 이용하여 동작하며, 이는 응용 계층이나 시스템 라이브러리 (e.g. Microsoft DLL)내에 존재하는 버퍼 (Stack, Heap) 오버플로우 취약점을 이용하는 방법이 대부분을 차지한다[17].

그림 1은 버퍼 오버플로우 기법을 이용한 웜의 전형적인 메모리 구조이다. NOP Zone은 유해 Shell Code가 실행될 확률을 높이기 위해 사용되는 Sled[11]의 한 형태

그림 1. 버퍼 오버플로우 웜의 일반적인 구조
Fig. 1. Typical structure of buffer overflow worm.

로 존재한다. CPU의 프로그램 카운터(PC)가 NOP Zone에만 위치시키면 유해 Shell Code까지 프로그램 카운터가 자동적으로 증가되어 실행된다. Return Address는 OS의 스택의 복귀 주소를 덮어쓰는 방식으로 CPU 제어를 가로채는 역할을 한다.

콘텐츠를 기반으로 버퍼 오버플로우 웜을 차단하는 방법이 바이러스를 차단하는 방법과 근본적으로 다른 점이 아래 두 가지 이유를 들 수 있다.

i) 프로토콜 임베딩 : 웜의 기생코드(Malicious Shell Code)는 네트워크 프로토콜 내에 임베딩 되기 때문에 컴퓨터 프로그램 내에 존재하는 것 보다 탐지하기가 더 쉬울 수 있다. 네트워크 프로토콜의 목적은 정보 교환이라는 기본적인 목적을 두고 설계되며, 웜은 이 프로토콜의 규정을 지키면서 각 단말에서 또 단말로 전파된다. 여기에서 웜은 그 기생코드를 프로토콜의 어디에 숨길 것인가 하는 문제에 그 한계를 느낄 것이다.

ii) 길이 불편성 : 버퍼 오버플로우 기법의 웜 기생코드를 포함하는 네트워크 프로토콜의 제약성 중 하나는 취약점을 이용하여 충분한 길이의 유해 코드가 전달되어야 한다는 점이다. 유해 코드는 암호화와 같은 다양한 형태로 변형[11]될 수 있지만, 그 길이 정보만큼은 변화될 수 없는 불편성을 가지고 있다.

2. 관련 연구

콘텐츠를 기반으로 네트워크 유해 트래픽을 차단하는 기술의 공통 사항은 정의된 시그너처를 기준으로 패이로드 패턴 매칭을 통하여 유해 트래픽을 탐지할 수 있는 능력이다. Snort[12], Bro[16]와 같은 소프트웨어 기반의 필터들은 기가비트와 같은 고속 네트워크 링크의 모든 트래픽들을 모니터링 할 만큼 충분한 성능을 보여주지 못하고 있어서[13], 네트워크 프로세서 기술이나 FPGA와 같은 하드웨어 기반 시스템[14, 15]들로 성능상의 해결점을 찾고 있다. 이들은 고속으로 패킷 검사를 하기위해 병렬 처리 기술을 이용할 수 있기 때문에 높은 성능을 보일 수 있다[1]. 하지만 이와 같은 하드웨어 기반 필터들의 단점은 시그너처 급증으로 인하여 운용 및 관리 문제가 있을 뿐만 아니라, 탐지 성능에도 저하 요소로 작용되며 또한 변형된 웜에 대처하기 힘든 상태이다. 이러한 단점은 가능하면 많은 시그너처를 수용하려는 구조에 대한 연구에서 각 웜의 특성을 고려한 시그너처 생성에 관한 연구들로 진행되고 있는 실정이다.

대표적인 예로서, APE (Abstract Payload Execution)[9] 방식을 이용하여 정상 트래픽의 패킷 페이로드 내에서 "수행 가능한 코드"의 평균 길이 (MEL)를 기준으로 정상 유무를 판정하고, 해당 유래 패킷에 대한 시그너처 생성에 활용하는 방식이 있다. APE방식을 활용한 styx[10]는 Fast static 분석을 통하여 변형 웜의 exploit 코드를 탐지하고, 새로운 시그너처 생성을 제안하고 있다. 이들은 버퍼 오버플로우 웜에 최적화 된 시그너처를 생성하는 형태이지만, 그 자체로는 고속 네트워크의 모든 패킷을 감시하여 차단할 만한 성능을 보여 주지 못하고 있다.

3. Blooming Filter

초기의 Bloom Filter[2]는 네트워크에서 전달되는 데이타의 양과 호스트 프로세서에서 수행되는 데이타베이스의 Tuple 프로세싱 양을 줄여 성능을 향상시키려는 고속 분산 데이타베이스 환경에서 사용되었다. 여기에서 Bloom Filter는 탐색 키가 존재하는 지를 점검하기 위한 질의 키의 집합을 Bit Vector로 표현한 것이다.

그림 2와 같이 Bloom Filter의 기본적인 동작 방식은 초기에 2^n개의 비트 벡터가 0으로 초기화 되어 있고, k개의 독립적인 해쉬 함수를 동일한 패턴에 매핑하여 각 인덱스에 해당하는 비트 벡터에 셋팅된 값이 모두 1일 경우에만 집합에 속하는 것으로 판정한다. 다중 해쉬 기법을 적용하여 False Positive를 줄이고, 비트 벡터를 활용하여 공간 효율을 극대화하였다고 볼 수 있다. 따라서 Bloom Filter가 적용될 수 있는 원칙[3]은 "구성원 점검이 요구되는 프로세싱에서 리스트(List)나 집합(Set)을 사용함에 있어 메모리와 같은 공간 제약사항이

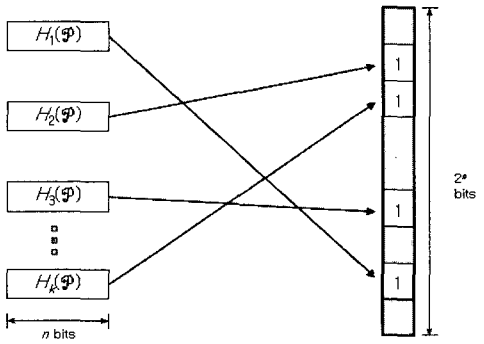


그림 2. Bloom Filter 동작
Fig. 2. Operation of the Bloom Filter.

