韓國 컴퓨터情報學會 論文誌
第8卷 第2號, 2003. 6.

2003-8-2-1-7

침입탐지시스템의 성능 향상을 위한 버퍼구조에 관한 연구

최 인 수* 장 덕 성**

A Study on Buffer Scheme enhancing Performance In Intrusion Detection System

In-Soo Choi * Doc-Sung Jang **

요 약

침입탐지 알고리즘이 다른 침입탐지 시스템의 알고리즘보다 월등히 뛰어나더라도 버퍼가 가득 찼을 때 포착된 패킷은 시스템 구조상에서 패킷을 손실시킨다. 만일 손실된 패킷이 해킹 될 것으로 우려된다면 시스템 전반에 영향을 미칠 것이다. 본 논문에서는 탐지시스템의 성능을 개선시키는 데에 초점 맞추고자 한다. 임계값을 갖고 있는 버퍼는 정상적인 패킷과 해킹 된 패킷을 구분할 것이다. 임계값 전까지는 버퍼는 정상적인 패킷과 해킹 될 수도 있는 패킷을 받아들일 것이다. 버퍼가 임계값에 도달하였을 때, 폐기된 패킷은 단지 정상적 패킷만 일 것이다. 제안된 해킹방법은 해킹 된 패킷이 바이패스되는 단점을 보완할 것이다.

Abstract

Even though algorithm of intrusion detection is superior to other algorithm in intrusion detection system, it is supposed that captured packet happened bustly lead to lose packet in system architecture when a buffer is full. If packet lost concerned to be hacked, it might impact to system all over. In this paper, try to focus on performance improvement of detection system.

Buffer with threshold value could classify normal packet and hacked packet. The buffer accept normal packet and supposed to be hacked packet until critical value. When buffer reached at threshold value, destroyed packet is only normal packet. Proposed method can complement weakness that bypass hacked packet.

* 동원대학 컴퓨터정보
** 동원대학 e-비즈니스

Ⅰ. 서론

정보화 사회에서의 인터넷 사용자가 급격히 증가하면서 정보통신 산업의 인프라 스트럭처는 우리에게 큰 편의를 제공하지만, 이에 따른 역기능 또한 컴퓨터네트워크를 통한 시스템 취약점을 이용한 해커 활동으로 기업이나 개인에게 큰 피해를 주는 사례가 사회적인 문제로 대두되었다. 이러한 인터넷 보안에 대한 관심의 증가로 사용자 인증시스템, 방화벽, 패스워드 사용 등이 외부의 일차적인 방어 수단으로 사용되고 있지만, 이를 차단하고, 탐지하는 기술이 발달하는 해킹기술을 앞지르지 못하고 있는 것이 현실이다. 침입이란 컴퓨터 사용자가 사용하는 자원의 무결성, 비밀성, 가용성을 저해하는 일련의 행위들의 집합을 말한다.

침입의 분류는 비정상적(anomalous)과 오용(misuse) 침입으로 나눌 수 있다. 비정상적 침입이란 컴퓨터 자원에 비정상적인 행위나 사용에서 근거한다[1]. 여기서 비정상적이란 설정된 프로파일을 위반하는 행위이며, 이를 탐지하는 방법으로는 발생 통계에 의존하는 방법과 특징을 추출하는 방법으로 구분한다. 오용 침입이란 시스템이나 응용 소프트웨어의 약점을 통하여 시스템에 침입할 수 있는 형태로 탐지 방법의 종류는 조건부확률, 전문가시스템, 상태 전이 분석, 키-스트로크 관찰(keystroke monitoring)에 근거해 분류한다. 또한, 침입탐지시스템(IDS : Intrusion Detection System)이란 내부자의 불법적인 사용이나 오용 또는 외부 침입자 등이 네트워크를 통해 시스템의 자원 및 중요한 자료들을 파괴하거나 유출하는 행위를 탐지하여 사전에 조치하는 시스템을 말한다.

