디지털산업정보학회 논문지 제13권 제4호-2017년 12월
http://dx.doi.org/10.17662/ksdim.2017.13.4.141

오픈소스 IDS/IPS Suricata를 적용한 Windows7과 Ubuntu 성능 비교 분석

석 진 욱* · 김 지 명** · 최 문 석***

The Comparative Study on Performance Analysis of Windows 7 and Ubuntu Applying Open Source IDS/IPS Suricata

Seok Jinug · Kim Jimyung · Choi Moonseok

〈Abstract〉

Nowadays, It is undeniable that the threat of network security is growing as time flows due to worldwide development of wire/wireless, various Internet platform and sophisticated hacking techniques. The amount of traffics that Network security solution has to handle is increasing and recently many occurrence of explosive traffic attacks from PulseWave are being observed which has many similar characteristics to New DDos. Medium and small sized firms abroad have developed and distributed Snort and Suricata that are based on open-source Intrusion Detection System(IDS) / Intrusion Prevention System (IPS).

The goal of this study is to compare between Windows7 by applying suicata 4.0.0 32bit version and Ubuntu 16.04.3 LTS by applying suicata 4.0.0 version which is an open source Intrusion Detection System / Intrusion Protection System that uses multi threads method. This experiment's environment was set as followed C1100 server model of Dell, Intel Xeon CPU L5520 2.27GHz*2 with 8 cores and 16 threads, 72GB of RAM, Samsung SSD 250GB*4 of HDD which was set on RAID0.

According to the result, Suricata in Ubuntu is superior to Suricata in Windows7 in performance and this result indicates that Ubuntu's performance is far advanced than Windows7. This meaningful result is derived because Ubuntu that applied Suricata used multi core CPU and RAM more effectively.

Key Words : Suricata, Intrusion Detection System(IDS), Intrusion Prevention System (IPS), Windows7, Ubuntu

Ⅰ. 서론

현재 우리나라 및 전세계적으로 증가하고 있는 네트워크 보안 위협의 증가는 네트워크 보안 솔루션 및 각종 보안 솔루션들이 감당해야 할 트래픽이 폭

* 시큐어원 컨설팅팀 전임컨설턴트(주저자)
** 한국인터넷진흥원 공인전자주소팀 선임연구원
*** 우리에프아이에스 정보보안부

발적으로 늘어나고 최근 들어 신종 디도스의 행태인 펄스웨이브 등과 같이 수초 내 폭발적인 트래픽 공격을 발생하는 공격이 생겨나고 있다. 이에 따라 네트워크 보안 시스템인 방화벽, 침입탐지시스템, 침입방지시스템 등의 보안 솔루션 시장이 더욱 커지고 있다. 해외에서는 중소기업 및 연구로 이용하는 대표적인 오픈소스 침입탐지시스템/침입방지시스템인 Snort와 Snort의 대안으로 만들어진 Suricata를 개발하여 무료 배포하고 있다.

네트워크 공격기법을 방어하기 위하여 과거부터 현재까지 방화벽을 많이 사용하고 있고, 이러한 방화벽은 네트워크 Port를 Allow/Deny 함으로 네트워크 공격을 차단하는 시스템이다. 침입탐지시스템/침입차단시스템은 네트워크를 통한 트래픽을 분석하여 유해한 트래픽이 있는지 실시간으로 검사를 하며 침입이 탐지되었을 경우 해당하는 내용을 관리자에게 알려준다[1].

양환석[2]의 연구에서는 사물인터넷 환경에서의 악의적인 공격에 대하여 탐지 성능 향상을 위하여 듀얼침입탐지시스템 기법에 대하여 연구하였고 DLC 기법보다 듀얼침입탐지시스템 기법이 우수한 성능 보였다는 것을 증명하였다. 석진욱[3]외 연구에서는 Alpha 버전의 Snort와 Suricata를 성능에 대한 비교 연구를 하였고 Snort의 Multi Threads의 기술은 아직 미완성단계 이지만 Snort의 성능이 조금 더 좋았다는 것을 증명하였다. 유상규[4] 연구에서는 다중 큐를 이용하여 Suricata 시스템을 구현하여 성능 비교를 하였고 Single Thread Snort, Multi Threads Suricata, 다중 큐를 구현한 Suricata 순으로 성능 개선이 이루어짐을 증명하였다.

현재 진행되고 있는 연구를 보면, 침입탐지시스템/침입방지시스템의 알고리즘을 개선하는 연구가 주를 이루고 있으며, 오탐률을 개선하는 연구가 많이 이루어지고 있다.

본 연구의 차별화는 전세계 운영체제 1위의 점유율의 Windows7과 Ubuntu에 Suricata를 적용 및 설치하여 어떠한 운영체제 환경에서 Suricata가 더욱 효과적인 성능을 내는지에 대하여 제언하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 침입탐지시스템, 침입방지시스템 Windows7과 Ubuntu, Suricata를 정리하고, 3장에서는 실험환경 및 방법, 4장에서는 성능평가 및 비교를 분석한다. 마지막으로 5장에서는 연구결과, 시사점, 향후 연구에 대하여 논의하였다.