발생할 때, Bloom Filter를 활용한다면 합리적인 수준의 False Positive를 가지면서, 공간 및 성능의 효율을 볼 수 있다"라는 점이다.

4. Blooming Filter의 응용

1970년에 Bloom Filter가 개발된 후 다른 영역에서는 거의 사용하지 않고 있다가, 최근 웹 캐쉬[4], P2P 네트워크의 리소스 라우팅[5], 네트워크 보안 IP 추적[6]등과 같은 네트워크 응용에 광범위하게 활용되고 있다. 본 절에서는 이들 중 네트워크 보안에 관련된 응용만을 살펴본다.

가. IP Traceback

네트워크에서 얻은 패킷의 경로정보를 추적하는 한 가지 방법으로는 각각의 라우터에서 포워딩한 모든 패킷의 정보를 기록해 놓은 후, 각 라우터가 주어진 패킷이 자신을 통과하는 지를 점검하는 방식일 것이다. 이 방법은 아주 단순하면서 공간 자원 측면에서 소모적인 방식이지만, Bloom Filter를 통해 패킷 집합을 요약하여 정보의 양을 최소화한 방식이 Source Path Isolation Engine(SPIE)[6]라는 이름으로 소개 되었다.

나. 네트워크 패킷 검사

네트워크 패킷의 페이로드에 대하여 미리 정의된 시그너처를 검사하기 위하여 하드웨어 기반의 FPGA 로직으로 구현함에 있어 Bloom Filter를 활용하였다. 미리 정의된 시그너처를 그 길이별로 병렬 Bloom Filter로 표현하여, 하드웨어 병렬처리의 해쉬 기법을 사용하여 2.4 Gbps 성능을 보여주었다[7]. 이는 병렬 처리가 가능한 하드웨어의 특성에 Bloom Filter의 다중 해쉬 방법이 적절하게 적용되었다고 볼 수 있다.

III. NP 기반 웜 필터 구조

본 장에서는, 인터넷 웜 필터링을 고속으로 수행하기 위해 Intel 네트워크 프로세서 환경에서 Bloom Filter가 다른 형태로 활용되는 것에 대해 기술한다. 먼저 웜 필터링 구현 및 벤치마킹한 두 가지 모델의 IXP 네트워크 프로세서에 대해 간단히 기술하고 구현 구조를 제시한다.

1. Intel IXP1200 과 IXP2800 구조

Intel IXP1200 네트워크 프로세서 모델은 초기 네트워크 프로세서 모델로서 Radisys 사에서 보급한 ENP 2506 (Gigabit Interface) 개발 환경을 사용하였다. IXP1200은 각 4개의 하드웨어 Thread를 가진 6개의 RISC 마이크로 엔진이 MIMD방식으로 병렬 프로세싱하는 구조로 되어 있다. 각 마이크로 엔진은 1-2K 인스트럭션 스토어를 가지고 있고, 4KB 스크래치 패드라는 온칩 메모리를 공유할 수 있어서 각 마이크로 엔진에서 고속 메모리 엑세스가 가능하다. Gigabit 이더넷 IXP1200 은 두개의 마이크로 엔진이 Ingress트래픽을, 또 다른 마이크로 엔진이 Egress 트래픽을 담당한다. 각 마이크로 엔진은 4개의 하드웨어 Thread를 통하여 Zero-Time 콘텍스트 스위칭이 가능하여 마이크로엔진이 메모리 I/O 수행 시 효율적인 프로세싱이 보장된다. StrongArm은 ARM 프로세서 기반의 범용 CPU이다. 이는 최고 232Mhz까지 동작하며, 최고 16Kbyte의 인스트럭션 캐쉬와 8Kbyte의 데이터 캐쉬를 가지고 있다.

10Giga 이더넷 인터페이스의 처리능력을 가진 IXP2800의 기본적인 구조는 IXP1200과 개념적으로 동일한 구조이지만, 고속 처리를 위한 마이크로 엔진의 구조와 메모리 인터페이스가 완전히 달라진 형태라고 볼 수 있다. 먼저 병렬 처리가 가능한 마이크로 엔진이 16개로서 8 하드웨어 Thread 가 구현되어 있고, 마이크로 엔진의 간의 통신이 내부 Scratch 메모리에 16개 하드웨어 링 버퍼를 통해 이루어진다. XScale Core는 임베디드 RISC 범용 CPU이다. 이는 최고 700Mhz까지 동작하며, 최고 32Kbyte의 인스트럭션 캐쉬와 32Kbyte의 데이터 캐쉬를 가지고 있다. RDRAM은 네트워크 패킷을 저장하는데 사용되는데, 64bit 데이터 버스로 최고 800Mhz (혹은 1066Mhz) 로 동작하므로 최고 16 Gib/s 처리율로 동작이 가능하다. QDR SRAM은 패킷 포워딩 테이블등의 각종 데이터 구조를 저장하기 위해 사용되는데, 32bit 데이터 버스로 250 Mhz로 동작한다. 스위치 및 외부 네트워크 인터페이스로 SPI-4 또는 CSIX 인터페이스를 지원하는데 채널당 500Mhz까지 동작가능하다.