침입탐지시스템을 데이터 수집원으로 분류하면 호스트 기반(Host-based IDS)침입 탐지 시스템과 네트워크 기반(Network-based IDS)침입 탐지시스템으로 분류하는데, H-IDS는 호스트에서 침입을 탐지하는 것으로, 그 호스트에 들어온 감사(audit) 기록이나, 패킷을 검사(inspection)하여 침입을 탐지하게 된다. 이러한 H-IDS 침입 행동은 실제로 호스트 로그인 프로세스(login process)를 감시하고, 루트(root) 사용자 행동인 파일처리를 감시해 침입을 발견하는 것으로, 시스템 로깅을 통해 공격자가 어떤 행위를 했는지 등을 알 수 있다. 그러나, 이 방법에서는 우선 호스트에 IDS를 설치하므로 시스템 성능이 저하되고, 네트워크 내의 다른 호스트들이 공격받은 정보를 공유할 수 없다는 것이다. 반면, N-IDS은 네트워크의 모든 수신 트래픽 패킷 데이터를 분석하여 침입을 탐지하는 방법이다. 이는 네트워크 특성 상 LAN에 연결된 여러 시스템 종류나 플랫폼(platform)에 구애받지 않는다. 따라서, 실시간 탐지가 용이하며, 해당호스트에 부하를 주지 않기 때문에 설치로 인한 성능 저하 영향을 제거할 수 있을 뿐만 아니라, 시스템 인증 없이 접근하거나 권한을 초과하는 접근에서도 뛰어난 탐지를 가진다[2]. 그러나, 공격을 탐지하지 하는데는 복잡한 정보 요소가 있어 이를 탐지하고, 분석하는 데는 많은 양의 데이터 교환이 필요하므로, 이를 위한 데이터 축약 과정의 필터링(filtering) 문제가 생긴다. 본 논문에서는 데이터 축약 과정의 필터링의 이전 단계 과정에서 포착되는 패킷의 양을 줄일 수 있는 패킷모듈의 효율적인 버퍼관리 시스템을 제안한다.

Ⅱ. 관련연구

2.1 침입 탐지 시스템의 기술적 구성요소

침입 탐지 시스템은 크게 정보 수집 단계, 정보 가공 및 축약 단계, 침입 분석 및 탐지 단계 그리고 보고 및 조치 단계의 4단계 구성 요소를 갖는다.

(1) 정보 수집(raw data collection) 단계

침입 탐지 시스템이 컴퓨터 통신에 사용되는 패킷 등과 같은 탐지 대상으로부터 생성되는 데이터를 수집하는 감사 데이터(audit data) 수집 단계로써 정상행위 데이터베이스 생성과 비정상 행위 및 시나리오 정책 분석 시 필요한 감사 자료들을 추출한다.

(2) 정보 가공 및 축약(data reduction and filtering) 단계

수집된 감사 데이터는 침입 판정이 가능할 수 있도록 의미 있는 정보로 전환시키는 기능을 한다. 수집 변환된 감사자료에서 ID정보를 이용하여 정상 행위데이터베이스를 구축한다.

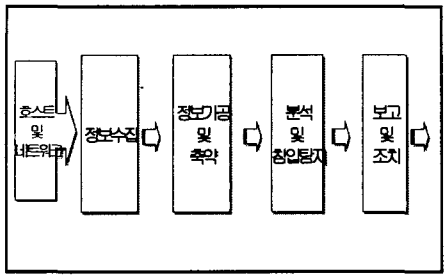

그림1. 기술적 구성요소

(3) 침입분석 및 탐지 단계

침입 탐지 시스템의 핵심 단계이며, 정상행위 데이터베이스를 바탕으로 시스템의 비정상적인 사용인 경우, 시스템의 취약점이나 응용 프로그램의 버그를 이용한 침입에 대한 탐지를 목적으로 하는 오용행위 데이터베이스의 규칙을 비교하여 오용이 아니라고 판단되면 이상 징후로 설정하고 계속 추적을 하고 감시한다. 감시 결과, 정상으로 판단되면 정상행위 데이터베이스를 갱신하고 그러하지 않을 경우 오용으로 판단하여 침입 보고 및 제어 모듈로 전송한다.

