情報保護學會論文誌
第13券 第6號, 2003. 12

리눅스 Netfilter 시스템과 CBQ 라우팅 기능을 이용한 비정상 트래픽 제어 프레임워크 설계 및 구현

조 은 경*†, 고 광 선*, 이 태 근*, 강 용 혁**, 엄 영 익*‡

Design and Implementation of Anomaly Traffic Control Framework based on Linux Netfilter System and CBQ Routing Mechanisms

Eun-kyung Cho*†, Kwang-sun Ko*, Taekeun Lee*,
Yong-hyeog Kang**, Young Ik Eom*‡

요 약

최근 바이러스가 날로 지능화되고 있고 해킹수법이 교묘해지면서 이에 대응하는 보안기술 또한 발전을 거듭하고 있다. IP 주소 등을 통해 네트워크를 관리하는 방화벽과 방화벽을 뚫고 침입한 해커를 탐지해 알려주는 침입탐지시스템(IDS)에 이어 최근에는 침입을 사전에 차단한다는 측면에서 한 단계 진보한 IDS라고 볼 수 있는 침입방지시스템(IPS)이 보안기술의 새로운 패러다임으로 인식되고 있다. 그러나 현재 대부분의 침입방지시스템은 정상 트래픽과 공격 트래픽을 실시간으로 오류없이 구별할 수 있는 정확성과 사후공격패턴분석 능력 등을 보장하지 못하고 기존의 침입탐지시스템 위에 단순히 패킷 차단 기능을 추가한 과도기적 형태를 취하고 있다. 이에 본 논문에서는 침입방지시스템의 패킷 분석 능력과 공격에 대한 실시간 대응성을 높이기 위하여 netfilter 시스템을 기반으로 커널 레벨에서 동작하는 침입 탐지 프레임워크와, iptables를 이용한 패킷 필터링 기술에 CBQ 기반의 QoS 메커니즘을 적용한 비정상 트래픽 제어 기술을 제시한다. 이는 분석된 트래픽의 침입 유형에 따라 패킷의 대역폭 및 속도를 단계적으로 할당할 수 있도록 하여 보다 정확하고 능동적인 네트워크 기반의 침입 대응 기술을 구현할 수 있도록 한다.

ABSTRACT

Recently viruses and various hacking tools that threat hosts on a network becomes more intelligent and cleverer, and so the various security mechanisms against them have been developed during last decades. To detect these network attacks, many NIPSs(Network-based Intrusion Prevention Systems) that are more functional than traditional NIDSs are developed by several companies and organizations. But, many previous NIPSs are known to have some weakness in protecting important hosts from network attacks because of its incorrectness and post-management aspects. The aspect of incorrectness means that many NIPSs incorrectly discriminate between normal and attack network traffic in real time. The aspect of post-management means that they generally respond to attacks after the intrusions are already performed to a large extent. Therefore, to detect network attacks in realtime and to increase the capability of analyzing packets, faster and more active responding capabilities are required for NIPS frameworks. In this paper, we propose a framework for real-time intrusion prevention. This framework consists of packet filtering component that works on netfilter in Linux kernel and traffic control component that have a capability of step-by-step control over abnormal network traffic with the CBQ mechanism.

keyword : *IPS(Intrusion Prevention System), Netfilter system, CBQ(Class Based Queue)*

* 성균관대학교 정보통신공학부 분산컴퓨팅연구실({iuno, rilla91, tedlee, yhkang1, yieom}@ece.skku..ac.kr)
** 극동대학교 경영학부 전자상거래학과(yhkang@kdu.ac.kr)
† 주저자, ‡ 교신저자, 논문접수일 : 2003년 8월 14일, 심사완료일 : 2003년 11월 7일

Ⅰ. 서 론

최근 기업, 정부기관, 교육기관의 네트워크는 폭발적으로 성장하고 있으며, 인터넷을 통한 상호간 연결도 크게 증가하고 있다. 그러나 인터넷의 발전과 함께 네트워크 상에 존재하는 호스트들의 보안 취약점을 자동으로 검색해주는 등 다양한 기능을 지닌 침입 도구 및 기법들이 공개되어, 네트워크의 성능저하 및 시스템 침입과 같이 악의적인 목표로 네트워크에 유입되는 비정상 트래픽의 양도 현저하게 증가하고 있다.[1]

네트워크에 의존적인 현재의 컴퓨팅 환경에서 이는 침입탐지시스템(IDS)과 같은 보안기술의 발전을 가져왔다. 하지만 인터넷 대란과 같은 대규모 해킹사고를 겪으면서 기존의 방어 위주의 침입탐지시스템에 대한 한계가 지적되기 시작했는데, 이는 침입탐지시스템들의 대부분이 감사 자료를 이용한 호스트 기반의 시스템들이어서 다양한 대규모 네트워크 공격을 모두 탐지하지 못할 뿐 아니라 침입 발생 후 탐지 및 복구에 중점을 둠으로써 실시간으로 발생되는 공격에 대한 적절한 대응 수단을 제공할 수 없기 때문이다. 대다수의 네트워크 기반 침입탐지시스템들 역시 tcpdump 및 libpcap과 같이 응용 레벨에서 동작하는 프로그램을 이용한 패킷 미러링을 통해 침입을 탐지하므로 공격 패킷에 대한 통제력이 떨어져 실시간 탐지 능력이 저하된다는 취약점을 지니고 있다.[2]

이러한 침입탐지시스템의 제한된 탐지 능력과, 탐지와 차단을 실시간으로 처리할 수 없다는 문제점을 해결하기 위한 대안으로 최근 침입을 능동적으로 사전에 차단하는 침입방지시스템(IPS)이 각광받고 있으나 현재까지 개발된 침입방지시스템의 대부분은 기존의 침입탐지시스템 위에 단순히 패킷 차단기능을 추가한 과도기적 형태를 취하고 있다. 이에 본 논문에서는 실시간 패킷 분석 능력 및 유연하고 세밀한 패킷 제어 정책을 제공하는 네트워크 기반의 침입방지시스템을 구현하기 위하여 netfilter 시스템을 기반으로 커널 레벨에서 동작하는 침입 탐지 프레임워크와, iptables를 이용한 패킷 필터링 기술에 CBQ 기반의 QoS 메커니즘을 적용한 비정상 트래픽 제어 기술을 제시한다.

CBQ와 같은 우선순위 기반의 QoS 메커니즘을 보안기술에 적용함으로써 정밀하고 효율적인 패킷 제어 정책이 가능해지며 netfilter를 이용한 커널 레벨의 침입탐지 프레임워크는 침입에 의한 실시간 대응성을 높여준다. 이는 비정상 트래픽으로 간주된 패킷들을 차단하기 이전 분석된 침입 유형에 따라 단계적으로 제한된 대역폭과 속도를 할당하는 과정을 거치도록 하여 탐지오류에 의해 발생할 수 있는 패킷 손실을 줄일 수 있을 뿐 아니라 침입 패킷이 라우터를 거쳐 네트워크로 나가기 전 실시간으로 적절한 제어가 이루어지도록 할 수 있으므로 네트워크에 속한 하위 시스템들에게 큰 피해를 입힐 가능성이 있는 공격에 의한 피해를 줄일 수 있다.[3,4]

본 논문의 2장에서는 기존의 네트워크 기반 침입탐지 및 침입 방지 시스템에 대한 관련 연구를 살펴보고, 3장에서는 본 논문에서 제시할 시스템의 구성 및 알고리즘을 설명하며, 4장에서 실제 구현된 라우팅 기반 패킷 제어 시스템의 프로토타입을 이용하여 비정상 패킷에 대한 침입 탐지 및 능동적인 대처가 가능한지에 대한 실험 및 결과를 분석한다. 마지막으로 5장에서 제시한 기법들에 관한 결론 및 향후 연구로 마무리 짓고자 한다.