2. 웜 필터링 시그너처 표현

본 논문의 네트워크 프로세서 기반의 웜 필터 구조는 버퍼 오버플로우 기법 웜의 특성을 최대한 활용하여 시그너처 자료 구조를 최적화하였고, 이를 통하여 내부 메모리 엑세스 타임을 최소화함으로서 처리 속도를 향상 시켰다. 웜 시그너처를 표현하기 위해 그림 3과 같이 한 쌍의 32bit-Cell이 오퍼레이션의 최소 단위가 되

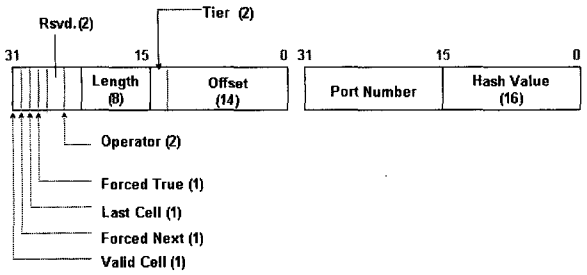


그림 3. 웜 시그너처 Cell 구조
Fig. 3. Cell structure of worm signature.

고 1개의 웜 시그너처를 표현하기 위해서는 최대 4 쌍의 Cell이 상용될 수 있다.

Cell 제어 표시자는 MSB 4 bits로서 Valid Cell, Forced Next, Last Cell, Forced True로 구성된다. Valid Cell 은 해당 Cell이 유효한지를 나타내고, 초기화시에는 모두 0으로 리셋되며 시그너처 캐쉬에 추가될 때 1로 유효 비트가 셋팅된다. Forced Next 비트는 시그너처의 인덱스가 해쉬 상에서 충돌이 발생하였을 경우, 다음 인덱스을 사용할 수 있도록 보장을 해준다. 기본적으로 시그너처 관리 평면에서 해쉬 인덱스 충돌이 발생되지 않도록 분배를 하고 재 조정을 하지만 최후에도 재 조정이 불가능할 때를 대비하여 사용되는 비트이다.

Last Cell 비트는 최대 4 쌍의 Cell로 구성되므로 마지막 Cell임을 표현하여 처리 동작이 정지 될 수 있도록 한다. Forced True 비트는 시그너처 간의 공통 부분을 처리하여 ORing 연산이 가능하도록 하기 위한 비트이다. 특히 이는 TCP 80과 같은 많은 시그너처가 존재하는 Cell인 경우에 유용하게 활용된다. 예를 들면, TCP 80 GET으로 시작하는 웜이 다수 존재하므로 GET 이후에 발생되는 Cell에 대해 불일치가 일어나더라도 다른 웜 규칙을 위해 프로세싱을 계속 진행하기 위함이다. 본 논문의 Cell 표현의 특징 중 하나는 Operator가 시그너처 표현에 내장된다는 점이다. 웜의 특성 중 하나 인 패이로드 길이 정보를 비교하기 위해, Equal 연산자 이외에 부등호 연산자를 두어 패턴에 대한 Hash 값 외에 APE방법에서 사용되는 MEL[9]과 같은 각 필드의 길이 정보를 비교할 수 있도록 하기 위함이다. Length 필드는 비교할 패턴의 길이 정보를 표현하고, Tier 필드와 Offset 필드는 패킷내 패턴의 위치 정보를 나타내고, Port Number는 TCP, UDP의 포트 번호를 나타낸다. 해쉬 필드는 연산자가 Equal일 경우에 비교 패턴의 CRC16 해쉬 결과 값을 나타내고, 부등호일 경우에 길이 정보에 대한 값을 나타낸다.

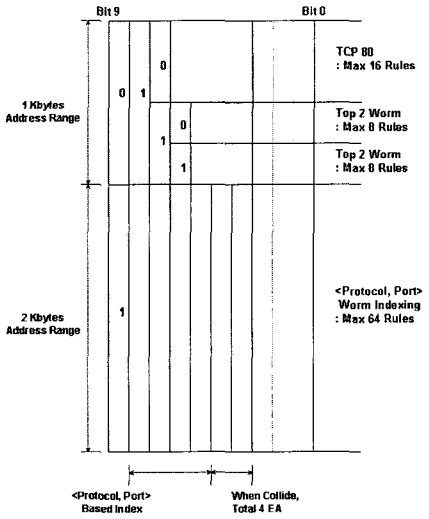


그림 4. 시그너처 캐쉬 분포 예
Fig. 4. An example of signature Layout in the Rule Cache.

그림 4는 내부 Scratch 메모리에 최소의 시그너처를 배치할 때 Cell 배치 구조에 대한 일예이다. Scratch 메모리를 이용한 룰 캐쉬에는 최소한의 Cell들이 배치되는데 TCP 80과 같은 트래픽의 큰 볼륨을 차지하는 영역과 그 이외의 프로토콜 및 서비스 포트에 대한 Hash 인덱스를 이용하여 직접 Addressing이 가능하도록 한다. 시그너처 관리 평면에서 LRU(Least Recently Used) 개념으로 시그너처를 갱신할 경우에는 속도가 가장 빠른 Scratch 메모리에 적용이 가능하다. 하지만, IXP2800과 같은 경우에는 속도가 빠른 Scratch 메모리가 마이크로 엔진간의 통신용 하드웨어 링 버퍼로 사용하기 있기 때문에 SRAM에서 운용된다.

