濟州大學校 尖端技術硏究所 論文集, 第13卷, 2號, pp.153-160, 2002. 12
Cheju Nat'l Univ. Res. Inst. Adv. Tech. Jour., Vol.13, No.2, pp.153-160, 2002. 12

Windows IM Driver의 HFP 필터를 통한 IDS 성능 향상

김 성 윤* · 김 경 식**

Enhancement of IDS performance by using a HFP filter in Windows IM Driver

Soung-Yun Kim* and Kyung-Sik Kim**

ABSTRACT

This thesis propose a system that designed and implemented to overcome efficiently the bottleneck occurring in IDS(Intrusion Detection System). IDS must analyze all incoming packets to detect the signatures of intrusion. With a single packet to fail analyze, it may lose a critical clue of intrusion and fail to detect an intrusion. The proposed system is developed at the layer of IM(Intermediate) driver in MS(Microsoft) Windows. Most of packet capture modules used in IDS is developed in protocol driver layer, and then the proposed system is independent of various IDS. The proposed system can analyze all of packets existing in a fast ethernet network and filter HFP(Hacking-Free-Packet) completely. The performance of IDS using the proposed system is improved by increasing the amount of web traffic due to heavy network traffic.

Key Words : IDS, bottleneck, intermediate driver, packet capture module, independent, HFP, heavy network traffic.

Ⅰ. 서 론

인터넷 사용자의 폭발적인 증가와 함께 해킹에 의한 피해 사례 또한 급증하고 있다[1]. 이에 보안 관련 연구 및 제품 또한 급속도로 진전, 출시되고 있는 상황이며, 그 중에서도 방화벽이나 침입탐지 시스템은 이와 관련된 잘 알려진 분야라고 할 수 있다. 이러한 제품들은 인터넷 접속량이 늘어남에 따라 높은 네트 트래픽을 다룰 수 있어야 하는데, 이는 이들의 특성상 네트워크로 수신되는 모든 패킷에 대한 분석이 필요하기 때문이다. 네트워크의 모든 패킷을 분석하고, 그에 대응하기 위해서는 높은 오버헤드가 발생하게 되고, 결국 패킷 손실이나, 실시간 적인 대응을 할 수 없게 만든다[2]. 실례로 Mirecom Lab의 보고서에 따르면, 기가비트 이더넷에서 986.94Mbps 백그라운드 트래픽을 갖는 네트워크 망에서 침입탐지율은 44%정도라고 보고되고 있다[3].

이러한 높은 네트워크 트래픽에서의 침입 탐지율을

* 제주대학교 전기전자공학과 대학원
Graduate course, Dept. Electrical & Electronic Engineering, Cheju Nat'l Univ.
** 제주대학교 전기전자공학부, 첨단 기술 연구소
Faculty of Dept. Electrical & Electronic Engineering, Res. Inst Tech., Cheju Nat'l Univ.

개선시키고자 많은 연구가 이루어지고 있는데, 그 분야는 크게 하드웨어적인 분야와, 소프트웨어적인 분야로 나누어진다. 하드웨어적인 분야로는 침입탐지 시스템과의 연동을 통해 패킷 필터링을 할 수 있는 임베디드 시스템이 제안되어지고 있다[4]. 소프트웨어적인 측면에서는, 침입탐지 시스템 자체의 탐지 알고리즘 개선 분야와[5], 침입탐지 시스템의 패킷 캡쳐 모듈 의 처리속도 개선에 관한 분야로 진행되어지고 있다. 패킷 캡쳐 모듈로는 공개 소스인 Pcap[6]과 같은 프로토콜 드라이버 계층에서 구현된 패킷 드라이버가 사용되어 진다. 하드웨어적으로 새로운 임베디드 시스템을 구축하는 것은 비용적인 면에서 효율성을 얻기 힘든 단점이 있다. 소프트웨어적인 측면에서, 침입탐지 시스템의 탐지 알고리즘을 개선하거나 패킷 캡쳐 모듈을 교체하는 것은 기존의 시스템에 변경을 가하는 것으로, 다양한 기존 침입탐지 시스템에 독립적으로 적용될 수 없는 단점이 있다.