(4) 보고 및 조치(reporting and response) 단계

침입 탐지 시스템이 침입탐지 결정 모듈에서 결정된 결과를 보안관리 모듈에 보고하며, 침입으로 판단된 경우 이에 대한 적절한 대응을 자동으로 취하거나, 보안관리자에게 침입 사실을 보고하며, 새로운 침입 형태일 경우 오용행위 데이터베이스를 갱신한다.

2.2 패킷 포착

2.2.1. 스니퍼링

패킷 포착란 네트워크를 돌아다니는 패킷들을 보는 것을 말한다.

일반적인 이더넷 환경에서 라우터(Router)는 내부 네트워크로 향하는 패킷들을 브로드캐스팅(Broadcasting)하게 되고 각 컴퓨터들은 자신의 인터페이스로 들어오는 패킷 중 목적지가 자신인 경우에만 받아들여 이를 운영체제가 처리한다. 패킷 포착은 이처럼 자신에게 전달되는 패킷을 받아 들여 패킷의 내용을 확인할 수 있음을 의미한다. 이러한 패킷 포착은 네트워크의 사용에 대한 통계나 보안을 목적으로 하는 모니터링, 네트워크를 디버깅하기 위한 목적, 스니퍼링 등 다양한 형태로 응용이 가능하다. 일반적으로 호스트들은 필터링 과정을 통해서 자신을 목적지로 하는 패킷만 선택하여 받아들이고 처리하도록 되어 있다.

그러나 이더넷 인터페이스를 방송모드(Promiscuous mode)로 전환하여, 네트워크 트래픽의 실제 목적지에 관계없이 모두 지나가는 패킷을 감시하고 잡아내도록 하는 프로그램을 제작할 수가 있는데 스니핑은 바로 이러한 원리를 역이용한 해킹 기법이다. 스니핑을 하는 방법은 첫째로, 네트워크 디바이스를 열어서 promiscuous mode로 만들고 둘째로, 네트워크를 지나가는 모든 패킷을 포착 후 셋째로, 포착된 패킷을 필터링해서 발신 및 수신 주소, 서비스(telnet, rlogin, ftp....) 그리고 계정과 패스워드가 포함된 데이터를 구분해서 출력한다. 이와 같이 스니퍼 프로그램을 다운받은 뒤 PC에 설치하면 네트워크를 통해 전송되는 개인의 IP가 화면에 자동적으로 포착되어 보이므로, 이를 이용해 개인정보를 자연스럽게 빼낼 수 있다는 것이다. 그러므로 관리자는 특성 컴퓨터가 방송모드로 설정되어 있는지를 주기적으로 점검하여 스니퍼가 실행되고 있는 시스템을 탐지하여 경고 메시지를 보내 후 일정시간이 흐른 후에 해당시스템을 강제로 로그아웃시키는 등 대응 조치를 취하여야 한다.

2.2.2. libpcap

libpcap은 패킷 포착을 용이하게 해주는 라이브러리이다. 각 운영체제 벤더들이 각각의 패킷 포착 도구들을 제공(Linux : SOCKET_PACKET 타입의 소켓을 사용, BSD 계열의 운영체제 : BPF를 사용)하고 있어 개발이나 포팅 등에 어려움이 있기 때문에 각 도구들의 기능을 포함하면서 시스템에 독립적인 libpcap이 등장하게 되었다. 때문에 libpcap은 커널 수준이 아닌 사용자 레벨에서 저수준의 네트워크 모니터링을 가능케 하는 인터페이스를 제공하게 된다. 또한 libpcap은 BPF(BSD Packet Filter)에 기초하는 필터링 메카니즘을 지원하고 있다. 커널 레벨이 아닌 사용자 레벨에서의 필터링이기 때문에 오버헤드가 따르게 됨은 당연하다. BPF는 BSD 계열의 운영체제에서 표준화되어 있다. libpcap을 사용해 패킷을 포착하는 방식의 전체적인 구조는 어딘가에서 정보를 얻어오는 다른 프로그램들과 마찬가지이다. 예를 들어 파일에서 정보를 얻어오는 경우, 우선 파일을 열고, 내용을 읽고, 분석하고, 파일을 닫게 된다. 패킷 포착도 우선 패킷 포착할 디바이스(NIC)나 기존의 저장된 파일을 열고, 패킷을 읽고, 분석한 후 디바이스나 파일을 닫게 된다. libpcap은 각 부분에서 사용되는 인터페이스들을 제공한다. libpcap에서는 파일을 다룰 때 사용하는 file descriptor와 유사한 개념의 packet capture descriptor를 사용한다.