Ⅱ. 관련 연구

본 장에서는 네트워크 기반에서 능동적으로 침입을 탐지하고 방지하는 시스템을 개발하는데 필요한 기존 연구들을 살펴보도록 한다.

2.1 침입탐지시스템

침입은 시스템 자원에 대한 무결성, 기밀성 또는 가용성을 침해하는 행위를 의미하며 현재 보안제품의 주류를 이루고 있는 침입탐지시스템은 이러한 비인가된 사용자로부터의 침입을 탐지하여 시스템 자원을 보호하는 역할을 한다.

2.1.1 침입탐지시스템의 분류

침입탐지시스템은 크게 네트워크 기반 침입 탐지와 호스트 기반 침입 탐지 그리고 이 두 기법이 모두 적용된 혼합형 침입 탐지로 나누어진다. 네트워크 기반 침입 탐지는 네트워크를 통해 전송되는 트래픽을 검사하여 침입을 탐지하는 반면 호스트 기반 침입 탐지는 로컬 호스트에서 사용자의 행위나 프로세스들을 검사하여 침입을 탐지한다.[5]

2.1.2 침입탐지 기법의 분류

침입탐지 기법에는 두 개의 상보적인 흐름이 존재하는데, 이는 각각 오용 탐지 기법(misuse detection mo-

del)과 비정상 행위 탐지 기법(anomaly detection model)으로 알려져 있다.[6~8]

오용 탐지 기법은 지식 기반 침입 탐지 기법으로, 알려진 침입 행위에 관한 축적된 지식을 이용하여 정해진 모델과 일치하는 경우를 침입으로 간주한다. 이를 구현하기 위하여 전문가 시스템(expert system), 시그너처 분석(signature analysis), 페트리넷(petri-net), 상태전이 분석(state transition analysis), 신경망(neural network), 유전 알고리즘(genetic algorithm)과 같은 기법들이 존재한다. 이러한 기법은 알려진 침입 패턴에 대해서는 비교적 높은 정확성으로 침입을 탐지하지만 새로운 패턴의 공격들은 탐지하기 힘들다는 단점이 존재한다.

이에 반하여 비정상 행위 탐지 기법은 기준의 정상적인 행위에 대한 참조 모델을 생성한 후 그 행위에서 벗어나는 경우가 있을 경우 이를 침입으로 간주하기 때문에 새로운 공격에 대한 탐지는 가능하게 되나 공격과 정상행위를 구별하기 위한 임계값을 설정하기 힘들다는 어려움이 존재한다. 이를 구현하기 위하여 통계적(statistical) 방법, 전문가 시스템(expert system), 신경망(neural network), 컴퓨터 면역학(computer immunology), 데이터 마이닝(data mining), HMM (Hidden Markov Model) 등의 기법들이 존재한다.

2.2 침입방지시스템

최근 잇따른 대형 인터넷 대란으로 침입방지시스템이 침입탐지시스템을 대체할 차세대 보안기술의 패러다임으로 인식되고 있다. 기존의 침입탐지시스템이 현재보다 더 많은 공격을 정확하게 탐지하는 데 목적이 있다면 침입방지시스템은 공격을 탐지하는 것뿐만 아니라 공격이 일어나는 것을 근본적으로 방어하는 것을 목적으로 한다. 따라서 침입방지시스템은 다양한 보안 기술을 이용해, 실시간으로 침입을 탐지함과 동시에 패킷을 차단하는 등의 능동적인 대응을 한다는 개념에서 수동적인 방어위주의 방화벽이나 침입탐지시스템과 차별된다.

그러나 현재 개발되고 있는 대부분의 국내외 침입방지시스템들은 방화벽과 침입탐지시스템의 기능을 확장하는 과도기적 형태가 주류를 이루고 있다.

2.3 리눅스 기반 침입 탐지 지원 기술

본 논문에서 제시할 비정상 트래픽 제어 시스템은 다음과 같은 기술을 이용하여 커널 모드에서 동작하도록 구현된다.

2.3.1 커널 모듈 프로그래밍 기술

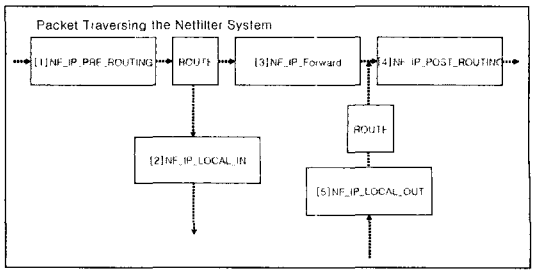
커널 모듈은 리눅스에서 다양한 하드웨어 장치를 효율적으로 지원하기 위한 기술로서, 디바이스 드라이버나 파일 시스템 등을 모듈로 구현하면 시스템이 동작하고 있는 상태에서도 커널의 수정 없이 필요할 때마다 동적으로 추가할 수 있다는 장점이 있다.

따라서 비정상 패킷 제어 시스템을 모듈로 구현하여 커널에 삽입하면시스템의 전체적인 가용성을 증대시킬 수 있을 뿐 아니라 프로그램 실행 속도를 보다 향상시킬 수 있다. 또한 프로세스가 가지는 특징들인 부모와 자식 프로세스 간 메모리 공간 공유, 동일한 파일시스템 정보 공유, 그리고 동일한 시그널 핸들러 공유 등의 기능을 이용하고자 할 경우 커널 쓰레드로 구현하여 모듈에 삽입하면 커널 모드에서 사용자 모드 프로세스의 기능을 이용할 수도 있다. 이는 커널 모드에서 동작하는 프로세스를 구현할 수 있도록 하여 일반 사용자 모드에서 동작하는 프로세스보다 보안에 대한 안정성을 확보할 수 있게 한다.[9~11]

2.3.2 Netfilter 시스템을 이용한 트래픽 감시

Netfilter는 버클리 소켓 인터페이스의 외부에 존재하는 패킷 재조합에 대한 프레임워크로, 이 기능을 이용하면 패킷을 실시간으로 감시하면서 보안기능을 갖는 리눅스 시스템을 운영할 수 있다.[12]

Netfilter 시스템은 크게 세 부분으로 구성되어 있는데, 첫 번째로 각각의 프로토콜은 훅(hook)이라는 것을 정의하며, 이는 프로토콜 스택의 패킷 트래버스에 있는 잘 정의된 포인터를 의미한다. 이러한 포인터를 이용하여 각각의 프로토콜은 패킷과 훅 넘버를 가지고 netfilter 시스템을 호출하게 되는데 이 과정은 [그림 1]과 같다.