여기에서는 기존의 시스템을 유지하면서, 높은 네트워크 트래픽에서의 침입탐지 시스템의 탐지 성능을 향상시킬 수 있는 시스템을 제안하고 있다. 이러한 제안은 MS Windows의 네트워크 드라이브 구조에 기인하는데, 대부분의 침입탐지 시스템의 패킷 캡쳐 모듈이 프로토콜 드라이버 계층에서 구현되었음을 고려한 것이다. 제안된 필터 모듈은 MS Windows 인터미디엇 드라이버 계층에서 구현되었으며, 이는 프로토콜 드라이버 보다 하위 계층으로, 기존의 네트워크 구조를 해치지 않고 패킷의 송수신에 관여할 수 있도록 제안된 구조이다[7, 8]. 이를 이용해 해킹과 무관한 패킷(Hacking-Free-Packet)을 필터링 하여, 침입탐지 시스템의 탐지 성능을 향상시키고자 하였다. 이러한 구현의 장점으로는 기존의 패킷 캡쳐 모듈에 독립성을 가져, 다양한 침입탐지 시스템을 변경하지 않고 적용 가능한 점이다.

이를 위해 우선 인터미디엇 드라이버를 개발하기 위한 MS Windows 네트워크 구조를 II 장에서 설명할 것이며, III 장에서는 개발된 인터미디엇 드라이버에 패킷 필터 모듈을 넣기 위한 필터 모듈 구현 방법에 대해 논의할 것이다. 또한 제안된 필터 모듈에서 필터링할 HFP(Hacking-Free-Packet)에 대해 설명할 것이다. 마지막 IV 장에서 구현된 필터 모듈을 인터미디엇 드라이버에 적용한 후, 실험을 위한 네트워크 환경 구현방법과 실험 과정에 대해 설명할 것이다. 그후에 실험 및 실험결과에 대한 고찰후 마지막으로 결론을 맺을 것이다.

II. MS Windows 인터미디엇 드라이버

MS Windows는 커널 모드 네트워크 드라이버로 미니포트, 인터미디엇, 프로토콜인 세 가지 형태의 드라이버 스펙을 제안하고 있다. 미니포트 드라이버는 NIC(Network Interface Card)을 직접적으로 제어하고, 상위레벨 드라이버와의 인터페이스를 제공하는, 일반적으로 네트워크 카드와 함께 제공되어지는 드라이버이다. 프로토콜 드라이버는 TDI(Transport Driver Interface)와의 통신과 하위 계층과의 통신을 인터페이스 하게 된다. 이러한 프로토콜 드라이버에는 TCP/IP, NETBEUI, IPX/SPX등이 있으며, 침입탐지 시스템에서 패킷 캡쳐를 위해 사용되어지는 패킷 캡쳐 모듈도 이 계층에서 구현되어 진다. 인터미디엇 드라이버는 일반적인 네트워크 구조에는 계층화되어 있지 않으며, 필요시 개발되어 계층화 될 수 있다. Fig. 1은 세가지 타입의 드라이버 사이의 계층 구조도와 NDIS(Network Driver Interface Specificateion) 관계를 보여준다. NDIS는 Windows 네트워크 드라이버사이의 인터페이스를 제공하며, 또한 드라이버간의 상태 정보나 각종 파라미터(함수 포인터, 핸들 등)에 대한 관리를 하는 라이브러리이다. 이러한 인터미디

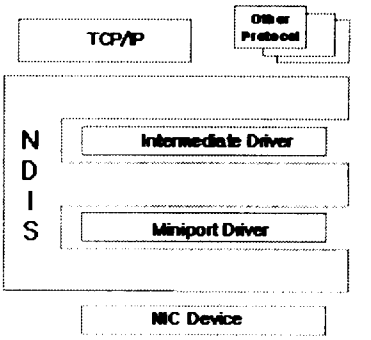

Fig. 1. Relationships between miniport drivers, protocol drivers, intermediate drivers and NDIS.