2.2.3 SNORT

snort는 network에서 실시간 traffic 분석과 IP 네트워크 상에서 packet logging을 뛰어나게 수행하는 네트워크 침입 탐지 시스템(NIDS)으로 오픈 소스이며 마이크로 소프트뿐만 아니라 다양한 유닉스 플랫폼에서 사용할 수 있다. NIDS는 호스트 기반 IDS 가 단지 IDS 가 실행되고 있는 호스트만을 감시하는 것과는 달리 전체 네트워크 세그먼트를 감시한다. NIDS는 대부분 방화벽과 함께 사용되기 때문에 공격 자체에 취약하지 않아야 하는 것이 필수적이다. 따라서 snort와 바인드되어 사용되는 모든 인터페이스들은 ip 주소 없이 설치되어야 한다. 그러나, 이는 모든 설정에서 가능한 것이 아니기 때문에, 예를 들어 snort를 isdn 인터페이스 ippp0에 바인드하려는 경우 snort에 대해 독립적인 컴퓨터를 사용해 이를 다이얼업 연결에 대한 방화벽 및 라우터로 설치하는 것을 고려해야 한다.

snort는 프로토콜 분석과 packet의 내용 조사, 패턴매칭이 가능하며, 버퍼 오버플로우나 스텔스 포트스캔, CGI 공격, SMB 탐지, OS fingerprinting 시도 등 다양한 공격과 탐지를 발견하는데 사용할 수 있다. 또한 traffic을 잡아내기 위해 유용한 modular plugin 구조 뿐 만 아니라 모으거나 넘겨 버려야만 하는 탐지 엔진과 같은 매우 유연한 rule language를 사용한다. 또한 이러한 탐지 rule들은 보안 Community를 통해 지속적으로 업데이트 되고, 본인이 쉽게 rule을 작성하여 추가할 수 있으므로 최신 공격에 적응이 쉽다. snort는 syslog, 사용자가 지정한 특정 파일, UNIX의 socket이나 Sambaclient를 이용한 windows popup 메세지와 일체된 alerting mechanism 같은 real-time 경보가 가능하다.

snort 주요한 세 가지 기능이 있는데 첫째로 snort는 tcpdump와 같은 packet sniffer로 바로 사용할 수 있다. 둘째로 네트워크 traffic debugging에 유용한 packet logger 기능이 있다. 마지막으로 완벽한 네트워크 침입 탐지 시스템(NIDS) 기능을 가지고 있다. snort는 외부 호스트의 ip 주소를 이름으로 한 logging 디렉토리에 tcpdump binary format이나 snort의 ASCII format packet을 기록한다. snort는 libpcap이 있는 곳이면 어디에서든 사용이 가능하다.

Ⅲ. 제안 침입탐지 시스템의 설계

기존의 침입탐지시스템은 범용하드웨어시스템의 한계로 인해 네트워크의 부하가 클 때 패킷이 버스트하게 들어올 때 어쩔 수 없는 패킷 손실이 발생하여 침입의 패턴의 일부가 디스카드 되어 버리는 단점이 발생하게 된다. 제안된 기법은 임의의 임계값까지는 버퍼를 공유하다가 큐의 길이가 임계값을 넘어서면 사용자들의 id와 패스워드만을 해킹할 수 있는 특정 포트(telnet, ftp, pop3, login)만을 받아들이는 기법이다. 이 방법은 구현하기가 쉽고, 하드웨어적으로도 처리 할 수 있다는 장점이 있다.