〔그림 1〕 Netfilter 시스템의 동작 과정

패킷은 [그림 1]의 좌측으로부터 들어와서 단순한 데이터 체크를 거친 후, netfilter 시스템의 NF_IP_PRE_ROUTING 훅으로 전달된다. 다음으로 패킷은 라우팅 코드로 들어가며, 여기서 패킷이 다른 인터페이스로 향하는지 또는 로컬 프로세스로 향하는지 결정된다. 패킷이 라우팅 될 수 없는 경우, 라우팅 코드는 패킷을 버리기도 한다.

만일 패킷이 로컬 프로세스로 들어 왔다면 netfilter 시스템은 패킷을 프로세스로 전달하기 전에 NF_IP_LOCAL_IN 훅을 다시 한번 호출하게 되며 다른 인터페이스로 전달하고자 할 경우에는, NF_IP_FORWARD 훅을 호출한다. 그 후 패킷은 네트워크로 보내지기 전에 마지막 훅인 NF_IP_POST_ROUTING 훅으로 전달된다. 로컬 생성 패킷에 대해서는 NF_IP_LOCAL_OUT 훅이 호출되며 이 훅이 호출된 후 라우팅이 발생한다.

Netfilter 시스템의 두 번째 부분으로서, 커널 모듈은 위에서 언급한 훅 포인트에서 특정 패킷이 들어오는 것을 기다리도록 함수를 등록할 수 있다. 이때 모듈은 해당하는 훅 포인트에서 자신이 등록할 함수가 가질 우선순위를 명시함으로써 그 후 패킷이 netfilter를 통과할 때, 자유롭게 패킷을 조작할 수 있도록 한다. 즉, 각각의 커널 모듈들이 해당 훅 포인트에 등록한 함수가 우선순위대로 호출되어 패킷을 검사한 후, 무시하거나(NF_DROP), 통과시키도록(NF_ACCEPT) 할 수 있으며, 또는 패킷을 가로채는 작업이나(NF_STOLEN), 사용자 공간에 패킷을 대기시키는 작업 등을 요청할 수 있다(NF_QUEUE).

마지막으로 netfilter 시스템의 세 번째 부분은 큐에 대기된 패킷들을 사용자 공간으로 보내기 위해 모으는 부분으로 이루어져 있다.

2.3.3 Iptables와 CBQ를 이용한 트래픽 제어

Iptables는 리눅스 시스템이 필터링 규칙에 따라 패킷을 차단할 수 있도록 지원하는 기능으로, 이를 이용하면 침입으로 간주된 패킷들이 네트워크로 유입되는 것을 막을 수 있으므로 하위 시스템들을 보호할 수 있다.

이 패킷 선택 시스템은 netfilter 시스템을 기반으로 구성되며 NF_IP_LOCAL_IN과 NF_IP_LOCAL_OUT, NF_IP_FORWARD 시점에서 패킷을 훅한 후 시스템 내에 정의되어 있는 필터링 테이블과 비교하여 패킷의 폐기 여부를 결정한다. 따라서 커널 모듈은 새로운 테이블을 등록한 후, 임의의 패킷이 주어진 테이블을 통과하도록 요청할 수 있다.

CBQ는 우선순위 큐잉 방식의 변형으로써 하나의 출력 큐 대신에 여러 개의 출력 큐를 클래스 별로 두어서 우선순위를 정하고 각 큐 별로 서비스 되는 트래픽의 양을 조절할 수 있는 방식이다. 이렇게 함으로써 어느 특정 클래스의 트래픽이 전 시스템 자원을 모두 독점하는 것을 방지한다. 즉, CBQ는 클래스 별로 자원이 완전 고갈되는 것을 막으면서도 각 클래스에 적절한 서비스를 제공할 수 있다.

따라서 CBQ는 라우터 역할을 하는 리눅스 시스템이 서비스(포트 번호) 또는 발신지 IP 주소에 따라 네트워크 대역폭을 할당할 수 있도록 한다. 조정된 대역폭에 따른 패킷들의 통계정보를 확인함으로써 재설정이 가능하고, 시스템을 운영하면서 동적으로 대역폭을 할당할 수 있다.

본 논문에서는 비정상 트래픽이 탐지될 때마다 CBQ를 이용해 분석된 침입 유형에 맞게 대역폭을 단계적으로 재설정할 수 있는 기능을 구현하여 침입 탐지 오판율을 줄이면서 네트워크의 효율성과 안정성을 증가시킬 수 있는 방법을 제시한다.

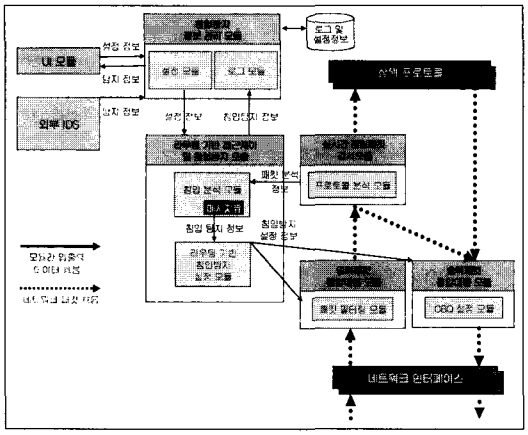
Ⅲ. 리눅스 라우팅 시스템 기반 비정상 트래픽 제어 기법

본 절에서는 리눅스 라우팅 시스템을 기반으로 커널 모드에서 동작하는 비정상 트래픽 제어 시스템을 구현하여 침입에 효율적으로 대응할 수 있는 방법을 제시한다.

3.1. 모듈 별 기본 동작 및 알고리즘

라우팅 시스템 기반 비정상 트래픽 제어 시스템의 전체적인 모듈 구성을 [그림 2]에서 보인다.

실시간 트래픽 감시를 통하여 수집된 비정상 트래픽들의 정보는 침입 경고 후, CBQ를 이용해 출력 대역폭이 제한되는 형식으로 해당 네트워크의 라우팅 정책에 반영된다. 추후 이미 비정상 트래픽으로 경고되어 제한된 대역폭을 할당 받은 호스트가 재차 침입으로 의심되는 패킷들을 전송할 경우 이는 동적으로 동작하는 입력 패킷 침입 대응 모듈을 통하여 라우팅 시스템의 상위 프로토콜 및 네트워크로의 진입이 차단되는 과정을 거치게 된다. 패킷의 흐름 순서에 따른 세부 모듈의 동작 과정은 다음과 같다.



〔그림 2〕 전체 모듈 구성도

3.1.1 입력 패킷 침입 대응 모듈

입력 패킷 침입 대응 모듈에 속한 PFM(Packet Filtering Module)은 동적으로 갱신되는 필터링 ruleset을 이용하여 해당되는 트래픽을 모두 차단한다. 필터링 rule은 IAM(Intrusion Analyzing Module)이나 외부 IDS에 의해 이미 비정상 트래픽으로 분류되어 CBQ를 이용해 대역폭이 제한된 호스트가 재차 침입으로 의심되는 패킷들을 보낼 경우, 이 패킷들을 차단하기 위해 라우팅 기반 IPCM(Intrusion Prevention Configuration Module)이 iptables를 이용하여 동적으로 생성한다. 또한 관리자가 자신의 네트워크로 들어오는 트래픽을 원하는 형태에 따라 제한해야 할 필요성이 있는 경우 미리 정의된 형식으로 필터링 파일에 생성한 rule이 존재한다.