Ⅱ. 관련연구

2.1 침입탐지시스템/침입방지시스템

2.1.1 침입탐지시스템/침입방지시스템의 개념

침입탐지시스템/침입방지시스템은 네트워크의 패킷을 분석하여 취약점 공격, 비정상접근, 권한상승, 악성코드 등의 공격을 탐지하거나 방지하는 역할을 한다. 이러한 공격들을 찾아내는 것이 침입탐지시스템이고 탐지하여 차단하는 기능을 하는 것이 침입방지 혹은 침입차단시스템이다[5]. 기업이나 공공기관들은 유지보수의 편의성을 위하여 대게 상용화 국·내외 침입방지/탐지시스템 솔루션을 사용하고 있다. 하지만 이러한 솔루션의 구축이 부담스러운 중소기업이나 연구의 목적으로 사용이 가능한 오픈소스 침입탐지시스템/침입방지시스템인 Snort와 Multi Threads 기반 Suricata를 사용하고 있다.

2.1.2 침입방지시스템/침입탐지시스템의 특징

침입탐지시스템/침입방지시스템의 기법은 다섯 가지의 방법으로 나눌 수 있는데 화이트/블랙리스

트, 오용 탐지, 이상 탐지, 하이브리드 탐지, 데이터 마이닝 탐지의 방법이 있다. 오용 탐지는 일반적으로 해킹 패턴이라고 정한 모델과 일치할 경우 탐지를 하는 것을 말하고 이러한 정하여진 패턴이 아닌 것을 탐지하는 것을 이상 탐지라고 한다[6]. 하이브리드 탐지는 오용 탐지와 이상 탐지 방법을 결합하여 탐지 기능을 향상 시킨 탐지 방법으로 호스트에 에이전트를 설치하여 오용 탐지 및 이상 탐지 데이터를 네트워크 정보와 결합하여 탐지하는 방법이다[7]. 데이터마이닝 탐지는 데이터의 중요한 이벤트, 이상 징후, 구조의 패턴 변경에 따른 침입 탐지의 프로세스 개선에 도움을 주며 침입 탐지에 사용하는 룰셋 및 이상치 탐지의 군집을 분류하여 알려지지 않은 해킹 행위를 분석하여 예측을 한다[8]. 마지막으로 화이트/블랙리스트 탐지는 도메인 정보와 IP 정보를 사용하여 리스트를 작성하는데 그 리스트를 기반으로 탐지 및 차단을 하거나 예외처리를 하는 방법이다[7].

2.2 Suricata

2.2.1 Suricata의 개념

2009년 Suricata는 미국 국토안보부에서 자금 지원을 하여 OISF를 설립, 2010년 침입방지/탐지시스템인 Suricata를 개발하였다. 컴퓨터의 하드웨어의 발전과 네트워크 트래픽의 폭발적인 증가는 Snort의 Single Threads 방식은 너무 큰 약점으로 지적되었고, 이에 Suricata를 개발하였다. Snort의 룰셋 호환과 성능향상 및 기능의 추가는 Suricata의 매력이라 할 수 있겠다[9].

2.2.2 Suricata의 특징

Suricata의 특징은 앞서 설명하였듯 대용량의 트래픽을 Single Thread 방식의 Snort로는 탐지의 한계가 있었고 Multi Threads 방식의 Suricata는 더욱 향상된 기술로 대용량의 트래픽 탐지를 지원할 수 있다. 또한 기존에 사용하던 Snort의 룰셋 호환이 가능하고, NVDIA사의 그래픽카드의 GPU를 사용하여 탐지 성능을 향상 시킬 수 있다. LUA 스크립트로 시그니처를 만들 수 있는 장점이 있고 Snort에는 없던 몇 가지의 탐지 기능이 추가되었다[5].

2.3 Windows7

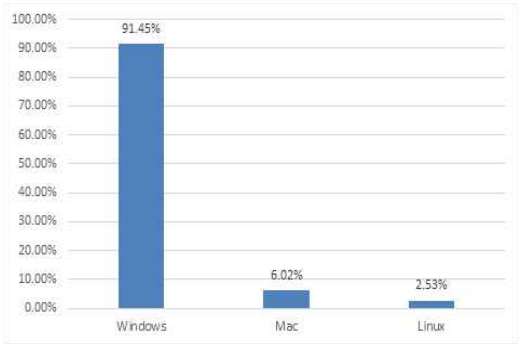
2.3.1 Windows7의 개념

1981년 PC-DOS를 시작으로 1982년 MS-DOS 1.25버전의 운영체제를 개발하였다. 1985년 Windows 1.0을 시작으로 기존 Command Line Interface 방식의 틀을 깨고 전문가 및 일반 사용자들도 사용하기 쉬운 Graphic User Interface 방식의 Windows는 3.0, 3.1, Windows 95. 98, me를 거쳐 XP, VISTA, Windows7, 8, 현재 가장 최신 버전인 Windows10까지 개발을 하였다. 또한 Windows Server 제품군도 현재 많은 사용자들이 이용하고 있다. Windows7은 2009년 출시를 하였으며 64비트는 RAM용량을 최대 192GB까지 지원한다. Windows7의 편의성 및 안정성으로 인하여 Windows10이 나온 지금도 많은 사용자들이 사용하고 있는 운영체제이다[10]. <그림 1>에서 보듯이 2017년 7월 현재 Windows 점유율은 91.45%, Mac OS X는 6.02, Linux는 2.53%로 나타났다[11]. <그림 2>는 2017년 7월 현재 Windows 버전 별 점유율을 나타냈으며 Windows 2000은 0.01%, Windows XP는 6.10%, Windows Vista는 0.51%, Windows 7은 48.91%, Windows 8은 1.42%, Windows 8.1은 6.48%,