엇 드라이버의 주 사용 목적은 크게, 프로토콜 드라이버에 알려지지 않은 미니포트 드라이버간의 송수신 미디어를 변환하는 것이다. 그 외에도 최근 들어 로드 밸런싱이나, 네트워크 데이터의 모니터링에도 사용되어지고 있다.

여기에서는 인터미디엇 드라이버를 패킷 필터링을 위해 제작, 사용하였다. 이를 위해서는 Windows 네트워크 드라이버간의 데이터 송수신 과정에 대한 이해가 필요하다. 우선, 드라이버간 데이터 송수신에는 NDIS_PACKET이라는 형태가 사용되어진다.

2.1. NDIS_PACKET

드라이버간 데이터의 송수신은 NDIS_PACKET이라는 포인터를 이용한다. 이는 단지 NDIS_BUFFER라는 포인터를 가리키는 포인터로, 실제 데이터를 이동시키는 대신, 이를 이동시키게 된다. NDIS_BUFFER는 가상 메모리 번지 값을 지시하고, 이 가상 메모리는 실제 데이터를 가리키게 된다. Fig. 2는 이를 도식적으로 보여준다.

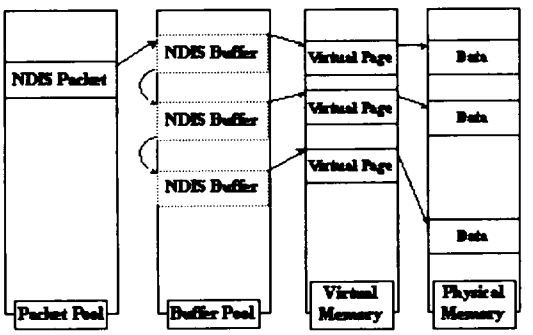

Fig. 2. NDIS_PACKET structure.

이러한 구조는 Windows의 네트워크 구조의 계층화에 기인한 것으로, 인터미디엇 드라이버에서 패킷을 분석하기 위해서는 NDIS_PACKET에서 실제 데이터, 이더넷 프레임(Ethernet Frame)을 얻어내야 하는 과정이 필수이다.

인터미디엇 드라이버의 동작과정은 크게 초기화/종료, 송/수신, 설정과정으로 나누어진다. 다른 과정에 비해 여기에서 중점적으로 다루는 것은 수신 과정이다. 이는 침입탐지 시스템이 수신되는 패킷에 대한 탐지가 주목적인 데에 기인한다.

2.2. 인터미디엇 드라이버 수신과정

인터미디엇 드라이버의 수신과정은 Fig. 3에서 보듯이, NIC을 통해 데이터가 수신되면 미니포트 드라이버는 *NdisMIndicateReceivePacket* 이라는 NDIS 함수를 통해 상위 드라이버로 NDIS_PACKET을 올려보내고, 인터미디엇 드라이버 수신단의 ProtocolReceive(또는 ProtocolReceivePacket) 함수를 호출하게 된다. 일반적으로 인터미디엇 드라이버가 어떠한 패킷 조작과정이 필요 없다면, 단지 다시 *NdisMindicateReceivePacket* 함수를 호출함으로서 프로토콜 드라이버로 데이터를 넘겨주게 된다. 패킷을 수신한 프로토콜 드라이버는 수신완료 메시지를 미니포트로 보내기 위해 NdisReturnPacket을 호출하게 되고, 이는 인터미디엇 드라이버를 통해 미니포트 드라이버로 전송되어 진다. 이러한 수신과정이 완료 되면, 미니포트는 수신된 패킷에 할당되었던 자원을 제거하게 된다.

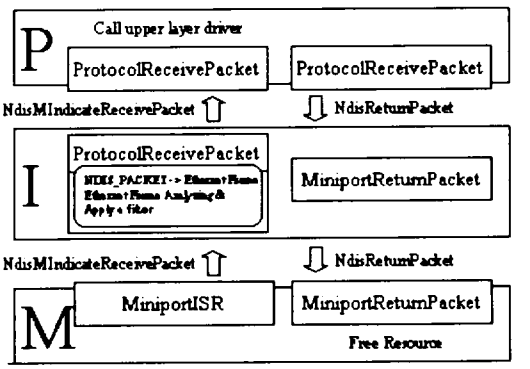

Fig. 3. Receiving Process in intermediate driver.