Iptables는 커널 쓰레드를 이용하여 호출되며, 이를 이용하여 설정 또는 제거할 수 있는 필터링 rule의 형식은 [그림 3]을 따른다.

여기서 -I 옵션은 SRC_ADDR를 IP 주소로 갖는 침입 호스트로부터 라우터의 네트워크 인터페이스로 들어온 패킷을 차단함을 의미한다. 추후 -D 옵션을 사용하여 iptables를 호출하면 기존의 필터링 rule이 제거되어 SRC_ADDR로부터 전송된 패킷들이 상위 레벨로의 전달되는 것을 다시 허가할 수 있다. Iptables를 이용하여 등록된 필터링 rule은 netfilter 시스템의 NF_IP_LOCAL_IN, NF_IP_LOCAL_OUT, NF_IP_FORWARD 시점에서 패킷을 차단하는데 사용된다. 이

```
/sbin/iptables -I INPUT -s SRC_ADDR -j DROP
/sbin/iptables -D INPUT -s SRC_ADDR -j DROP
```

〔그림 3〕 iptables를 이용하여 작성된 rule의 형식

〔표 1-1〕 필터링 파일에 정의된 rule의 형식

Filtering ruleset	Meaning
S : P_s -> D : P_d	If the packet is transmitted from port P_s in host S to port P_d in host D, then the packet should be dropped
S : any_port -> any_host : any_port	If the packet is transmitted from host S, then the packet should be dropped
any_host : any_port -> D : any_port	If the packet is transmitted from host D then the packet should be dropped
S : any_port -> D : any_port	If the packet is transmitted from host S to host D, then the packet should be dropped
any_host : P_s -> any_host : any_port	If the source port number of packet is P_s, then the packet should be dropped
any_host : any_port -> any_host : P_d	If the destination port number of packet is P_d, then the packet should be dropped

〔표 1-2〕 필터링 rule의 구성 요소

Component	Meaning
S	source ip address
D	destination ip address
P_s	source port number
P_d	destination port number

는 비정상 트래픽이 라우팅 시스템의 상위 프로토콜로 전송되거나 네트워크 인터페이스를 통하여 시스템 밖으로 나가는 것을 방지한다.

필터링 파일에 설정할 수 있는 rule은 차단하려는 트래픽 형태에 따라 [표 1]에서 보이는 바와 같은 형식을 따른다.

[표 1]에 명시된 형태의 필터링 rule을 사용하면 상황에 맞게 트래픽을 차단함으로써 유연한 패킷 필터링 정책 운영이 가능하다.

IAM에 의해 침입으로 탐지 되지 않았더라도 시스템들에 피해를 줄 수 있다고 판단되는 트래픽을 제한하려는 경우 필터링 파일에 원하는 형태의 rule을 등록하여 해당 트래픽이 라우팅 시스템 내에서 차단되도록 할 수 있다. 또한 iptables를 이용하여 침입으로 간주된 호스트의 모든 패킷을 차단하는 것을 원치 않을 경우, -D iptables 옵션을 사용하여 해당 필터링 rule을 제거한 후 침입의 유형에 맞게 특정 목적 호스트로 가는 트래픽만 차단하거나, 특정 목적

호스트의 특정 서비스를 이용하는 트래픽만을 차단할 수 있도록 필터링 rule을 파일에 등록함으로써 손실될 수 있는 정상 패킷의 비율을 줄일 수 있다.

이는 라우팅 시스템의 네트워크 인터페이스로부터 유입되는 모든 패킷들이 netfilter 시스템을 정상적으로 통과한 후에만 상위 레벨의 프로토콜로 전송(NF_ACCEPT)되거나 라우팅 되도록 훅을 등록할 때, 필터링 파일에 포함된 rule을 만족하는 패킷은 다음 단계로의 진입이 차단(NF_DROP)되도록 하는 함수를 추가함으로써 이루어진다.

3.1.2 실시간 침입 패킷 감시 모듈

실시간 침입 패킷 감시 모듈에서 동작하는 PAM (Protocol Analyzing Module)의 구조 및 작업 수행 과정을 [그림 4] 및 [그림 5]에서 보인다.

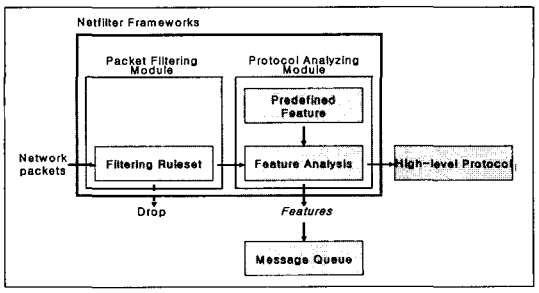
Netfilter 시스템을 통과하는 패킷이 PFM에 등록된 필터링 ruleset을 정상적으로 통과했을 경우 이는 상위 레벨의 프로토콜로 전달(NF_ACCEPT)되기 전에 PAM을 다시 한번 통과하여 추가의 패킷 처리 작업을 거치게 된다.

즉, PAM은 PFM과 마찬가지로 netfilter 시스템이 패킷들을 처리하는 과정에 추가되어 PFM 통과 후 정상적으로 상위 레벨의 프로토콜로 전송되는 모든 패킷의 헤더 정보를 분석하여 침입을 탐지하는데 필요한 메시지(M) 형식으로 작성한다.

메시지 형식은 프로토콜 유형에 따라 다음과 같이 미리 정의되어 있다. 각 구성 요소들의 의미는 [표 2]에서 보인다.

$$M_{tcp} = \{P, F, S, D, P_s, P_d\}$$
$$M_{udp} = \{P, S, D, P_s, P_d\}$$
$$M_{icmp} = \{P, S, D, T, C\}$$

PAM은 패킷을 상위 프로토콜로 전송하는 과정에



[그림 4] 프로토콜 분석 모듈의 구조

```
OP1: When a network packet P arrives in sk_buff:
/* P is hooked by netfilter framework before it is
   sent to high-level protocol */
P is sent to PFM by netfilter system; /* ref: OP1-1 */
if (P successfully passes PFM) {
   sends P to PAM;                    /* ref: OP1-2 */
   if (P successfully passes PAM)
      sends P to high-level protocol;
}
else
   P is dropped;

OP1-1: When PFM receives a network packet P:
/* P is checked whether it's pattern is matched the
   filtering ruleset */
/* R_ipt means the filtering ruleset which is set up by
   iptables and R_file means the filtering ruleset which
   is set up in filtering file */
if ((the pattern of P) ∈ R_ipt)
   P is automatically dropped at NF_IP_LOCAL_IN,
     NF_IP_LOCAL_OUT, NF_IP_FORWARD hooks by iptables;
elseif ((the pattern of P) ∈ R_file) {
   sends a filtered packet information to filtering
   message queue;
   P is dropped;
}
else
   P is accepted;

OP1-2: When PAM receives packet P that passed PFM:
switch(P->protocol)
  case TCP:
    M_tcp = {P->protocol, P->flag, P->saddr, P->daddr,
          P->sport, P ->dport};
    sends M_tcp to tcp message queue(Q_tcp) in IAM;
    break;
  case UDP:
    M_udp = {P ->protocol, P ->saddr, P ->daddr, p->sport,
          P ->dport};
    sends M_udp to udp message queue(Q_udp) in IAM;
    break;
  case ICMP:
    M_icmp = {P ->protocol, P ->saddr, P ->daddr,
          P ->msg_type, P ->msg_code};
    sends M_icmp to icmp message queue(Q_icmp) in IAM;
    break;
```