3. Bloom 필터의 응용

본 논문에서 성능 향상을 위해 Bloom 필터를 활용하는 기본 전제는 다음과 같다. 인입 트래픽의 조합에 대하여 자주 발생하면서 가장 중요한 트래픽(예, TCP 80)에 대하여, 정상 트래픽과 비정상 트래픽을 구분하기 위한 방법으로 Bloom Filter를 사용함으로서 정상 트래픽에 대한 판별을 최우선으로 수행시킨다는 점이다. 이는 웜 트래픽으로 될 가능성이 높은 트래픽에 대해서는 많은 프로세싱이 이루어지므로, Bloom Filter를 이용하여 정상 트래픽에 대해 구성원 점검을 수행하여 판별하게 한다. 본 논문에서는 두 가지 종류의 Bloom Filter를 사용되는데, VC(Virtual Conjunction)에 대한 것과 <프로토콜, 포트>에 대한 것이다. VC는 TCP 포트 80 과

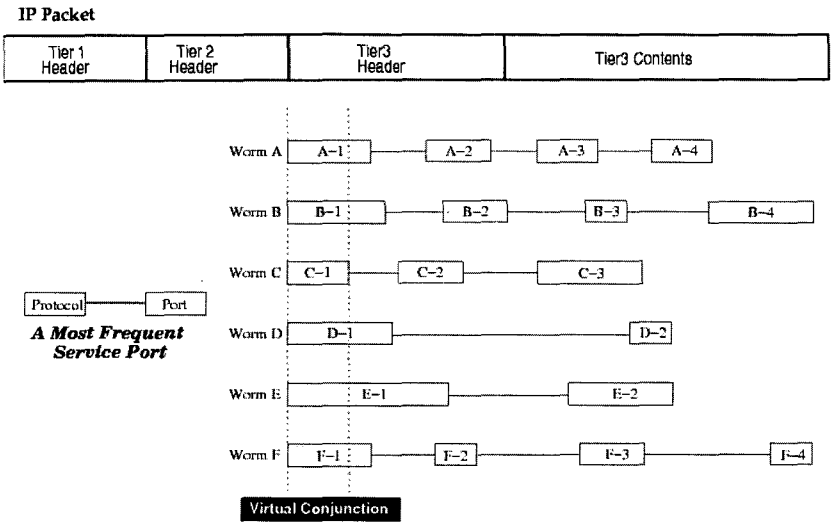


그림 5. Virtual Conjunction 개념도
Fig. 5. Conceptual diagram of the virtual conjunction.

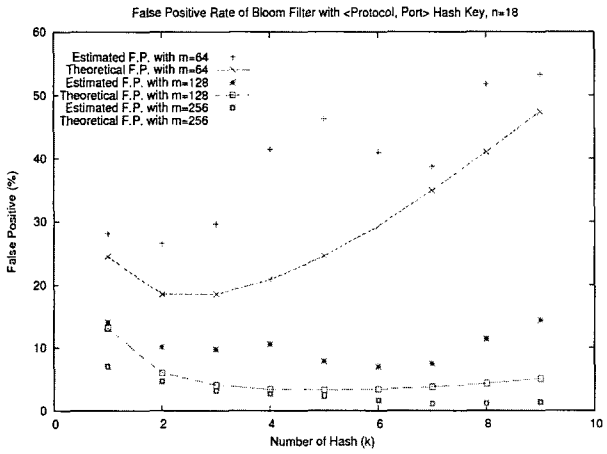


그림 6. <프로토콜, 포트> 기반의 Bloom Filter False Positive
Fig. 6. False positive of the <protocol, port>-based bloom filter.

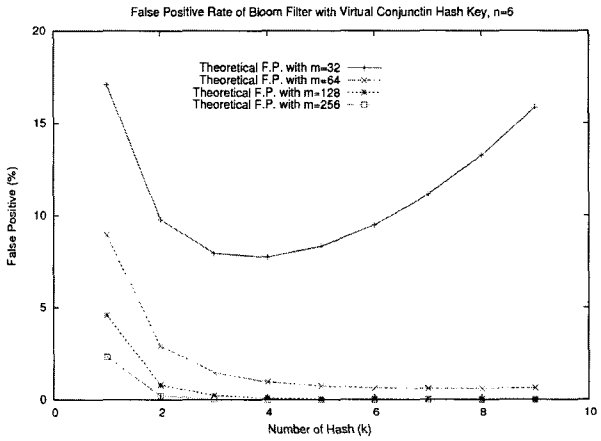


그림 7. HTTP Port Virtual Conjunction 기반의 Bloom Filter False Positive
Fig. 7. False positive of the virtual conjunction-based bloom filter for HTTP port.

같이 높은 트래픽비율을 차지하는 서비스에 있어서 많은 웜이 발생하고 이들 웜에 대한 시그너처 패턴의 공통 분포가 많이 발생된다는 규칙에 의해 구성된 공통 패턴의 일부이다. 이들에 대한 Bloom Filter를 메모리 레벨보다 빠른 레지스터 레벨로 구현하여 웜 트래픽이 아닌 정상 트래픽을 고속으로 판별하는데 사용하였다. 그림 5는 웜 시그너처에 대한 Virtual Conjunction의 개념을 나타낸다. VC는 동일한 서비스 포트에서 발생할 수 있는 시그너처들 중 응용 계층 프로토콜 헤더에 해당되는 공통된 패턴들의 모임으로 정의된다. 앞 절에서 기술한 것처럼 웜의 특징 중의 하나가 프로토콜 임베딩이고, 이는 응용 계층 헤더에서 동일하게 적용되기 때문에 서비스의 양이 크면 클수록 동일한 서비스 포트에 대해 VC가 존재할 가능성이 높아진다고 할 수 있다.

Bloom Filter에 있어서 가장 중요한 평가 요소로 적용되는 부분이 F.P(False Positive) 비율이다. 본 논문의 Bloom Filter 활용에서 F.P의 의미는 웜이 아니면서 웜으로 인식되어 불필요한 프로세싱 경로를 부가적으로 수행할 확률로서 이 값이 높을수록 성능 저하요소로 작용하게 된다. n개의 웜 규칙에 대한 k개의 독립된 해쉬 함수를 사용하고, 그 인덱스의 범위가 $\{1,2,\ldots,m\}$인 경우의 이론적인 F.P 확률은 $(1-e^{-kn/m})^k$ 이다[3]. 그림 7은 HTTP 서비스의 VC기반 Bloom Filter를 위한 실질적인 k와 m의 파라메터 값에 대한 시뮬레이션 결과를 보여준다. 여기에서 해쉬 함수의 개수(k)는 3개, 룰 고속 캐쉬에 저장되는 인덱스 총 수 (m)를 64개로