여기에서는 수신된 패킷에 대해 필터 모듈을 추가해야 하므로 인터미디엇 드라이버 수신단에 추가적인 과정이 필요하다. 우선, 미니포트 드라이버에서 넘어온 NDIS_PACKET을 통해 이더넷 프레임을 얻어오는 과정이 필요하고, 그 후에 이더넷 프레임 분석 후, 필터 적용 유무를 결정하는 과정이 추가되어야 한다.

III. HFP 필터

3.1. 패킷 필터

패킷 필터링이란 특정한 패킷에 대하여 일정한 정책을 적용하여 해당 패킷을 받아들일 것인지 말 것인지(Accept or Reject)를 결정하는 과정을 말한다. 일반적으로 네트워크 카드는 자신에게 오거나 브로드캐스트 되는 패킷만을 수신하게 된다. 하지만 네트워크의 모든 패킷을 모니터링 하고, 침입탐지 시스템에서 불법적인 접근 패킷을 탐지하기 위해서는 네트워크 카드를 스니핑 모드(PROMISCUOUS)로 설정하여야 한다.

패킷 필터는 다음의 4가지 조건을 만족해야 한다.

- Real-time performance
- No packet dropping
- Flexibility
- Scalability

이 조건 중 패킷필터의 가장 중요한 요소는 실시간적인 패킷 분석 능력과 놓치는 패킷이 없어야 한다는 것이다. 이러한 패킷 필터 모듈 중 안정적인 측면과, 속도적인 측면에서 가장 많이 사용되는 것이 BPF (BSD Packet Filter)이다. BPF의 가장 큰 특징은 패킷을 찾아내는데 있어 전통적인 Tree 모델이 아닌 CFG(Control Flow Graph)를 사용 한다는 것이다 [9]. Fig. 4는 CFG 모델을 이용하여 수신된 패킷에서 "foo" 호스트인 패킷을 찾아내는 과정을 보여준다.

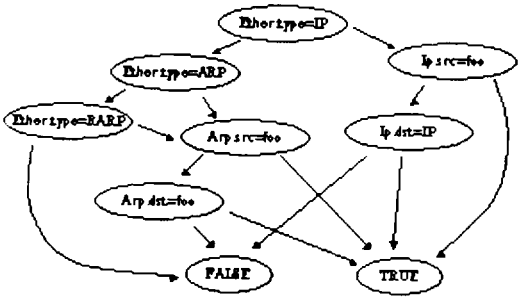

Fig. 4. CFG Model for "host foo".

이러한 CFG 모델의 장점은 Tree 모델에 비해 패킷에 대해 각각의 헤더 값을 조사할 필요가 없이, 한번의 조사로, 패킷이 IP 패킷인지, ARP 패킷인지를 결정할 수 있다는 것이다. 즉, 패킷 정보는 필터에 기억되어지고 한번 계산되어진 값은 매번 다시 계산되어질 필요가 없게 된다.

3.2. ASL에 의한 패킷 필터 모델 구현

제안된 필터 모듈도 CFG 모델로 구현되었는데, 이를 위해 ASL(Audit Specification Language)[10]언어를 사용하였다. ASL은 기본적으로 다음과 같이 정의되어진다.

pattern / condition -> reaction

*pattern*은 이 식에 사용되어지는 값을 말한다. 패킷 필터 모델에서는 수신된 네트워크 패킷, 즉 이더넷 프레임을 말한다. *condition*은 패킷에 대한 간단한 비교 과정으로, 패킷의 각각의 필드 갓에 대한 연산 과정이다. *reaction*은 *condition* 연산과정 이후, 행해져야할 행동을 말한다.