3.1 버퍼의 구조

본 논문에서 제안하는 버퍼는 FIFO구조로 되어 있으며 임계값을 기준으로 2부분으로 나누어져 있다. 임계값을 기준으로, 포착되는 패킷들은 임계값보다 적을 경우는 버퍼에 저장되어 처리되지만 점차적으로 버퍼의 점유 영역이 커져서 임계값에 도달하면 특정포트만 필터링하기 위하여 다른 패킷들은 손실된다. 서버가 어느 정도의 패킷을 처리하여 버퍼의 여유 공간이 생기면 다시 모든 패킷을 저장하여 처리한다.

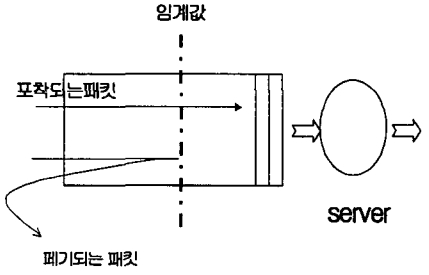

그림2. 제안된 버퍼 구조

2. 알고리즘

포착되는 패킷들이 현재 이용 가능한 버퍼의 크기보다 작으면 처리되고 계속해서 증가하여 임계값에 도달하면 포착되는 패킷의 양은 줄어들 것이다. 또한 버퍼에 저장된

패킷을 처리하여 drop_flag가 true로 세트되어 버퍼의 크기가 임계값 미만으로 줄어들 경우, 다시 모든 패킷들을 처리할 수 있게 되며, 갑자기 패킷이 과도하게 버스트한 상태로 폭주하여 버퍼의 크기를 넘어 설 경우와 서버에서 저장된 패킷을 포착한 속도보다 늦게 처리하여 버퍼의 크기가 꽉 찬 경우에 포착되는 패킷들은 손실된다.

```
While (packet capture) do
    if (drop_flag== false) {
      if(0<=buffer_size<threshold or
  threshold+1<=full) then
            {
               accept packet();
                모든패킷을 받아처리
          buffer_size= buffer_size + 1;}
                if(buffer_size==threshold) then
                  {
        drop_flag= true;
          specialport_check_accept();
     특정포트(23,21,110,513)만 받아 처리
        buffer_size= buffer_size - 1;
      if(0<=buffer_size<threshold ) then
        drop_flag= false)};
      elseif  packet_discard();
        모든 패킷이 손실됨
    }
Endwhile
```

그림3. 제안한 알고리즘 의사 코드

Ⅳ. 분석 및 모의 실험

제안된 임계값을 갖는 버퍼관리기법을 분석하기 위해서 다음과 같은 가정을 둔다.

(1) 임계값 전의 지역에서는 모든 포착되는 패킷의 손실은 발생하지 않는다.

(2) 포착된 패킷들을 처리할 때 패킷 손실이 일어나지 않는다.

(3) 모든 패킷은 독립적으로 분산된 포아성 프로세스를 따른다.

패킷의 전체 수신량은 $\lambda = \lambda 1 + \lambda 2$를 같는 일반적인 M/G/1큐잉시스템과 같다. 여기서 λ1은 특정포트의 패킷들이며 λ2는 버퍼가 임계값을 넘었을 때 손실되는 패킷들이다. 도한 버퍼가 완전히 찼을 때 도착되는 손실들은 특정포트의 패킷들과 임계값을 넘었을 때 손실되는 패킷들 중 하나이므로 셀손실은 특정포트의 패킷손실률 B1과 임계값을 넘었을 때 발생하는 패킷손실률 B2 로 나타낸다. 포착되는 모든 패킷들은 같은 서비스 분포를 갖기 때문에 모든 상태 확률과 모든 손실률에 대한 보존 법칙이 성립된다.

$$\lambda 1B1 + \lambda 2B2 = (\lambda 1 + \lambda 2)B = \lambda B \quad \cdots\cdots\cdots\cdots (1)$$

포착되는 패킷의 확률은 M/G/1큐잉 시스템의 안정상태 확률 PK(K=0,1,2,3,4,T,...N)일 때 B2는 전체에서 B1의 값을 뺀 값이므로 다음 식으로 나타낼 수 있다.