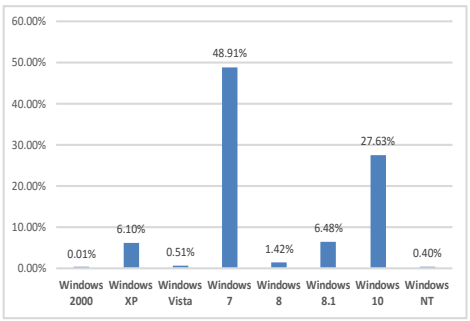
Windows 10은 27.63%, Windows NT 0.40%로 집계되었으며 아직도 Windows 7의 점유율은 거의 50%의 점유율을 유지하고 있으며 아직도 많은 사용자들이 사용하는 운영체제이다[12].

2.3.2 Windows7의 특징

Windows7의 여러 가지 특징이 있는데 멀티터치 포인트, 터치스크린, GPS, 주변 환경을 감지할 수 있는 센서 등의 기능을 가진 운영체제이다. 또 다른 특징으로 문제 해결 플랫폼이 있다. Windows7은 여타 마이크로소프트의 운영체제보다 디스크 용량을 작게 차지하며 더 적은 메모리를 사용한다. 또한 프로세스를 최대 256개까지 지원한다[13].



<그림 1> 운영체제 별 점유율[11]



<그림 2> Windows 버전별 점유율[12]

2.4 Ubuntu

2.4.1 Ubuntu의 개념

Ubuntu는 2004년 마크 셔틀워스에 의하여 시작된 오픈소스 리눅스 프로젝트 이름이고 Debian GNU/Linux 기반으로 만들어졌다. 여타 다른 리눅스기반 운영체제에 비하여 일반 사용자에게 사용하기 쉬운 오픈소스 운영체제를 무료로 제공하려고 노력하고 있다[14].

2.4.2 Ubuntu의 특징

Ubuntu는 첫째 다른 리눅스 운영체재와 다르게 설치과정이 복잡하지 않다. 둘째 우분투 및 데비안 기반 운영체재들은 소프트웨어 관리 도구로 APT(Advaned Package Tool), 계열의 유틸리티를 사용한다. 셋째 root 권한을 설치과정에서 만들지 못하도록 하였다. 만약 root 권한으로 실행을 해야 하는 명령어의 경우 sudo를 실행하고자 하는 명령어 앞에 사용하여 이 명령어를 실행 후 패스워드를 입력하여 명령어 실행을 하고 있다[15].

오픈소스 운영체제답게 사용하기 쉽고 변경 및 설계가 자유로워 오픈소스의 특징을 이용하여 여러 분야에서 사용하고 있는 운영체제이다. Ubuntu의 가장 큰 특징 보안성을 꼽을 수 있다. 바이러스나 악성코드 등 악성행위에 대하여 Windows보다 안전하다고 볼 수 있다. 둘째로는 개인 용도의 서버 구축이 있다. Ubuntu는 서버 구축을 초보자도 쉽고 간단하게 할 수 있다[14, 16].

Ⅲ. Suricata를 적용한 Ubuntu와 Windows7

3.1 실험환경 및 방법

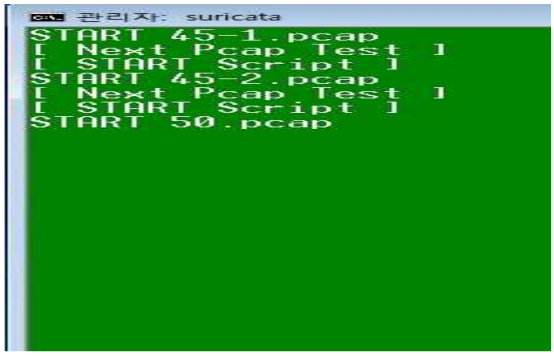
실험을 위하여 DELL사의 C1100 서버를 이용하였고 CPU는 Intel Xeon CPU L5520 2.27GHz*2 총 8core 16threads RAM은 72GB, 삼성 SSD EVO 250Gb*4를 사용하였다. 성능 측정 실험에 사용된 운영체제는 Windows7 SP1, Ubuntu는 16.04.3 LTS 버전을 설치하였다. 실험에 사용된 Suricata 버전은 Windows7에는 Suricata 4.0.0 32bit, Ubuntu에는 Suricata 4.0.0을 설치하였다.

3.2 Suricata 테스트

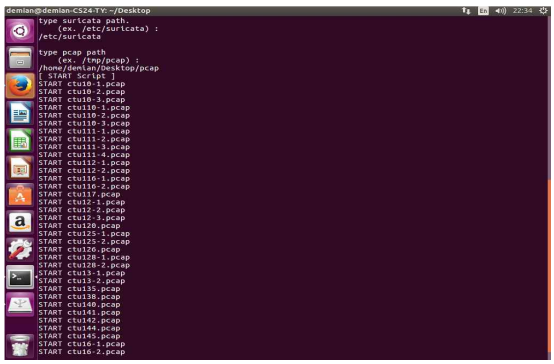
Suricata 설치 후 모든 설정은 Default 설정으로 진행하였으며, 룰셋의 개수는 11902로 동일하게 맞추었다. 이 때 사용된 알고리즘은 Aho-Corasick 알고리즘을 사용하였다. 성능 측정에 사용된 Pcap파일의 경우 다양한 용량 및 악성행위 트래픽을 포함하고 있는 CTU 홈페이지 내 Pcap 파일을 다운 받아 실험하였고 실험에서 사용 한 파일의 개수는 71개였다.