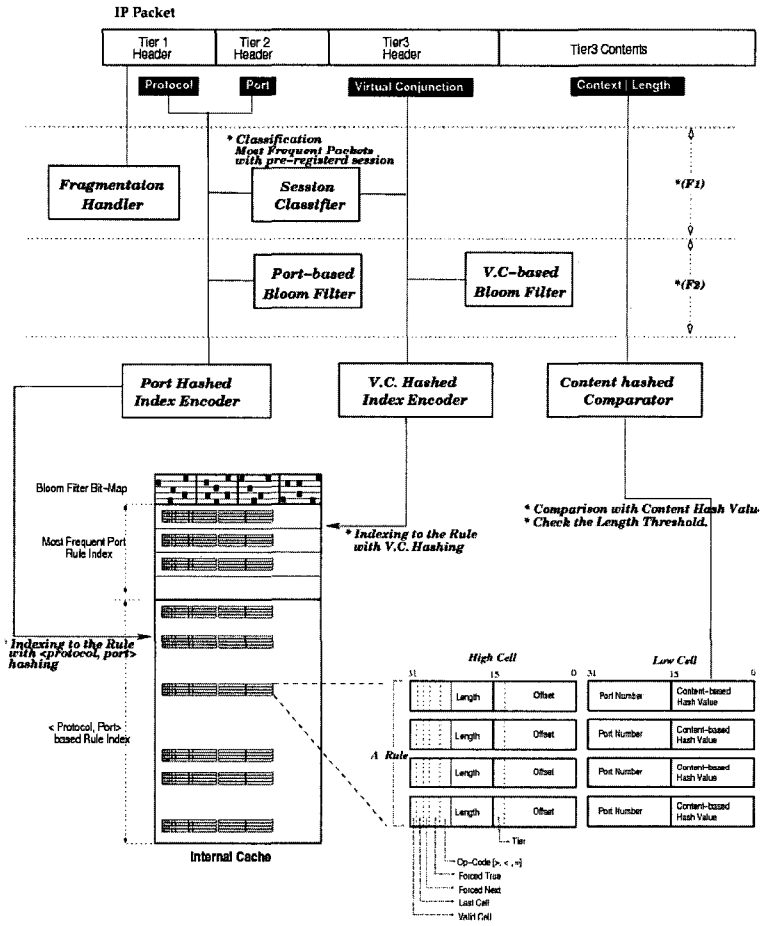


그림 8. Bloom Filter 기반 웜 필터 기능 블럭도
Fig. 8. Functional diagram of worm filter architecture using bloom filter.

서, 대략 2% F.P의 값으로 네트워크 프로세서 내부 자원의 제약 사항을 고려한 최적의 값으로 선택되었다. 그림 6은 <프로토콜, 포트> 기반 Bloom Filter의 각 파라메터 별 F.P를 보여준다. 이 것 역시 k 값은 3, m 값은 128로 사용하는 것이 대략 5% 이내의 FP 값으로 자원 대비 성능의 적정치로 선택되었다.

Bloom Filter에서 사용하는 해쉬 함수는, IXP 네트워크 프로세서에서 지원하는 하드웨어 해쉬 유닛이 있지만 Bloom Filter의 다양한 해쉬 함수와 구현상 호환되기가 어려웠기 때문에, CRC16 기반의 해쉬 함수를 사용하였다. 16bit CRC (Cyclic Redundant Checksum) 알고리즘[8]은 패턴 인식을 위한 해쉬 함수의 방법으로 제안되어 왔다.

그림 8은 Bloom Filter를 활용한 네트워크 프로세서 기반 고속 웜 필터의 기능도를 나타낸다. VC(Virtual Conjunction)기반, <프로토콜, 포트> 기반 Bloom Filter가 네트워크 프로세서의 레지스터 비트맵으로 표현되어 웜이 아닌 트래픽의 95% 이상을 단 한 번의 메모리의 엑세스없이 마이크로 엔진 내부에서 처리될 수 있는 구조를 나타낸다. 두 개의 Bloom Filter를 통해서 웜 멤버쉽 점검에서 확인된 웜 의심 트래픽만이 캐쉬에 저장된 시그너처 Cell과 비교를 시작한다. 앞 절에서 설명한 버퍼 오버플로우 웜 시그너처를 표현한 고정 크기의 Cell 구조를 Fetch하여 지정된 오퍼레이션을 수행한다.

IV. 벤치 마킹

본 절에서는 두 가지 Intel IXP 네트워크 프로세서에서 구현된 버퍼 오버플로우 웜 필터에 대한 성능 시험을 기술한다.

1. IXP1200 웜 필터 벤치마킹

성능 시험을 위한 테스트베드는 그림 9와 같이 5개의 트래픽 발생 호스트와 5대의 수신 호스트로 구성되는데, 각 x86 PC는 Linux 2.3x로 운영되며 기가비트 이더넷으로 L2 스위치에 dumbbell 형식으로 연결되어, 두