또한 제안된 시스템에서는 패킷 접근시 바이트 흐름으로 다루었다. 예를 들면, 이더넷 헤더의 프로토콜 필드는 "(short)packet[12]"로 접근하는 것을 말한다. 이러한 접근의 단점은 헤더 각각의 필드 접근시 타입 변환을 해주어야 하는 단점이 있다. 하지만 불필요한 필드의 접근을 막고, 여러 프로토콜에 독립적이고, 확장성 있게 만들어 준다. 즉, 헤더구조를 정의하지 않아도, UDP나 ICMP와 같은 프로토콜에 쉽게 접근 가능하도록 한다[11].

ASL에 표현되어진 구문은 그 규칙에 의해 쉽게 CFG 모델로 변환이 가능하다. 수신된 패킷에서 "foo" 호스트를 찾는 과정은 아래와 같다. (foo 호스트의 IP 주소가 11.22.33.44 인 경우)

packet(p)/(p.e_type == ETHER_IP) && (p.s_addr == 11.22.33.44) -> message("host foo")

*packet(p)*는 수신된 이더넷 프레임을 말한다. *p.e_type*은 *p.s_addr*을 수행하기 위한 조건 확인(Constaint Check) 연산 과정으로, IP 주소를 접근하기 위해서는 반드시 해당 패킷이 IP 패킷인지를 확인해야 하는 과정이다. message("host foo")는 해당패킷이 검출 되었을 때 행해져야할 행동을 말하는 것으로 간단히 메시지를 뿌리는 것이다. 이렇게 ASL을

통해 구현된 구문은 Fig. 5와 같이 간단히 CFG 모델로 구현되어 질 수 있다.

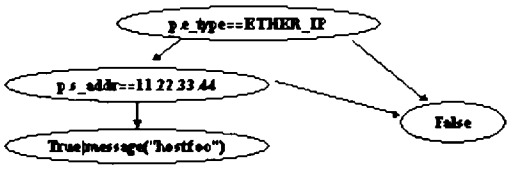

Fig. 5. Sample filter for "host foo".

3.3. HFP(Hacking-Free-Packet)

인터넷에서 HTTP(Web)의 패킷은 가장 많은 트래픽을 차지하고 있다[12]. 이러한 패킷은 대부분 해킹과 무관한 패킷일 경우가 많고, 또한 침입탐지 시스템의 작업 능률을 떨어트리는 주 요인중 하나이다. 이러한 HTTP 패킷에서의 해킹과 무관한 패킷, HFP를 찾아내어 필터링 한다면, 침입탐지 시스템의 탐지율은 더욱 높아질 것이다.

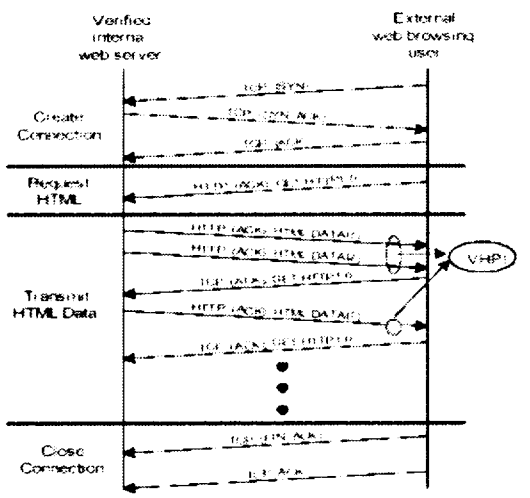

Fig. 6. Sequence of HTTP and HFP.

HTTP 패킷은 크게 3 종류로 구분되어진다. Fig. 6은 클라이언트가 서버에 접속하여 정보를 수신하는 과정을 보여준다[11].

1) TCP 연결 설정 패킷

대부분의 침입탐지 시스템은 IP 주소에 기반한 TCP 세션을 관리하며, 이 세션 정보에 의해 침입을 탐지한다. TCP 연결 설정이 없다면, 침입탐지 시스템은 세션을 재구성 할 수 없다. 그리고 해킹 기술중에는 "SYN Flooding"이나 "ACK Storm"을 이용한 DDoS(Distributed Denial of Service)와 같은 것이 있다[1]. 그러므로 이 패킷은 HFP가 될 수 없고, 침입탐지 시스템으로 보내져야 한다.