Ubuntu와 Windows7 모두 같은 Pcap파일 실행 명령어를 사용하였다. suricata -v -r c:\pcap폴더\pcap파일.Pcap 명령어를 사용하여 실험을 진행한다. -v -r 옵션을 넣어주어 offline mode에서 Pcap 파일을 실행 시킨다. Pcap 파일 실행 후 CPU core, 룰셋의 정보, 실행 후 총 걸린 시간, 로그 파일 정보 등 세부 사항이 기록되어 있다.

<그림 3>은 Windows 배치파일 Suricata 실험 자동화를 구현한 모습이고, <그림 4>는 Ubuntu에서 Suricata 실험 자동화를 구현한 모습이다, 각각의 운영체제에서 Suricata를 Pcap파일 실행 후 완료 시간을 측정하였고 총 3회 실험을 진행하여 평균값을 가지고 비교 하였다. 마지막으로 하나의 Pcap을 실행 후 생기는 로그 파일들은 삭제 후 다음 Pcap파일을 실행하는 배치 파일 및 쉘스크립트를 작성하여 실행하였다.



<그림 3> Windows7에서의 Suricata 배치파일 동작 화면



<그림 4> Ubuntu에서의 Suricata 스크립트 동작 화면

Ⅳ. 성능평가 및 비교

<그림 5>의 실험결과 그래프는 10개의 Pcap파일을 세 차례 실행시켜 나온 평균값으로 Windows7의 평균값은 99094.49초 걸렸으며,

Ubuntu는 305.88초가 걸렸다. 세부적으로 살펴보면 409MB Pcap 파일은 Windows7에서 2555.71초가 걸렸으며, Ubuntu에서는 7.73초가 걸렸다. 2.48GB Pcap 파일은 Windows7에서 2560.08초가 걸렸으며, Ubuntu에서는 78.57초가 걸렸다. 2.79GB Pcap 파일은 Windows7에서 51044.08초가 걸렸으며, Ubuntu에서는 112.96초가 걸렸다. 747MB Pcap 파일은 Windows7에서 7619.03초가 걸렸으며, Ubuntu에서는 25.73초가 걸렸다. 989MB Pcap 파일은 Windows7에서 8122.63초가 걸렸으며, Ubuntu에서는 38.58초가 걸렸다. 120MB Pcap 파일은 Windows7에서 818.93초가 걸렸으며, Ubuntu에서는 6.42초가 걸렸다. 1.32GB Pcap 파일은 Windows7에서 8112.88초가 걸렸으며, Ubuntu에서는 40.99초가 걸렸다. 183MB Pcap 파일은 Windows7에서 1099.05초가 걸렸으며, Ubuntu에서는 6.21초가 걸렸다. 1.59GB Pcap 파일은 Windows7에서 7665.29초가 걸렸으며, Ubuntu에서는 73.21초가 걸렸다. 1.17GB Pcap 파일은 Windows7에서 9496.81초가 걸렸으며, Ubuntu에서는 22.45초가 걸렸다.

<그림 6>의 실험결과 그래프는 10개의 Pcap파일을 세 차례 실행시켜 나온 평균값으로 Windows7의 평균값은 194484.72초 걸렸으며, Ubuntu는 372.05초가 걸렸다. 370MB Pcap 파일은 Windows7에서 51.33초가 걸렸으며, Ubuntu에서는 3.31초가 걸렸다. 316MB Pcap 파일은 Windows7에서 36.73초가 걸렸으며, Ubuntu에서는 2.27초가 걸렸다. 149MB Pcap 파일은 Windows7에서 2452.26초가 걸렸으며, Ubuntu에서는 8.80초가 걸렸다. 222MB Pcap 파일은 Windows7에서 3707.08초가 걸렸으며, Ubuntu에서는 10.33초가 걸렸다. 186MB Pcap 파일은 Windows7에서 2768.40초가 걸렸으며, Ubuntu에서는 9.36초가 걸렸다. 13.2GB Pcap 파일은 Windows7에서 173482.52초가 걸렸으며, Ubuntu에서는 302.44초가 걸렸다. 470MB Pcap 파일은 Windows7에서 6190.67초가 걸렸으며, Ubuntu에서는 15.99초가 걸렸다. 490MB Pcap 파일은 Windows7에서 5698.07초가 걸렸으며, Ubuntu에서는 16.01초가 걸렸다. 254MB(win18) Pcap 파일은 Windows7에서 48.01초가 걸렸으며, Ubuntu에서는 1.77초가 걸렸다. 254MB(win19) Pcap 파일은 Windows7에서 49.66초가 걸렸으며, Ubuntu에서는 1.76초가 걸렸다.