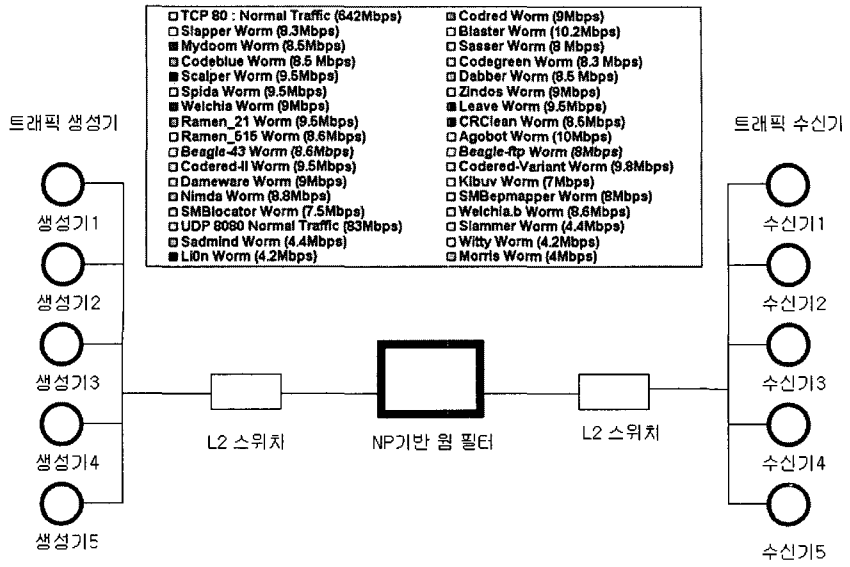


그림 9. 시험 환경 구성 : 버퍼오버플로우 웜과 정상 트래픽
Fig. 9. Benchmark environment configuration : buffer overflow worm and normal trafffic.

표 1. IXP1200 웜 필터의 활성/비활성시의 성능과 손실율
Table 1. Gigabit IXP1200 worm filter performance: filter turn off and on.

	Filter Off			Filter On		
	송신율	수신율	손실율	송신율	수신율	손실율
생성기1	220Mbps	220Mbps	0 %	220Mbps	160Mbps	0 %
생성기2	220Mbps	220Mbps	0 %	220Mbps	160Mbps	0 %
생성기3	220Mbps	220Mbps	0 %	220Mbps	160Mbps	0 %
생성기4	220Mbps	220Mbps	0 %	220Mbps	160Mbps	0 %
생성기5	97Mbps	97Mbps	0 %	60.5Mbps	48.3Mbps	0 %
총계	977Mbps (80,833PPS)	977Mbps	0 %	940Mbps (77,708PPS)	688.3Mbps	
과부하	999.2Mbps (82,581PPS)		2.9 %	999.2Mbps (82,581PPS)		1.8 %

대의 L2 스위치 사이에서 In-Line모드로 동작하는 웜 필터링의 성능을 시험하였다. 트래픽 생성기에서 송신된 CBR, VBR 특성의 웜 및 정상트래픽은 송신측 L2 스위치에서 집선되어 NP기반 웜 필터로 인입되게 하였다. 웜 필터의 성능을 비교하기 위하여 웜 필터링 기능을 동적으로 활성화/비활성화 하여 측정하였다. 그림 11과 같이 사용된 웜 트래픽은 32개의 현재와 과거의 주요 버퍼오버플로우 웜* 을 사용하였고, 웜 트래픽은 TCP 80, UDP 8080의 정상 트래픽(약 80%)과 혼합되어 생성되었으며, 트래픽 수신측에서 트래픽 플로우 별로 일련번호 점검을 수행하여 패킷 손실율을 확인하였다.

표 1은 전체 977Mbps 속도로 웜과 정상 트래픽을 인가한 상태에서 Intel IXP1200 웜 필터 기능을 활성/비활성 하였을 경우의 성능과 손실율을 나타낸다. 4개의 생성기에서는 TCP기반의 웜과 정상트래픽을, 나머지 한 개의 생성기 5에서는 UDP기반의 웜과 정상트래픽을 사용하였다. 이때 MTU 크기를 1500 바이트로 생성하였으며 전체 80,733 PPS로 트래픽이 집선되어 인가되고 있다. 표 1의 왼쪽 행과 같이 웜 필터를 비활성화한 경우, 977Mbps 부하에 대해 손실없이 모든 패킷이 수신측에 도착하였고, 999.2 Mbps로 과부하 트래픽을 인가하였을 때는 2.9%의 패킷 손실이 발생하였는데, 이 중 0.2%는 NP에서 발생하였으며 나머지는 L2 스위치에서 집선되는 과정에서 트래픽 폭주 상황으로 발생하였던 것으로 확인되었다. 표1의 오른쪽 행은 웜 필터를 활성화한 경우이다. 4개의 TCP 기반 웜과 정상 트래픽이 이전과 같이 220Mbps 속도로 전달되었고, 반면에 나머지 UDP기반 생성기에서는 97Mbps에서 60.5Mbps로 속도를 낮추어 전송하여야 정상 트래픽에 대하여 무손실로 전달될 수 있었다. 따라서, IXP1200 기반 웜 필터에서는 940Mbps 정도의 부하에서 웜과 정상트래픽을 오류없이 판단하고, 정상트래픽에 대해 손실 없이 전달됨을 확인하였다.

웜 필터의 정상 트래픽 전달에 대한 또 다른 성능 척도로서 패킷지연 시간(Latency)이나 지터(jitter)는 VoIP의 엔드간 QoS나 다른 QoS-sensitive 응용의 서비스 품질에 큰 영향을 미칠 수 있다. 그림 10은 IXP1200 웜 필터의 평균 패킷 지연시간과 표준 편차를 보여준다. 950Mbps이하의 트래픽 부하상태에서는 평균

* Blaster, Code-RedII, Dabber, Sasser, Slammer, Slapper, Welchia, Witty, Li0n, Sadmind, Mydoom, ...

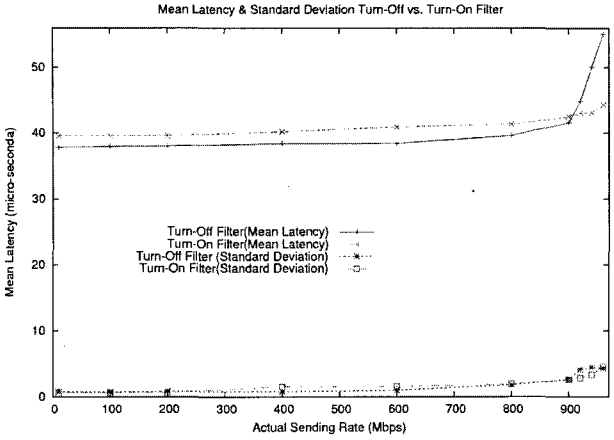


그림 10. 평균 패킷 지연 시간과 표준편차
Fig. 10. Average packet latency and standard deviation.