[그림 5] Netfilter 시스템에 등록되어 수행되는 PFM과 PAM의 동작 과정

[표 2] 프로토콜 분석 모듈의 메시지 구성 요소

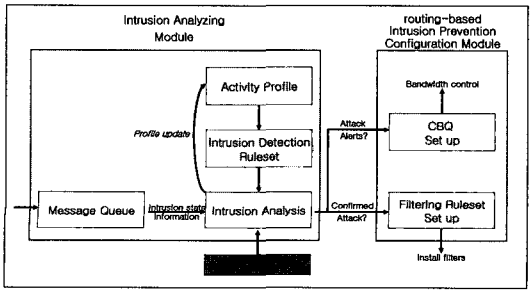
Component	Meaning
P	Protocol
F	{URG, PSH, SYN, FIN, ACK, RST}
S	source ip address of the packet
D	destination ip address of the packet
P_s	source port number of the packet
P_d	destination port number of the packet
T	icmp message type of the packet
C	icmp message code of the packet

추가된 모듈이기 때문에 PAM에서의 작업 처리 시간은 전체 패킷 처리 시간에 영향을 준다. 따라서 패킷 처리의 효율성을 위하여 PAM에서는 침입탐지를 위한 메시지를 생성하여 각 메시지 큐에 전송하는 작업만 하도록 하고 이 메시지를 바탕으로 침입을 탐지하기 위한 상태 정보를 갱신하고 침입을 탐지하는 작업은 PAM과 독립적으로 동작하는 IAM에서 수행하도록 한다.

이와 같이 netfilter 시스템을 이용하면 커널 레벨에서 시스템이 패킷을 처리하는 과정에 직접 참여하여 침입 발생 여부를 모니터링 할 수 있기 때문에 침입에 따른 즉각적인 대응이 가능하다는 장점이 있다. 이는 기존의 tcpdump 또는 libpcap과 같이 응용 레벨에서 돌아가는 프로그램들이 수신된 패킷들의 복사본을 이용하여 침입을 분석하기 때문에 침입이 발생했을 때 실시간 대응이 어렵다는 단점을 극복한다.

3.1.3 라우팅 기반 접근 제어 및 침입 방지 모듈

라우팅 기반 접근 제어 및 침입 방지 모듈에 속한 IAM은 커널 타이머에 등록되어 메시지 큐로부터 주기적으로 헤더 정보를 읽어온 후 activity profile에 저장된 패킷의 침입 상태 정보를 갱신한다. 이 상태 정보와 등록된 ruleset을 바탕으로 침입 여부를 판단하며, 탐지된 침입 정보는 공격에 대한 능동적인 대응을 위하여 IPCM으로 넘겨준다. 그러나 공격 패킷을 제어하는데 사용되는 침입 정보를 snort와 같은 기존의 상용 침입탐지시스템과 연동해서 생성시킬 경우 IAM과 같은 내부의 침입탐지 프레임워크는 생략될 수 있다. 이 경우 외부 침입탐지시스템으로부터 받은 침입탐지 전달 정보는 IAM을 거치지 않고 IPCM으로 직접 전달된다. IAM의 세부적인 동작과정은 [그림 6]에서 보인다.



[그림 6] 규칙 기반 접근 제어 및 침입 방지 모듈의 동작 과정

(1) IAM(Intrusion Analyzing Module)

IAM은 제어시스템 내부에 구현되는 침입탐지 프레임워크이다. 만약 외부 침입탐지시스템과 연동하지 않고 비정상 트래픽 제어 시스템 내부에 침입을 분석하기 위한 모듈을 구현하려면 다음과 같은 구성요소를 이용하여 침입을 탐지하는 IAM에 원하는 형태의 rule을 등록해야 한다.

○ Message Queue

IAM은 PAM에서 보내는 메시지들(M_{tcp}, M_{udp}, M_{icmp})을 저장하기 위하여 별도의 큐들(Q_{tcp}, Q_{udp}, Q_{icmp})을 관리한다. 이 메시지 큐들로부터 패킷의 헤더정보를 읽어온 후 주기적으로 갱신되는 각각의 침입 상태 정보를 activity profile에 반영한다.

○ Activity Profile

전체적인 침입탐지시스템의 상태를 나타낸다. IAM이 커널 타이머에 의해 호출되는 주기가 정의되어 있으며, 이 주기는 시스템의 성능에 따라 IAM이 효율적으로 동작할 수 있는 값의 범위로 설정되어야 한다. 주기마다 메시지 큐들을 분석하여 새롭게 생성되는 침입 상태 정보들을 저장한다. 또한 IAM이 통계적 기법을 사용하여 침입 여부를 판단한다면 각 공격 유형에 해당하는 패킷의 임계값들이 미리 정의되어 있어야 한다.

○ Ruleset

Activity profile에 저장된 침입 상태 정보를 바탕으로 침입 여부를 판단하기 위한 일반적인 추론 메커니즘을 나타낸다. 즉 침입을 탐지하기 위해 관리자가 등록한 모든 종류의 침입탐지 ruleset을 의미한다.

[표 3]은 위 사항들을 적용하여 port scan을 탐지하기 위해 작성된 rule의 예를 보이고 있다. 이 경우 IAM은 탐지를 회피하기 위하여 정해진 시간 동안

[표 3] PROBING 공격을 탐지하기 위한 rule의 예

Component	Meaning
port scan :: op_time = 2(or 8, or 32, or 128, or 512), msg_type = Mtcp, src = A, dest = B, diff_srv = 10 (or 32, or 140, or 600, or 2400),	If, for the past 2(8, 32, 128, 512) seconds, the number of tcp packets (they have the same destination and source ip address, but they have the different port numbers) is at least 10(32, 140, 600, 2400), then this is a port scan(a PROBING attack)

임계값 이하의 공격 패킷으로 꾸준히 침입을 시도하는 느린 공격(slow attack)을 탐지해내기 위하여 정의된 주기(T)동안 뿐 아니라 4T, 16T, 64T, 128T, 256T 동안 발생된 누적 상태정보를 activity profile에 관리한다.[13~15] 따라서 IAM은 주기(T)뿐만 아니라 4T, 16T, 64T, 128T, 256T 시간동안의 누적된 침입 상태값을 미리 정의된 각각의 임계값과 비교하여 port scan을 탐지하는 작업을 수행한다.

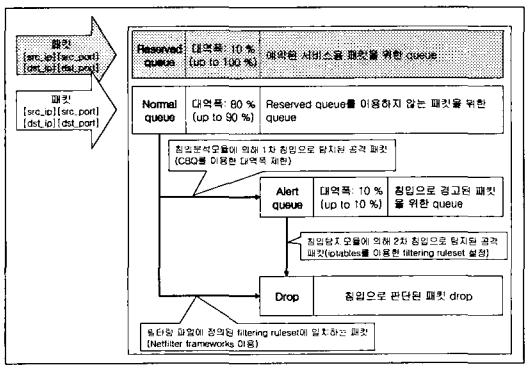
IAM에 의해 침입으로 판단된 패킷의 정보는 IPCM으로 전달되어 침입에 대한 능동적 대응이 이루어지게 된다. 이는 추후 발생할 또 다른 공격에 의한 네트워크 내 시스템 피해를 줄일 수 있게 한다.