2) 클라이언트에서 보내어 지는 HTTP 데이터

웹서버가 공격당하는 주된 원인은 CGI(Common Gateway Interface)와 같은 웹 페이지의 구조적인 약점에 기인한다. 이 경우에 침입자가 쿠키와 같은 정보를 이용하여, 웹 프로그램에 불법적인 실행 명령이나 스크립트를 실행할 수 있다. 이러한 패킷도 침입탐지 시스템에 의해 분석되어져야 한다.

3) 서버로부터 전송되어지는 HTTP 데이터

일반적으로 공인된 웹서버들은 침입에 대한 피해를 입고 있지 않고, 여기서 전송되는 데이터는 대부분이 무해하다. 공개 침입탐지 시스테믄 Snort인 경우 HTTP와 관련된 약 440개의 룰을 가지고 있다. 이는 대부분 연결설정과 관련된 것으로, 서버로부터 전송되는 데이터에 대한 룰은 "잘못된 URL"이나 "403 Forbidden"과 관련된 3가지 룰만이 존재하는데, 이는 전체 룰의 0.68%밖에 되지 않으므로, 높은 네트워크 트래픽 상황에서 이를 회생함으로서 침입탐지 시스템의 탐지율을 향상시킬 수 있다.

HTTP 패킷에서 HFP를 찾아내여 Drop, 즉 침입탐지 시스템으로 보내지 않는 필터 모델은 ASL에 의해 다음과 같이 나타내어 질 수 있다.

```
packet(p)|(p.e_type == ETHER_IP) &&
(p.protocol == IP_TCP) &&
((p.tcp_sport == HTTP) ||
(p.tcp_dport == HTTP))&&
((p.tcp_flag == ACK) ||
(p.tcp_flag == ACKPSH))&&
(p.tcp_data != GET) -> drop
```

Fig. 7은 HTTP에서 HFP에 대한 CFG 필터 모델이다. HTTP에서의 HFP의 조건식은 7개가 필요함을

알 수 있다. 이 7개의 BOOLEAN 조건식 함수를 구현하여 인터미디엇 드라이버에 필터 모듈로서 적용시킬 수 있으며, 추가적으로 NDIS_PACKET을 이더넷 프레임으로 변환하는 함수가 필요하다.

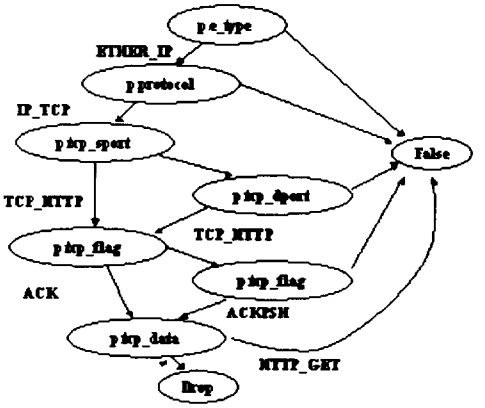

Fig. 7. CFG Model for HFP filter.

Ⅳ. 제안된 시스템의 구현 및 고찰

4.1. 시스템 구현

침입탐지 시스템의 탐지율을 측정하기 위해 Fig. 8과 같은 독립적인 네트워크 및 시스템을 구축하였다.

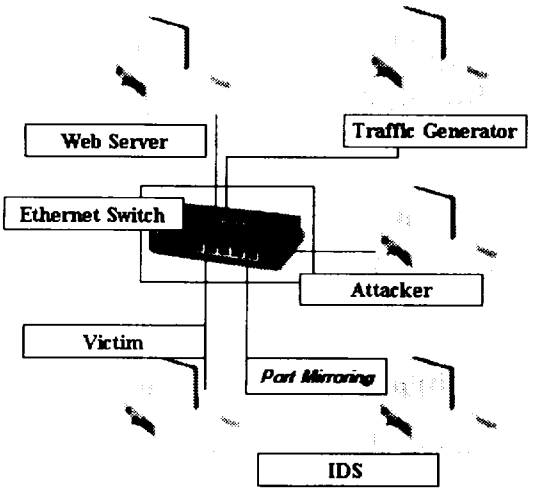

Fig. 8. Empirical network configuration.