지연시간이 $43usec$ 이하로 유지됨으로써, 다중 노드 경로에서 엔드 간 서비스 품질에 영향이 없음을 보여주고 있다.

2. IXP2800 웜 필터 벤치마킹

그림 11은 멀티기가 환경의 IXP2801 웜 필터 테스트 베드를 보여준다. 이 웜 필터의 4개 기가비트이더넷 포트가 IXIA 트래픽 계측기에 직접 연결되었고, 각 포트는 양방향 각 1 Gbps의 성능을 가진다. 본 테스트는 IXIA의 4개의 트래픽 생성기를 앞 절에서 사용된 트래픽 생성 비율과 동일하게 구성하였고, 웜과 비정상 트픽을 생성하여 수신측에서 손실율 확인이 동시에 가능하다.

표 2의 왼쪽 행과 같이 IXP2801 웜 필터를 비활성화하였을 경우, 각 기가비트 이더넷 포트의 95% (972.4 Mbps, 80,067 PPS)로 운용하여 전체 3.9 Gbps (320,264PPS) 트래픽이 손실없이 전달됨을 확인하였다. 표의 오른쪽행에서 보듯이, 웜 필터를 활성화하였을 경우 수신측에서는 정상트래픽 (2,920Mbps)만이 손실없이 수신되고, 웜 트래픽은 모두 필터링됨을 확인하였다.

IXP2801웜 필터의 경우에는 RFC2544[18] 기반의 패킷

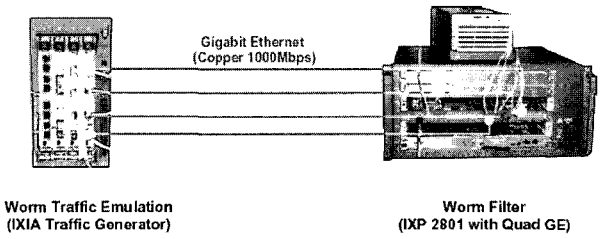


그림 11. IXP 2800 웜 필터 테스트 베드
Fig. 11. IXP 2800 Worm filter testbed.

표 2. IXP2801 웜 필터의 활성/비활성시의 성능과 손실율
Table 2. Gigabit IXP2801 worm filter performance: filter turn off and on.

	Filter Off			Filter On		
	송신율	수신율	손실율	송신율	수신율	손실율
생성기1	972.4Mbps	972.4Mbps	0 %	972.4Mbps	730Mbps	0 %
생성기2	972.4Mbps	972.4Mbps	0 %	972.4Mbps	730Mbps	0 %
생성기3	972.4Mbps	972.4Mbps	0 %	972.4Mbps	730Mbps	0 %
생성기4	972.4Mbps	972.4Mbps	0 %	972.4Mbps	730Mbps	0 %
총계	3,889Mbps (320,264PPS)	3,889Mbps	0 %	3,889Mbps (320,264PPS)	2,920Mbps	

표 3. IXP2801 웜 필터의 패킷 크기별 평균 지연시간
Table 3. Average latency under varing packet size on the IXP2801 worm filter.

패킷 크기(Bytes)	손실율 (%)	평균 지연 시간(usec)
64	0.0000	84.660
256	0.0000	89.840
1024	0.0000	103.220
1518	0.0000	112.480

지연 시험을 IXIA IxScriptMate*를 통해 결과를 얻었으며 그 결과는 표 3을 통해 나타내었다. 이번 지연 측정방법은 패킷의 SoP(Start of Packet)에서 시간 측정을 시작하여 EoP(Eno of Pakcet)에서 측정을 종료하는 방식을 사용하였기 때문에, 패킷의 각 크기별로 평균 지연 시간이 달라진다. 표 3에서 보듯이 64Byte 크기 기준에서 $84,660usec$이 발생하였다. 그림 12의 IXP1200 웜 필터의 평균 지연 시간보다 길어진 것은 지연 시간 측정 방법이 달랐고, Scratch 메모리를 캐쉬로 운용하지 않은 결과인 것으로 확인되었다.

V. 결 론

본 논문에서는 버퍼 오버플로우 웜의 특성을 반영하여 고속 프로세싱이 가능한 웜 시그너처 표현방법을 제시하였고, 이를 기반으로 고속 네트워크 장치 솔루션으로 널리 사용되는 네트워크 프로세서 환경에서 웜 필터 응용으로 활용하기 위해 마이크로 코드로 구현하였다. 구현 방법론으로 최근 네트워크 응용으로 널리 활용되는 Bloom Filter를 사용하여 그 성능을 향상시켰으며, 두 가지의 네트워크 프로세서 환경에서 그 성능을 벤치마킹하였다. 본 논문에서 제안된 시그너처 표현 구조와 처리 알고리즘은 버퍼 오버플로우 기법 웜에 최적화된 구조로서, 시그너처 급증 문제를 해결할 수 있고, 변형

* IxScriptMate 는 IXIA 계측기에서 제공하는 RFC2544 기반 자동 벤치마킹 테스트 툴킷임.

된 웜에 대응이 가능하며, 네트워크 프로세서에서 성능상 문제점으로 지적된 패이로드 패턴 매칭으로 인한 처리 속도 저하 문제를 피할 수 있다. 두 가지 네트워크 프로세서 플랫폼에서 마이크로 코드 형태로 구현된 웜 필터는 그 사이즈가 700 인스트럭션 이하로 구현이 가능하였으며, 다양한 네트워크 프로세서에서 사용 가능한 웜 필터의 한 가지 솔루션으로 활용이 가능한 장점을 가지고 있다. 본 논문에서 사용된 고속 웜 필터는 향후 개발될 자동 웜 시그너처 생성 방법론과 연동되어 구성될 수 있도록 최적화 할 계획이다.