(2) IPCM(routing-based Intrusion Prevention Configuration Module)

IPCM은 IAM 이나 외부 침입탐지모듈로부터 전달받은 침입 패킷 정보를 이용하여 공격에 능동적으로 대응한다. CBQ를 이용한 우선순위 기반의 QoS 메커니즘을 IPCM의 패킷 제어 정책에 적용하여 각각의 비정상 트래픽을 침입 유형에 따라 유연하고 세부적으로 관리할 수 있다 . 이를 위해 IPCM에서 이루어지는 트래픽 제어 정책을 [그림 7]에서 보인다.

라우팅 시스템을 경유하여 출력되는 모든 네트워크 트래픽들은 CBQ를 이용하여 구성된 세 종류의 출력 큐에 의해 대역폭이 관리된다. 예약된 서비스(혹은 중요 서비스)를 제공하는데 사용되는 대역폭은 패킷의 손실이 발생하지 않도록 reserved queue를 이용하여 네트워크 혼잡도에 따라 전제 대역폭의 최소 10%에서 최대 100%까지 보장된다.

Reserved queue를 이용하지 않는 일반 패킷들을 위한 대역폭은 normal queue에서 관리한다. 그러나 normal queue에 의해서 정상적으로 관리되던 패킷들



[그림 7] 비정상 트래픽 제어 정책 구조도

이 IAM에 의해 침입으로 간주된 경우, 이는 비정상 패킷에 의하여 네트워크에 발생할 수 있는 피해를 방지하기 위해 alert queue 관리 하에 두어 제한된 대역폭을 할당 받도록 한다.

추후 alert queue에서 대역폭 관리가 이루어지고 있는 침입 호스트들이 재차 침입으로 판단되는 패킷을 전송할 경우 이는 iptables를 이용하여 네트워크로의 유입을 차단한다. [그림 7]에서 보인 출력 큐 이외에도 관리자의 의도에 따라 목적에 맞는 다양한 출력 큐를 구성하여 보다 정밀한 트래픽 제어 정책을 적용할 수 있다.

3.1.4 출력 패킷 대응 모듈

CCM(CBQ Configuration Module)은 IPCM의 비정상 트래픽 제어 정책([그림 7])에 근거한 CBQ 설정을 통해 출력 queue를 구성하며, 이를 이용하여 라우팅 시스템에서 네트워크로 출력되는 패킷들의 대역폭을 관리한다.

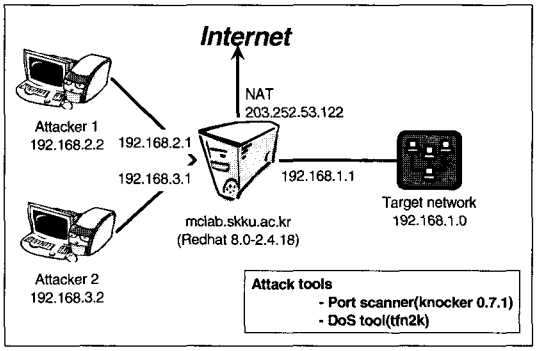
3.1.5 침입 탐지 정보 관리 모듈

CM(Configuration Module)은 사용자의 요청 또는 시스템 초기화 시, 라우팅 기반 접근제어 모듈 및 침입 방지 모듈에게 세부 모듈 별 설정 정보를 전송한다. 또한 외부 침입탐지시스템을 이용하여 공격을 탐지할 경우 탐지된 침입패킷에 대한 정보를 형식에 맞게 변환하여 IPCM에 직접 전송함으로써 내부 침입탐지 프레임워크 없이도 능동적인 침입 대응이 이루어지도록 한다. LM(Log Module)은 외부 침입탐지 시스템 뿐 아니라 라우팅 기반 접근제어 모듈 및 침입 방지 모듈로부터 침입탐지 및 패킷 처리 정보를 전송 받아 로그 정보로 유지한다.

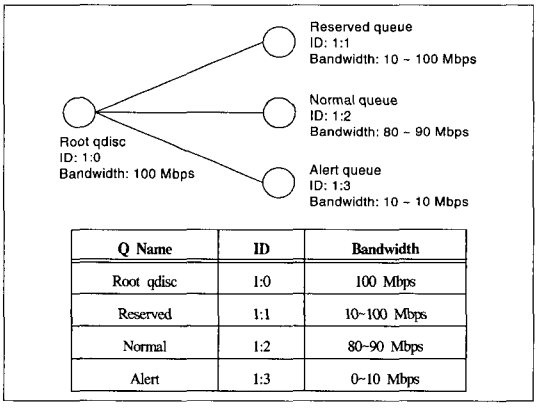
Ⅳ. 구현 및 실험 결과

본 논문에서 제시한 기법을 테스트하기 위하여 리눅스 기반의 라우팅 시스템에서 실시간으로 동작하는 비정상 트래픽 제어 시스템을 구현하여 침입 패킷에 대한 탐지와 능동적 대처가 가능한지 여부를 확인한다. 전체 시스템의 테스트 베드 구성도는 그림 8에서 보인다.

구현된 프로그램은 리눅스 기반 라우터로 동작할 mclab에 탑재되어 target network로 향하는 모든 패킷을 실시간으로 감시한다. 3장에서 제시한 비정상 트래픽 제어 정책에 따라 네트워크로 유입되는 침입



〔그림 8〕 테스트 베드 구성도



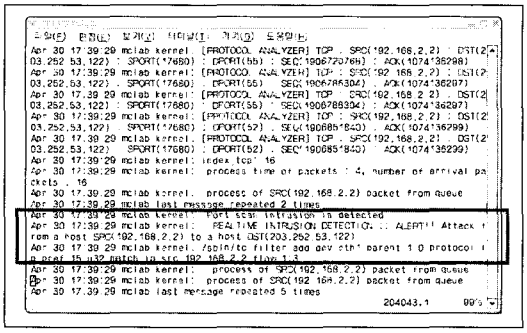
Q Name	ID	Bandwidth
Root qdisc	1:0	100 Mbps
Reserved	1:1	10~100 Mbps
Normal	1:2	80~90 Mbps
Alert	1:3	0~10 Mbps

〔그림 9〕 mclab에 적용된 CBQ 구성도

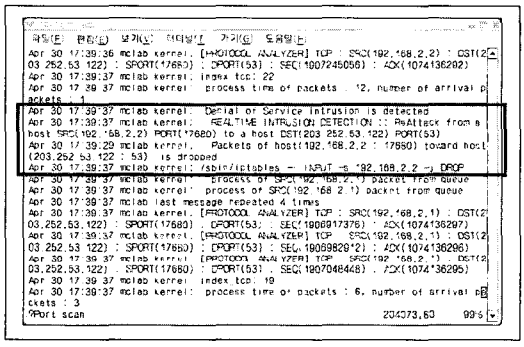
패킷의 대역폭을 제한하기 위하여 [그림 9]와 같이 CBQ를 이용하여 기본적인 세 개의 출력 큐를 구성하였다. 따라서 IAM이나 외부 침입탐지시스템과 같은 침입탐지 프레임워크에 의해 1차 침입으로 경고된 패킷들은 [그림 9]의 설정 내용에 따라 alert queue에 의해 최대 10Mbps의 대역폭을 갖도록 트래픽이 제한된다.