<그림 7>의 실험결과 그래프는 10개의 Pcap파일을 세 차례 실행시켜 나온 평균값으로 Windows7의 평균값은 199936.06초 걸렸으며, Ubuntu는 488.43초가 걸렸다. 437MB Pcap 파일은 Windows7에서 6060.92초가 걸렸으며, Ubuntu에서는 12.62초가 걸렸다. 116MB Pcap 파일은 Windows7에서 23.08초가 걸렸으며, Ubuntu에서는 1.29초가 걸렸다. 883MB Pcap 파일은 Windows7에서 2407.67초가 걸렸으며, Ubuntu에서는 9.36초가 걸렸다. 281MB Pcap 파일은 Windows7에서 511.03초가 걸렸으며, Ubuntu에서는 9.75초가 걸렸다. 133MB Pcap 파일은 Windows7에서 113.49초가 걸렸으며, Ubuntu에서는 151초가 걸렸다. 113MB Pcap 파일은 Windows7에서 15.94초가 걸렸으며, Ubuntu에서는 1.27초가 걸렸다. 3.09GB Pcap 파일은 Windows7에서 86927.61초가 걸렸으며, Ubuntu에서는 191.92초가 걸렸다. 14.0GB Pcap 파일은 Windows7에서 49324.10초가 걸렸으며, Ubuntu에서는 128.20초가 걸렸다. 8.17GB Pcap 파일은 Windows7에서 28719.37초가 걸렸으며, Ubuntu에서는 73.84초가 걸렸다.

<그림 8>의 실험결과 그래프는 10개의 Pcap파일을 세 차례 실행시켜 나온 평균값으로 Windows7의

평균값은 133040.75초 걸렸으며, Ubuntu는 339.47초가 걸렸다. 3.96GB Pcap 파일은 Windows7에서 16064.66초가 걸렸으며, Ubuntu에서는 36.01초가 걸렸다. 14.0GB Pcap 파일은 Windows7에서 54578.63초가 걸렸으며, Ubuntu에서는 129.64초가 걸렸다. 8.19GB Pcap 파일은 Windows7에서 28886.97초가 걸렸으며, Ubuntu에서는 73.73초가 걸렸다. 3.97GB Pcap 파일은 Windows7에서 15942.48초가 걸렸으며, Ubuntu에서는 38.77초가 걸렸다. 1.81GB Pcap 파일은 Windows7에서 7203.07초가 걸렸으며, Ubuntu에서는 20.94초가 걸렸다. 1.82GB Pcap 파일은 Windows7에서 7659.20초가 걸렸으며, Ubuntu에서는 20.98초가 걸렸다. 346MB Pcap 파일은 Windows7에서 319.10초가 걸렸으며, Ubuntu에서는 3.27초가 걸렸다. 122MB Pcap 파일은 Windows7에서 200.30초가 걸렸으며, Ubuntu에서는 0.98초가 걸렸다. 212MB Pcap 파일은 Windows7에서 15.26초가 걸렸으며, Ubuntu에서는 1.67초가 걸렸다. 1.04GB Pcap 파일은 Windows7에서 2171.08초가 걸렸으며, Ubuntu에서는 13.48초가 걸렸다.

<그림 9>의 실험결과 그래프는 10개의 Pcap파일을 세 차례 실행시켜 나온 평균값으로 Windows7의 평균값은 26966.81초 걸렸으며, Ubuntu는 119.10초가 걸렸다. 281MB Pcap 파일은 Windows7에서 108.75초가 걸렸으며, Ubuntu에서는 1.56초가 걸렸다. 109MB Pcap 파일은 Windows7에서 285.14초가 걸렸으며, Ubuntu에서는 1.42초가 걸렸다. 649MB Pcap 파일은 Windows7에서 707.58초가 걸렸으며, Ubuntu에서는 4.79초가 걸렸다. 6.0GB Pcap 파일은 Windows7에서 8057.81초가 걸렸으며, Ubuntu에서는 41.95초가 걸렸다. 2.44GB Pcap 파일은 Windows7에서 3679.28초가 걸렸으며, Ubuntu에서는 17.24초가 걸렸다. 249MB Pcap 파일은 Windows7에서 41.86초가 걸렸으며, Ubuntu에서는 1.93초가 걸렸다. 1.27GB Pcap 파일은 Windows7에서 2490.98초가 걸렸으며, Ubuntu에서는 9.42초가 걸렸다. 764MB Pcap 파일은 Windows7에서 1902.40초가 걸렸으며, Ubuntu에서는 7.04초가 걸렸다. 4.42GB Pcap 파일은 Windows7에서 8713.77초가 걸렸으며, Ubuntu에서는 29.22초가 걸렸다. 655MB Pcap 파일은 Windows7에서 979.23초가 걸렸으며, Ubuntu에서는 4.53초가 걸렸다.