참 고 문 헌

[1] J. W. Lockwood, "Evolvable Internet Hardware Platforms", Evolvable Hardware Workshop, Long Beach, CA, USA, July 12-14, 2001, pp. 271-279.

[2] B. Bloom. Space/time trade-offs in hash coding with allowable errors, *Communications of the ACM 13(7):422-426,* July 1970.

[3] Andrei Broder, Michael Mitzenmacherz, "Network Applications of Bloom Filter : Survey", In 40th Conference on Communication, Control, and Computing, 2002.

[4] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281{293, 2000.

[5] S. C. Rhea and J. Kubiatowicz. In Proc. of INFOCOM-02, June 2002.

[6] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-Based IP traceback. *In Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01), volume 31:4 of Computer Communication Review,* August 2001.

[7] S. Dharmapurikar and P. Krishnamurthy and T. Sproull and J. Lockwood, Deep packet inspection using parallel Bloom filters, in Hot Interconnects, (Stanford, CA), pp. 44--51, Aug. 2003.

[8] International Organization for Standardization *Information Processing Systems. Data Communication High-Level Data Link Control Procedure. Frame Structure.* ISO 3309, Oct, 1984.

[9] T. Toth and C. Krugel, Accurate buffer overflow detection via abstract payload execution. In RAID, pages 274-291, 2002.

[10] R. Chinchani and E. van den Berg. A fast static analysis approach to detect exploit code inside network flows. In RAID 2005.

[11] O. Kolesnikov and W. Lee. Advanced polymorphic worms : Evading IDS by blending in with normal traffic. Technical report, Georgia Tech, 2004.

[12] M. Roesch. SNORT-lightweight intrusion detection for networks. In *Proceedings of the 13th Systems Administration Conference, 1999.*

[13] J. Coit, S. Staniford, and J.McAlerney. Towards faster string matching for intrusion detection or exceeding the speed of snort. In *Proceedings of DISCEX II, June 2001.*

[13] R. Sidhu and V. K. Prasanna. Fast Regular Expression Matching using FPGAs. In *IEEE Symposium on Field-Programmable Custom ComputingMachines (FCCM), Rohnert Park, CA, USA, Apr. 2001.*

[14] R. Fanklin, D. Caraver, and B. Hutchings. Assisting network intrusion detection with reconfigurable hardware. In *Proceedings from Field Programmable Custom Computing* Machines, 2002.

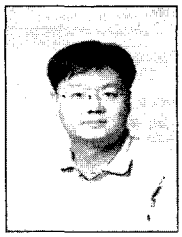
[15] J. Gustafson. Re-evaluating Amdahl's law. Communications of the ACM, 31(5), 532--533, May 1988.

[16] V. Paxson, Bro: A System for Detecting Network Intruders in Real-Time, Computer Networks, 31(23-24), pp. 2435-2463, 14 Dec. 1999.

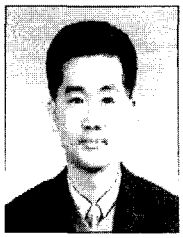
[17] E. Chien, and P. Szor, "Blended Attacks Exploits, Vulnerabilities and. Buffer-Overflow Techniques in Computer Viruses," In Proc. of Virus. Bulletin Conf, 2002.

[18] S. Bradner, J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", IETF, RFC2544, 1999.

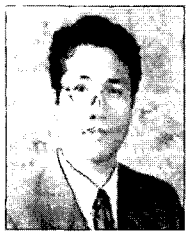
저 자 소 개



김 익 균(정회원)
1994년 경북대학교 컴퓨터공학과 학사 졸업.
1996년 경북대학교 컴퓨터공학과 석사 졸업.
1996년~1999년 한국전자통신 연구원
2000년~2001년 (주) 팍스콤 선임연구원
2004년~2005년 Purdue University 객원연구원
2001년~현재 한국 전자통신 연구원 선임연구원
<주관심분야 : 네트워크보안, 컴퓨터네트워크 >



장 종 수(정회원)
1984년 경북대학교 전자공학과 학사 졸업.
1986년 경북대학교 전자공학과 석사 졸업.
2000년 충북대학교 박사졸업
1989년~현재 한국전자통신연구원 네트워크보안 그룹 그룹장/책임연구원
<주관심분야 : 네트워크보안, 정책기반 보안관리 기술>



한 기 준(정회원)
1979년 서울대학교 전기공학과 학사졸업.
1981년 한국과기원전기공학과 석사졸업
1985년 University fo Arizona 전기 및 전산공학과 석사졸업
1987년 University of Arizona 전기 및 전산 공학과 박사졸업
1981년~1984년 국방과학연구소 연구원
1988년~현재 경북대학교 컴퓨터공학과 정교수
<주관심분야 : 컴퓨터네트워크, 데이타통신>

오 진 태(정회원)
1990년 경북대학교 전자공학과 학사 졸업.
1992년 경북대학교 전자공학과 석사 졸업.
1992년~1998년 한국전자통신 연구원 선임연구원
2003년~현재 한국전자통신연구원 네트워크보안 연구팀 팀장/선임 연구원
<주관심분야 : 네트워크보안, 비정상행위 탐지기술>

손 승 원(정회원)
1984년 경북대학교 전자공학과 학사 졸업.
1994년 연세대학교 전자공학과 석사 졸업.
1999년 충북대학교 박사졸업
1983년~1986년 삼성전자(주) 연구원
1986년~1991년 LG전자(주) 중앙연구소 팀장
1991년~현재 한국전자통신연구원 정보보호 연구단 단장/책임연구원
<주관심분야 : 네트워크 정보보호, RFID/USN정보보호, 유비쿼터스 정보보호, 정보보호 정책>