$$B2 = \sum_{K=0}^{N} PK\{1-P(SERVED|K)\} \quad \cdots\cdots\cdots\cdots (2)$$

임계값을 넘었을 때 발생하는 패킷이 처리될 확률 P(SERVERD |K)은 임계값 이전이거나 버퍼가 차지 않은 경우이고, 버퍼가 꽉 찬 경우에는 서비스되지 않고 손실되며 다음 식으로 나타낼 수 있다.

$$P(SERVED|0 < = K < T-1\ R\ T < + K,\ N) =)$$
(단, T : 임계값, N : 버퍼의 크기) $\cdots\cdots\cdots\cdots (3)$

$$P(SERVED|N) = 0 \quad \cdots\cdots\cdots\cdots (4)$$

여기서 1은 서비스될 확률이며 0는 서비스되지 않을 확률을 나타낸다. 임의의 간격동안 임계값을 넘었을 때 발생하는 패킷이 포아송 분포로 포착되는 확률은 다음과 같다.

$$A(n) = \int_0^{\infty} \frac{(\lambda_2 t)^n}{n!} \exp(-\lambda 2t)\ dH(t) \quad \cdots\cdots\cdots (5)$$

이 때 H(t)는 서비스 시간의 확률 분포 함수이며 큐의 위치가 k 일 때 포착되는 패킷은 방금 서비스된 패킷의 나머지 부분에 위치 할 것이다. 포아송 분포에 의해 포착되는 패킷은 현재 서비스되는 패킷의 나머지 시간으로서 상태 k의 조건적 서비스 시간을 갖는다. M/G/1 큐잉시스템에서 조건적 서비스 시간을 위한 평균값의 공식에서 임의의 시점에서는 K개의 패킷이 존재하고 서비중인 서비스 트래픽의 시간 간격이 [t, t+dt]내에서 확률밀도 함수 $r_k(t)$는 다음 식과 같게 된다.

$$r_k(t) = \int_0^{\infty} \lambda h(\mu + t)\exp(-\lambda\mu) \cdot$$
$$[P_0 \frac{(\lambda\mu)^{K-1}}{(K-1)!} + \sum_{j=0}^{K-1} P_{K-J} \frac{(\lambda\mu)^J}{J!}]du \cdots (6)$$

이러한 시간 간격에 대해 확률밀도 함수 $r_k(t)$를 가정할 때 해킹이 우려되지 않는 패킷(폐기시킬 패킷)은 확률 $A_K(n)$을 갖고 도착되며 다음 식으로 나타낼 수 있다.

$$A_K(n) = \int_0^{\infty} \frac{(\lambda_2 t)^n}{n!} \exp(-\lambda_2 t) r_k(t) dt \cdots\cdots (7)$$

지정된 트래픽이 큐잉위치 K(K = 1, 2,,, T,,, S)내에 입력 될 때 현재 서비스된 패킷의 나머지 서비스 동안은 해킹이 우려되는 패킷이 포착되지 않는다고 가정하면 큐잉 위치는 K-1로 이동할 것이다. 나아가 만약 포착된 패킷이 큐잉 위치 K-J로 이동 할 때 초기의 서비스 시간과 J 서비스 시간 동안에 해킹이 우려되는 패킷이 S-K+J개 이상 포착되지 않는다고 가정하면 큐잉공간은 K-J-1로 이동할 것이다. 지정된 패킷은 큐잉 위치 K-J로부터 K-J-1로 이동한다. 정상적인 패킷n은 지정된 패킷 이후에 포착되는 확률 $C_J(n)$을 이용해 정상적인 패킷에 대한 조건적 서비스 확률의 유도를 다음과 같이 나타낼 수 있다.