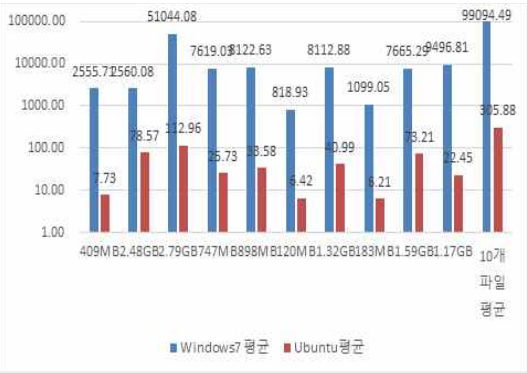
<그림 10>의 실험결과 그래프는 10개의 Pcap파일을 세 차례 실행시켜 나온 평균값으로 Windows7의 평균값은 24731.77초 걸렸으며, Ubuntu는 126.30초가 걸렸다. 1.29GB Pcap 파일은 Windows7에서 1903.11초가 걸렸으며, Ubuntu에서는 11.46초가 걸렸다. 2.28GB Pcap 파일은 Windows7에서 3398.90초가 걸렸으며, Ubuntu에서는 14.91초가 걸렸다. 1.78GB Pcap 파일은 Windows7에서 9853.81초가 걸렸으며, Ubuntu에서는 34.64초가 걸렸다. 3.85GB Pcap 파일은 Windows7에서 721.74초가 걸렸으며, Ubuntu에서는 25.20초가 걸렸다. 2.11GB Pcap 파일은 Windows7에서 3162.81초가 걸렸으며, Ubuntu에서는 15.66초가 걸렸다. 105MB Pcap 파일은 Windows7에서 122.25초가 걸렸으며, Ubuntu에서는 1.02초가 걸렸다. 186MB Pcap 파일은 Windows7에서 580.40초가 걸렸으며, Ubuntu에서는 4.33초가 걸렸다. 97.0MB Pcap 파일은 Windows7에서 385.56초가 걸렸으며, Ubuntu에서는 3.53초가 걸렸다. 305MB Pcap 파일은 Windows7에서 43.27초가 걸렸으며, Ubuntu에서는 3.19초가 걸렸다. 1.65GB Pcap 파일은 Windows7에서 4559.92초가 걸렸으며, Ubuntu에서는 12.35초가 걸렸다.

<그림 11>의 실험결과 그래프는 11개의 Pcap파일을 세 차례 실행시켜 나온 평균값으로 Windows7의 평균값은 10770.47초 걸렸으며, Ubuntu는 45.88

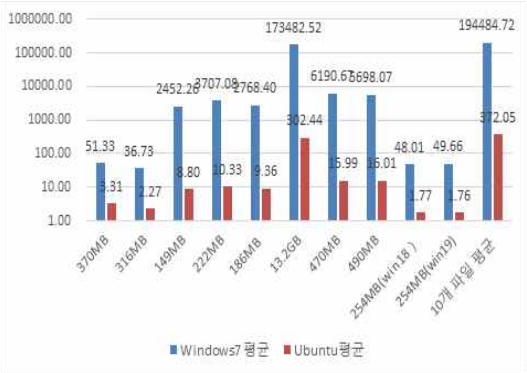
초가 걸렸다.

107MB Pcap 파일은 Windows7에서 60.47초가 걸렸으며, Ubuntu에서는 0.70초가 걸렸다. 1.17GB Pcap 파일은 Windows7에서 2343.69초가 걸렸으며, Ubuntu에서는 7.83초가 걸렸다. 336MB Pcap 파일은 Windows7에서 109.52초가 걸렸으며, Ubuntu에서는 4.89초가 걸렸다. 1.08GB Pcap 파일은 Windows7에서 1023.89초가 걸렸으며, Ubuntu에서는 13.11초가 걸렸다. 497MB Pcap 파일은 Windows7에서 4.11초가 걸렸으며, Ubuntu에서는 2.73초가 걸렸다. 1.37GB Pcap 파일은 Windows7에서 3224.94초가 걸렸으며, Ubuntu에서는 24.77초가 걸렸다. 221MB Pcap 파일은 Windows7에서 174.52초가 걸렸으며, Ubuntu에서는 2.02초가 걸렸다. 323MB Pcap 파일은 Windows7에서 343.76초가 걸렸으며, Ubuntu에서는 3.02초가 걸렸다. 236MB Pcap 파일은 Windows7에서 1399.42초가 걸렸으며, Ubuntu에서는 5.03초가 걸렸다. 283MB Pcap 파일은 Windows7에서 947.39초가 걸렸으며, Ubuntu에서는 5.76초가 걸렸다. 273MB Pcap 파일은 Windows7에서 1138.75초가 걸렸으며, Ubuntu에서는 6.02초가 걸렸다.

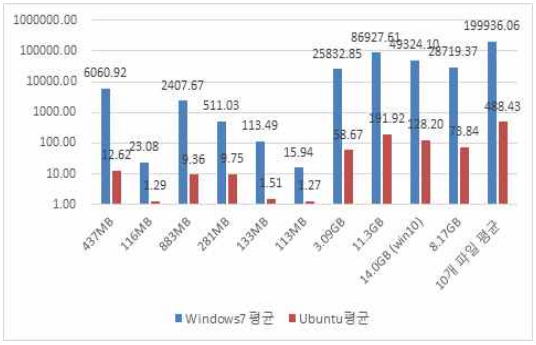
마지막으로 <그림 12>의 실험결과 그래프는 각각의 그래프에서 나온 Pcap파일 평균 실험값을 합친 결과로 Windows7은 689025.06초, Ubuntu는 1827.11초로 압도적으로 Ubuntu에서의 성능이 좋았다. 그이유로는 <그림 13>을 보면 Ubuntu에서 Suricata가 동작 시 모든 멀티코어 환경에서 균일한 사용률을 보였다. 하지만 <그림 14>의 Windows7에서는 전체 CPU의 사용이 6퍼센트 정도의 사용률을 보이며 불균등하게 사용됨으로써 이런 차이가 발생한 것으로 보인다. 또한 Windows7의 Suricata 32bit버전으로 제공되기 때문에 실험에 사용했던 72GB의 대용량의 RAM을 효과적으로 사용하지 못하기 때문에 이러한 성능의 차이가 발생했었을 것으로 보인다.