1) IDS

공개용 침입탐지 시스템인 Snort-1.9.0[13]의 win32 버전을 사용했다. Snort는 네트워크 패킷을 수집하기 위해 PCAP을 사용하는데 Win32용 버전인 Winpcap-2.3을 이용하였다.

2) Attacker

침입을 위해서 Land Attack[14] 기술을 사용하였다. Land Attack 공격 패킷 다량으로 보냈을 때, 침입탐지 시스템의 탐지해 내는 횟수의 비율을 탐지율로 계산하였다.

3) Traffic Generator

네트워크 트래픽은 두 종류의 패킷을 이용해 발생시켰다. 하나는 HTTP 패킷이고, 이는 http_load[15]를 이용하였는데, HTTP패킷을 무작위로 원하는 만큼 만들어 줄 수 있었다. 또 하나는 UDP 트래픽으로, MGEN[16]을 이용해 발생 시켰다.

4) Web Server, Client

Web Server는 Apache-1.82[17]을 사용하였으며, Client는 Windows 시스템이 설치된 PC를 사용하였다.

5) Ethernet Switch

각각의 시스템은 Extreme의 Summit24 Switching 허브를 통해 100Mbps로 연결되어지도록 했으며, 침입탐지 시스템으로 모든 네트워크 트래픽이 전달될 수 있도록 포트 미러링(Port Mirroring)을 설정하였다.

4.2. 구현 결과 및 고찰

네트워크 트래픽은 HTTP 패킷과 UDP 패킷을 발생시키면서, 총 네트워크 트래픽 사용량을 10%에서 99%까지 조절하며 실험하였다. Land Attack 패킷은 1000개를 사용했으며, 침입탐지율은 10번에 걸쳐 조사한 후 평균값을 택하였다. 또한 네트워크 트래픽에 대한 HTTP 트래픽 비율에 따른 탐지율을 측정하기 위해 HTTP 트래픽과 UDP 트래픽의 비율을 변화시켰다.

Table. 1은 네트워크 트래픽에 따른 침입탐지 율을

Table. 1. Attack Detecion Rate.

Traffic	10%	20%	30%	40%	50%	60%	70%	80%	90%	99%
H1U9	100	100	97.5	96.9	94.2	91.8	80.9	70.4	68.5	64.2
H1U9*	100	100	100	96.4	94.2	92.2	81.2	72.2	70.3	66.2
H3U7	100	99	97.9	97	94	91.8	82.3	71.3	68.2	65
H3U7*	100	100	100	100	96.4	94.2	86.2	79.9	76.2	74.4
H5U5	100	100	97.0	96.7	94.4	91.1	82.6	71.7	68.9	64.8
H5U5*	100	100	100	100	99	97.4	88	84.2	83	81.4
H7U3	100	100	97.9	96.4	94.8	91.9	81.6	72.4	69.6	65.4
H7U3*	100	100	100	100	100	98.3	98.2	98.4	97.8	96.4
H9U1	100	100	97.9	96.8	94.1	91.2	79.9	71.6	69	63.9
H9U1*	100	100	100	100	100	100	100	99	98.4	98.2

보여준다. HxUy는 전체 네트워크 트래픽에서의 HTTP 트래픽과 UDP 트래픽의 비율을 나타내며 그 때의 탐지율을 나타낸다. '*'은 제안된 필터 드라이버가 설치된 침입탐지 시스템에서의 탐지율을 나타낸다. 결과에서 보듯이, 네트워크 트래픽이 증가할수록 침입탐지 시스템의 탐지율은 감소하는 것을 알 수 있다. 하지만 네트워크 트래픽중 HTTP패킷의 비중이 높아질수록, 제안된 필터 드라이버를 설치한 시스템이 기존의 시스템 보다 탐지율이 높음을 알 수 있다.