다음은 실험 데이터를 얻기 위해 공격 도구(knocker, tfn2k)를 이용하여 인위적인 침입 샘플 트래픽을 발생시킨 후 시스템에 나타나는 결과를 관찰한 화면을 보인다. 실험 결과는 프로그램 코드 상에 printk 명령어를 삽입한 후 dmesg 명령을 이용하거나 /var/log/messages 파일에 기록된 메시지를 확인하여 시스템이 커널 내부에 출력하는 내용을 관찰하였다. 이를 통하여 세부 모듈간 통신이 정상적으로 이루어지고 있는가를 확인한다.

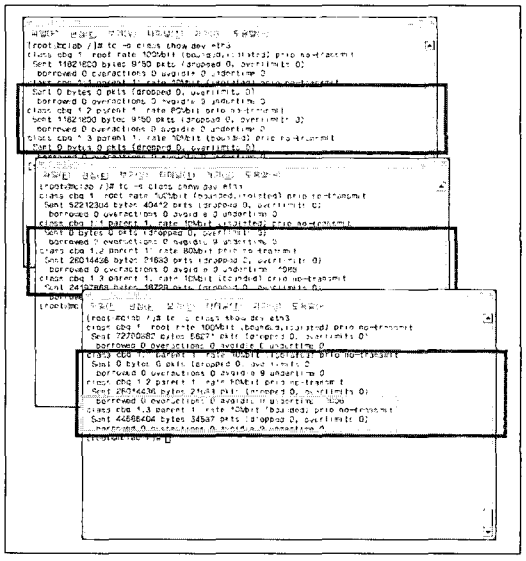
침입 패킷이 발생하였을 경우 mclab의 대응 과정은 [그림 10, 11]과 같다. 침입 경고 후 IPCM에 의해 제한된 트래픽만이 유입되도록 허용된 침입 호스트



〔그림 10〕 CBQ를 이용한 침입 패킷의 대역폭 제한



〔그림 11〕 iptables를 이용한 침입 패킷의 차단



〔그림 12〕 CBQ기반 트래픽 대역폭 할당 모듈에 의한 출력 큐의 변화

(192.168.2.2)가 그 후에도 재차 침입으로 판단되는 비정상 패킷을 네트워크로 전송할 경우 해당 패킷은 iptables에 의해 차단되는 과정을 거치게 된다.

공격에 대한 이러한 단계적 대응은 공격 전후의 상관성을 파악할 수 있도록 하여 [그림 10, 11] 같이 일반적으로 네트워크 PROBING 공격 선행 후 DoS 공격이 이루어지는 경우를 효과적으로 탐지하여 제어할 수 있게 한다. [그림 12]는 누적된 시스템 통계 정보를 확인하여 실제로 침입이 발생되었을 경우 CBQ에 의해서 해당 패킷의 대역폭이 조절됨을 보이고 있다.

침입 패킷이 발생하기 이전에는 모든 트래픽이 normal queue를 통하여 정상적으로 관리되고 있으나, 시스템이 침입 패킷을 탐지한 직후부터는 alert queue를 통해 대역폭이 제한되는 트래픽의 양이 서서히 증가함을 볼 수 있다.

V. 성능 비교

침입으로부터 네트워크 시스템을 보호하기 위하여 개발된 기존의 시스템들과 본 논문에서 제시한 시스템을 상대적으로 비교한 결과를 [표 4]에서 보인다.

[표 4]에서 보듯 침입탐지시스템은 좀 더 많은 공격을 정확하게 탐지하는데 목적을 두고 개발되어 왔기 때문에 다양한 공격에 대하여 높은 침입탐지율을 제공한다. 그러나 탐지한 침입정보를 로그로 남기거나 관리자에게 알리는 것 이외의 어떤 패킷 제어기능도 제공하지 않기 때문에 실시간으로 공격에 대한 적절한 대응방안을 제시하지 못한다는 단점이 존재한다. 이러한 침입탐지시스템의 대안으로 등장한 침입방지시스템은 정확한 침입분석모듈을 바탕으로 사전에 침입을 능동적으로 차단한다는 의미에서 침입에 의한 시스템의 피해를 줄일 수 있도록 한다.

그러나 패킷 차단의 근거가 되는 정확한 침입탐지 모듈을 구현하는 것이 어렵기 때문에 탐지오류에 의한 패킷 손실이 발생할 수 있다. 공격 유형별로 세밀한 트래픽 분류가 이루어진다 하더라도 차단 이외의 다른 제어 기능을 제공하지 않는 것 역시 침입방지 시스템의 아쉬운 점이다.

최근의 인터넷대란을 거치면서 새로운 보안장비로 떠오른 것이 바로 세 번째 비교 시스템인 네트워크 트래픽 관리 시스템이다. 이는 본래 QoS를 지원하기 위한 별도의 독립적인 장비로써 보안을 목적으로 개발되지는 않았으나 웜 바이러스나 DoS 공격으로 유발된 네트워크망의 과부하를 막는데 이들 시스템이 보안 시스템 이상의 성능을 보임으로써 또다른 능동적인 보안시스템의 형태로 인식되고 있다. 그러나 QoS 장비로 개발되었기 때문에 트래픽의 효율적인 분배 및 처리기능은 높이 살만 하나 공격을 탐지하는 어떤 기능도 제공하지 않으므로 별도의 침입탐지 시스템을 따로 구축하여 탐지와 제어 기능을 분리해야 한다는 단점이 있다.

[표 4] 네트워크 보안 시스템들의 성능 비교

	침입탐지 시스템	침입방지 시스템	네트워크 트래픽 관리 시스템	제안 시스템
침입 패킷 탐지능력	높음	높음	제공하지 않음	높음
침입 패킷 제어능력	제공하지 않음	중간 (패킷차단이외의 제어능력은 제공하지 않음)	높음	높음
침입탐지 오류에 의한 패킷손실율	없음 (패킷차단능력은 제공하지 않음)	높음 (침입으로 간주된 패킷은 차단되는 것이 원칙)	중간	중간
장점	다양한 침입탐지 능력	침입에 대한 능동적 대응 능력	네트워크 상황에 맞는 다양한 트래픽 제어 정책 적용 가능	다양한 침입탐지 및 트래픽 제어정책 적용가능
단점	패킷에 대해 어떠한 제어능력도 제공하지 않음	탐지 오류에 의한 패킷 손실 발생	IDS와 독립적인 장비로 구현되어 실시간 패킷 처리능력이 떨어짐	리눅스 시스템에서만 동작

제안 시스템은 네트워크 트래픽 관리 시스템의 이러한 단점을 극복한다. 시스템 내부에 구현된 침입탐지 프레임워크나 외부 침입탐지시스템의 탐지정보를 실시간으로 받아들여 이를 네트워크 트래픽 제어 정책에 맞게 처리한다. 또한 다양한 출력 큐를 사용하여 세밀한 대역폭 관리가 가능하게 되므로 비정상적인 트래픽으로 분류된 패킷이 곧바로 차단되지 않고 단계적인 대역폭 감소를 거치도록 함으로써 탐지 오류에 의한 패킷 손실도 줄일 수 있다. 그러나 제안 시스템은 리눅스 시스템에서만 사용이 가능하다는 단점이 있다.