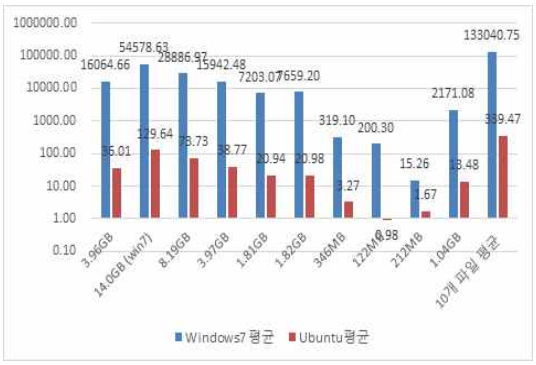
<그림 5> 첫 번째 10개 Pcap 파일 실행 후 결과 값



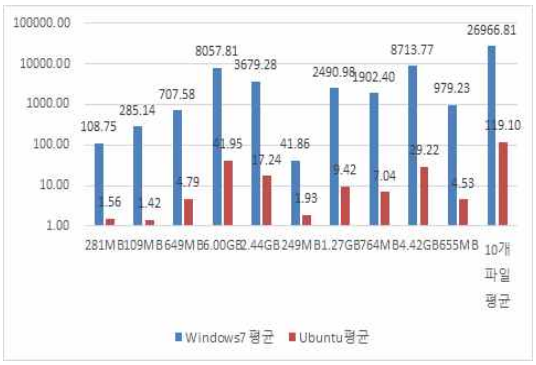
<그림 6> 두 번째 10개 Pcap 파일 실행 후 결과 값



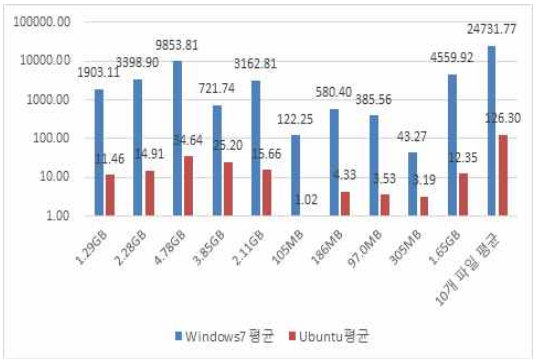
<그림 7> 세 번째 10개 Pcap 파일 실행 후 결과 값



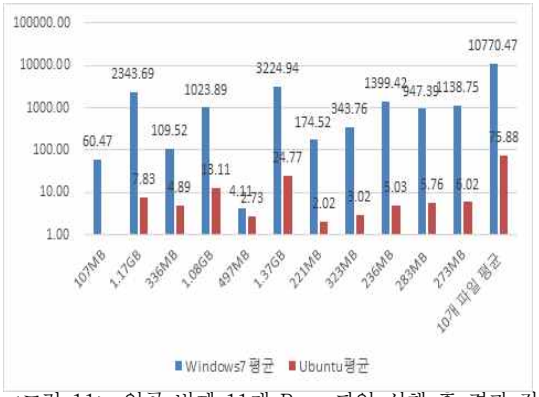
<그림 8> 네 번째 10개 Pcap 파일 실행 후 결과 값



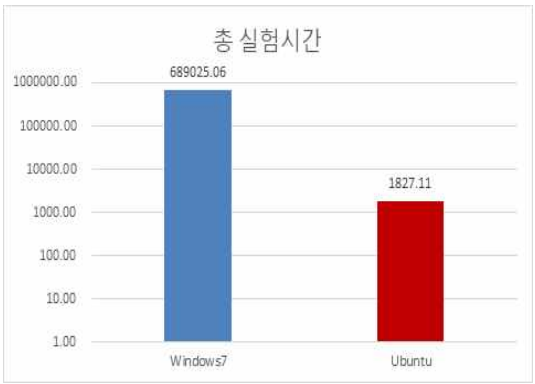
<그림 9> 다섯 번째 10개 Pcap 파일 실행 후 결과 값



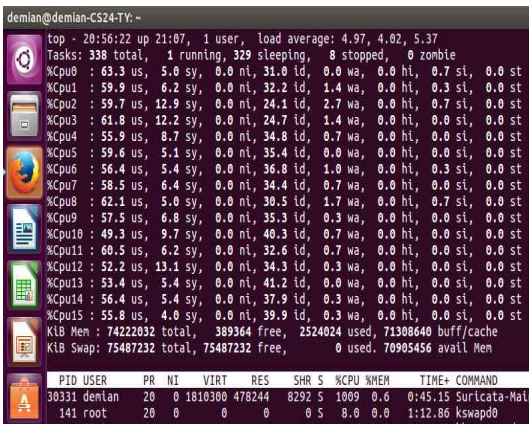
<그림 10> 여섯 번째 10개 Pcap 파일 실행 후 결과 값



<그림 11> 일곱 번째 11개 Pcap 파일 실행 후 결과 값



<그림 12> 총 71개 Pcap 파일 총 평균 실험시간



<그림 13> Ubuntu CPU 사용률

<그림 14> Windows7 CPU 사용률

Ⅳ. 결론

본 논문에서는 오픈소스 침입탐지시스템/침입방지시스템인 Suricata를 적용한 Windows7과 Ubuntu 성능 비교 분석 실험을 증명하였다.