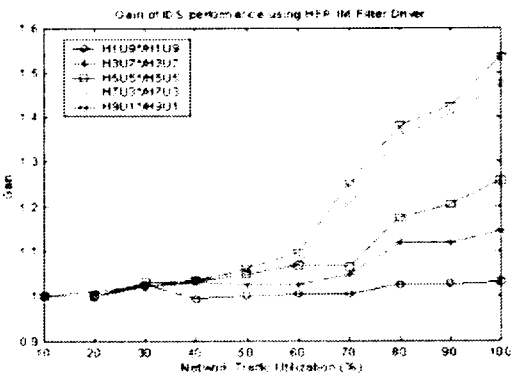

Fig. 9. Gain of IDS performance using HFP IM filter driver.

Fig. 9는 Table 1의 결과에 따른, 기존의 시스템에 대한 제안된 시스템의 탐지 이득을 보여준다. 네트워크 트래픽 사용량이 60% 이상이 되었을 경우, HTTP 트래픽이 UDP 트래픽에 비해 많은 비중을 차지할수록 탐지 이득이 현저히 높음을 알 수 있다.

V. 결 론

높은 네트워크 트래픽에서의 침입탐지 시스템의 탐지율이 떨어지는 현상을 MS Windows 인터미디엇 드라이버에 HFP 필터를 넣음으로서 이 문제를 해결하였다. 이를 위해 우선 인터미디엇 드라이버를 개발하였고, ASL을 이용해 BPF에 기반한 패킷 필터 모듈을 제안하였다. 실험에서 보듯이 제안된 시스템은 높은 네트워크 트래픽에서 HTTP 패킷의 비율이 높아질수록 탐지율은 기존의 시스템 보다 현저히 높아짐을 볼 수 있었다. 이러한 HFP 필터에 의한 침입탐지율 개선은 자칫 HFP를 이용한 침입은 탐지 할 수 없는 false-negative(침입이 아니라고 오인)문제를 가지고 있다. 그러므로 HFP에 대한 지속적인 분석 및 갱신이 필요하고, 이를 유연성 있게 필터링 모듈에 추가할 수 잇느 시스템에 대한 연구 및 디자인이 필요할 것이다.

참고문헌

1) 이현우. 네트워크 공격기법의 패러다임 변화와 대응 방안 -Part I. II : http://www.certcc.or.kr. 200 1.
2) Willam Stallings. 통신망 정보보호. 도서출판 그린. 1997.
3) Mier Communication Inc. "Lab Testing Summ ary Report". April 2001 : http://www.mier.com/ reports/intrusion/Intrusion-comPerfVl-5-04-01.pdf
4) Jongwook Moon. Enhancing IDS performance through dropping hacking-free packets. Ajou University. 2001.
5) Martin Roesch. "Snort-Lightweight Intrusion Detection for Networks". USENIX LISA '99 Conference : http://www.snort.org/docs/lisapap er.txt.
6) Pcap : http://www.tcpdump.org.
7) MSDN : http://msdn.microsoft.com/library.
8) HanTechSnS : http://hantech.cheju.ac.kr.
9) S. McCanne and V. Jacobson. "The BSD Packet Filter: A New Architecture for User-level Packet Capture". In Proceedings of the 1993 Winter USENIX Conference. SanDiego. CA. pp1-3. January 1993.
10) Guang Yang. A real time packet filtering module for network intrusion detection system. 1998.
11) W. Richard Stevens. "Unix Network Programming". Prentice Hall PTR.
12) K. Thompson. G.J. Miller. and R. Wilder. "Wide-Area Internet Traffic Patterns and Characteristics." IEEE/ACM Transations on Networking. pp. 10--23. November 1997.
13) Snort : http://www.snort.org.
14) Land Attack. Insecure : http://www.insecure. org/sploits/land.ip.DOS.html.
15) ACME Labs. "multiprocessing http test client" : http://www.acme.com/software/http_load.
16) Naval Research Lab. "MGEN User's Guide : http://manimac.itd.nrl.navy.mil/MGEN.
17) Apache : http://www.apache.org.