VI. 결론 및 향후 연구 과제

바이러스나 대규모 해킹에 의해서 네트워크가 점

점 더 취약해짐에 따라 침입탐지시스템과 같은 보안 시스템이 각광받고 있지만 날로 고도화되는 공격을 막기에는 역부족이다. 이에 대한 해결책으로 공격을 사전에 차단한다는 침입방지시스템이 등장하였으나 현재 대부분의 침입방지시스템은 패키징만을 달리한 침입탐지시스템 형태이며 효율적인 트래픽 관리 도구로서 각광받는 네트워크 트래픽 관리 시스템 역시 실시간 침입탐지능력을 제공하지 못한다는 점에서 보안 시스템이 아닌 QoS 시스템으로써의 한계를 드러내고 있다.

본 논문에서는 리눅스 netfilter 시스템을 기반으로 커널 모듈 프로그래밍 기법을 사용하여 침입에 능동적으로 대응하는 비정상 트래픽 제어 시스템을 구현하였다. 리눅스에서 제공하는 iptables를 이용한 패킷 차단기능에 CBQ를 이용한 QOS 메커니즘을 적용하여 트래픽의 침입 유형에 따라 단계적으로 보다 정밀하고 효율적인 패킷 제어 기술을 제공하고자 하였으며. netfilter 시스템을 이용한 커널 레벨의 실시간 침입탐지 프레임워크를 구현하여 기존 응용레벨에서 동작하는 침입탐지시스템 보다 안정성 및 실행 속도 측면에서 성능 향상을 가져올 수 있는 방법을 제시하였다.

향후 성능 평가를 통해 보다 정밀하고 효율적인 CBQ 대역폭 관리 정책을 제시한다면, 리눅스 시스템의 보안 기능을 강화하고 패킷 처리 능력을 향상시켜 보다 나은 서비스를 제공할 수 있는 기반이 될 것으로 기대한다.

참 고 문 헌

[1] Stephen Northcutt and Judy Novak, *Network Intrusion an Analyst's Handbook*, New Riders, 2nd Edition, pp.1～21, 2001.

[2] Qi Zhang and Ramaprabhu Janakiraman, *A Distributed Approach to Network Intrusion Detection and Prevention*, Washington University Technical Report # WUCS-01-30, pp.1～2, January 2001.

[3] Frank Kargl, Jrn Maier, Michael Weber, "Protecting Web Servers from Distributed Denial of Service Attacks", *Proceedings of the 10th tenthinternational conference on World Wide Web*, pp.514～524, April 2001.

[4] Rocky KC Chang, "Defending Against Flooding-Based, Distributed Denial-of-Service Attacks: A Tutorial", *IEEE Communications Magazine*, pp.42～51, October 2002.

[5] Aurobind Sundaram, "An Introduction to Intrusion Detection", *ACM Crossroads Magazine*, 2(4), April 1996.

[6] Joseph S. Sherif, and Tommy G. Dearmond, "Intrusion Detection: Systems and Models", *Proceedings of the 11th IEEE International Workshops on Enabling Technologies:Infrastructure for Collaborative Enterprises*, pp.115～136, June 2002.

[7] Fengmin Gong, *Next Generation Intrusion Detection Systems*, IntruVert Networks INC., pp.2～5, March 2002.

[8] David Marchette, "A Statistical Method for Profiling Network Traffic", *Proceedings of Workshop on Intrusion Detection and Network Monitoring*, pp.119～128, April 1999.

[9] Michael Beck, et al, *Linux Kernel Programming*, Addison Wesley, 3rd Edition, pp.261～272, 2001.

[10] Daniel P. Bovet, Marco Cesati, *Understanding the Linux Kernel*, O'Reilly, 2nd Edition, pp.692～701, 2002.

[11] Alessardo Rubini and Jonathan Corbet, *Linux Device Driver*, O'Relly, 2nd Edition, pp.15～53, 2002.

[12] Rusty Russell, *Linux Netfilter Hacking Howto*, http://kldp.org/Translations/html/Netfilter-hacking-KLDP/

[13] Wenke Lee, "Applying Data Mining to Intrusion Detection: The Quest for Automation, Efficiency, and Credibility", *SIGKDD Explorations*, 4(2), pp. 35～42, December 2002.

[14] Wenke Lee, Salvatore J. Stolfo and Kui W. Mok, "A Data Mining Framework for Building Intrusion Detection Models", *Proceedings of the 7th USENIX Security Symposium*, pp.120～132, May 1998.

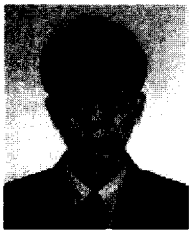
[15] W. Lee, S. Stolfo, P. Chan, E.Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang, "Real Time Data Mining-based Intrusion Detection", *Proceedings of the 2nd DARPA Information Survivability Conference and Exposition*, pp.85～100, June 2001.

〈著者紹介〉

조 은 경 (Eun-kyung Cho) 학생회원
2003년 2월 : 성균관대학교 정보통신공학부 졸업
2003년 2월~현재 : 성균관대학교 정보통신공학부 석사과정
<관심분야> 리눅스 커널, 네트워크 보안, 임베디드 시스템

고 광 선 (Kwang-sun Ko) 학생회원
1998년 2월 : 성균관대학교 정보공학과 졸업
2002년 9월~현재 : 성균관대학교 정보통신공학부 석사과정
<관심분야> 네트워크, 보안, 리눅스

이 태 근 (Taekeun Lee) 학생회원
2002년 2월 : 성균관대학교 전기전자 및 컴퓨터공학부 졸업
2002년 2월~현재 : 성균관대학교 정보통신공학부 석사과정
<관심분야> 임베디드 리눅스, 이동 컴퓨팅 시스템



강 용 혁 (Yong-hyeog Kang) 학생회원
1996년 2월 : 성균관대학교 정보공학과 졸업
1998년 2월 : 성균관대학교 전기전자 및 컴퓨터공학과 석사
2003년 8월 : 성균관대학교 전기전자 및 컴퓨터공학과 박사
2003년 3월~현재 : 극동대학교 경영학부 전자상거래학과 교수
<관심분야> 분산 컴퓨팅, 리눅스 보안, 모바일 에드-혹 네트워크, 임베디드 리눅스

엄 영 익 (Young Ik Eom) 정회원
1983년 2월 : 서울대학교 계산통계학과 졸업
1985년 2월 : 서울대학교 전산과학과 석사
1991년 2월 : 서울대학교 전산과학과 박사
2000년 9월~2001년 8월 : Dept. of Info. and Comm. Science at UCI 방문교수
1993년 3월~현재 : 성균관대학교 정보통신공학부 교수
<관심분야> 분산 컴퓨팅, 이동 컴퓨팅, 이동 에이전트, 시스템 보안, 운영체제