<그림 12>에서 나타났듯 71개 Pcap파일을 3번 돌려 나온 평균이 Windows7은 689025.06초, Ubuntu는 1827.11초로 나왔으며 이는 Ubuntu에서의 성능이 Windows7보다 월등한 성능을 보였다. 이러한 결과는 Suricata를 적용한 Ubuntu에서 멀티코어 CPU 사용을 더욱 효과적으로 사용했기 때문에 Ubuntu에서의 실험결과가 더욱 의미 있게 나온 것으로 해석된다.

이 실험을 통하여 나날이 발전하는 컴퓨터 하드웨어 환경에서 오픈소스 침입탐지시스템/침입방지시스템인 Suricata를 설치 및 운영에 있어 어떤 운영체제에서 최적화된 성능을 보이는지 실험을 통하여 증명하였다.

Suricata에서 Pcap파일을 실행을 하는 방법 실험을 진행하였다. 본 연구의 한계점으로는 실제 사용하는 네트워크 구성에 Suricata를 설치하여 실험을 진행을 하지 못하였다는 점이 본 논문의 한계점이라 할 수 있다. 향후 다양한 운영체제에서의 Suricata를 설치 및 적용하여 연구를 진행할 필요가 있겠고 토마호크와 같은 공격트래픽 생성기를 사용하여 실제 네트워크 및 보안시스템 구축 후 실험을 진행할 필요가 있을 것이다.

참고문헌

[1] 기원호, “침입탐지시스템을 위한 TCAM 기반 패턴 매칭 하드웨어 구조,” 연세대학교 석사학위논문, 2007.

[2] 양환석, “사물인터넷 환경에서 안전성과 신뢰성 향상을 위한 Dual-IDS 기법에 관한 연구,” 디지털산업정보학회 논문지, 제13권, 제1호, 2017, pp.49-57.

[3] 석진욱·최문석·김지명, “오픈소스 IDS IPS Snort와 Suricata의 탐지 성능에 대한 비교 연구,” 디지털산업정보학회 논문지, 제12권, 제1호, 2016, pp.89-95.

[4] 유상규, “멀티코어 환경에서 다중 큐를 이용한 멀티 스레드 기반 IPS 시스템의 설계 및 구현,” 서강대학교, 석사학위논문, 2013.

[5] 장상근, 네트워크 보안 시스템 구축과 보안 관제, 한빛출판사, 2015, p.44.

[6] 이용봉, “최적적응과 대응능력을 위한 침입탐지 시스템 해석에 대한 연구,” 한밭대학교 정보통신전문대학원, 박사학위논문, 2016.

[7] 손동식, “침입방지시스템(IPS)에서 알려지지 않은 사이버공격 탐지에 관한 순환탐지 및 방어 아키텍처 설계,” 건국대학교 정보통신대학원 석사학위논문, 2015.

[8] 박종명, “클러스터링기법을 이용한 실시간 네트워크 침입탐지 시스템의 설계,” 서울시립대학교 대학원 석사학위논문, 2005.

[9] 박우진 외, “Suricata의 Multi-Threading 효율성에 관한 실험적 연구,” 한국통신학회, 한국통신학회 학술대회논문집, Vol.2015, No.06, 2015, pp.874-875.
[10] 우재남, 뇌를 자극하는 Windows Server 2008, 한빛미디어(주), 2011, pp.78-79.
[11] 넷마켓쉐어 데스크탑 운영체제 마켓 쉐어, http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8&qpcustomd=0
[12] 넷마켓쉐어 데스크탑 운영체제 마켓 쉐어, http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0
[13] 김종원, C++와 C# 예제로 쉽게 배우는 윈도우7 프로그래밍, ㈜갑우문화사, 2011, pp.29-31.
[14] 이준희, 우분투 투게더, 인사이트, 2015, pp.4-21.
[15] 크리스토퍼 니거스, 프랑소와캉, Ubuntu Linux Toolbox, ㈜지앤선, 2009, pp.5-6.
[16] 이준희, 웰컴 투 우분투, 인사이트, 2010, pp.19-30.

■ 저자소개 ■

석 진 욱
(Seok Jinug)

2017년 12월~현재 시큐어원 보안컨설턴트
2016년 8월 성균관대학교 정보통신대학원 정보보호학과 (공학석사)

관심분야 : 네트워크 보안, 정보보호관리체계, 개인정보보호관리체계
E-mail : jinugi6@naver.com

김 지 명
(Kim Jimyung)

2017년 12월~현재 한국인터넷진흥원 공인전자주소팀 선임연구원
2016년 2월 성균관대학교 정보통신대학원 정보보호학과 (공학석사)

관심분야 : 네트워크 보안, 사물인터넷 보안
E-mail : bugeking@skku.edu

최 문 석
(Choi Moonseok)

2017년 12월~현재 우리에프아이에스 정보보안부
2016년 2월 성균관대학교 정보통신대학원 정보보호학과 (공학석사)

관심분야 : 네트워크 보안, 정보보안
E-mail : msoki@skku.edu

논문접수일 : 2017년 09월 15일
수 정 일 : 2017년 12월 02일
게재확정일 : 2017년 12월 05일