단계 1

$$C_0(n) = \{A_{k(n)} \text{ if } (0 \le n \le T-K-1 \text{ OR } T \le n \le s-k)$$
$$0 \text{ otherwise} \cdots\cdots (8)$$

단계 2

J $(0 \le J \le K-1)$:

$$C_J(n) = \{C_{J-1}(n) * A(n) \text{ if } (0 \le n \le T-K-1+J \text{ OR}$$
$$T \le n \le S-K+J) \ 0 \text{ otherwise} \cdots\cdots (9)$$

마지막 단계

$$P(served|K) = \sum_{n=T}^{S-1} C_{K-1}(n) + \sum_{n=0}^{T-1} C_{K-1}(n) \cdots (10)$$

$$P(served|K) = \sum_{n=0}^{S-1} C_{K-1}(n) \cdots\cdots (11)$$

정상적인 패킷이 폐기될 확률은 식(2)로부터 얻을 수 있다.

본문에서는 실험한 환경은 Redhat7.2, Snort V1.8.4, MYSQL V3.23.49a, ACID.9.6b20 환경에서 테스트하였다.

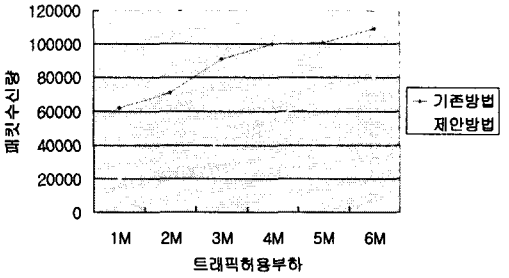

그림4. 포착한 패킷 수신량

〈그림 4〉에 나타난 결과는 기존의 방법 보다 패킷이 적게 수신되게 되는데 이것은 버퍼의 임계값을 조절하여 패킷의 수신을 결정하는 turn-off 신호의 제어구조와 패킷 필터링을 위한 비교 연산의 수행되기 때문이며 그림5의 결과로 침입의 여지가 있는 패킷의 손실이 감소된 것을 알 수 있었다.

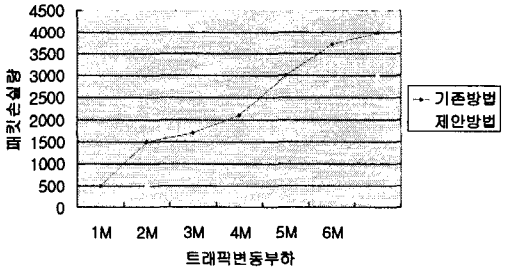

그림5. 침입의 여지가 있는 패킷 손실량

V. 결론

이 논문은 broadcast방식의 지역네트워크 특징을 이용해, 네트워크 데이터를 포착 분석하여 버퍼가 찾을 때 포착되는 패킷이 전부 손실되는 단점을 보완하기 위하여 버퍼에 임계값을 두어 임계값 이후에 포착되는 패킷을 필터링하여 침입이 여지가 있는 패킷만 포착하여 처리하는 방법을 제안하였다. 실험의 결과로 침입의 여지가 있는 패킷의 손실이 줄어든 것을 알 수 있었으며 turn-of의 시간이 충분히 빠르게 작동된다면 네트워크 부하도 줄일 수 있을 거라고 생각된다.

참고문헌

[1] Dorothy E. Denning, "An Intrusion-Detection Model.", IEEE Trans. on Software Engineering, No.2, pp.222-232, 1997.

[2] Biswanath Mukherjee, L. Todd Heberlein, and Larl N. Levitt, "Network Intrusion Detection", IEEE Network May/June, pp. 26-41, 1994.

[3] S. Kumar, 'Classification and Detection of Computer Intrusion", PhD thesis, Purdue University, West Lafayette, IN 47907,

[4] http://www.certcc.or.kr

[5] http://www.snort.ogr/docs

저 자 소 개

최 인 수

1992 광운대학교 전자계산학과 (이학석사)

2002 강원대학교 컴퓨터과학과 (박사수료)

현재 동원대학 컴퓨터정보과 교수

〈관심분야〉 network security, ESM, 객체지향시스템

장 덕 성

서울산업대학교산업공학과(학사)

고려대학교 경영정보(석사)

경원대학교 경영정보(박사)

동원대학 e-비즈니스과 교수

〈관심분야〉 e-비즈니스, 문서보안, Awareness