JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 19, No. 4, pp. 711-717, Apr. 2018

윈도우즈 커널 기반 침입탐지시스템의 탐지 성능 개선

김 의 탁[1,2] · 류 근 호[1*]
[1,2]하우리 기술연구소
[1]충북대학교 전기전자정보컴퓨터학부

An Improved Detection Performance for the Intrusion Detection System based on Windows Kernel

Eui-Tak Kim[1,2] · Keun Ho Ryu[1*]

[1,2]Hauri Technical Laboratory, Seoul, Korea

[1]Database/Bioinformatics Laboratory, School of Electrical and Computer Engineering, Chungbuk National University, Cheongju, Korea

[요 약]

컴퓨터와 네트워크의 비약적인 발전은 다양한 정보 교환을 쉽게 하였다. 하지만, 그와 동시에 다양한 위험 요소를 발생시켜 악의적 목적을 가진 사용자와 그룹은 취약한 시스템을 대상으로 공격을 하고 있다. 침입탐지시스템은 네트워크 패킷 분석을 통해 악의적인 행위를 탐지한다. 하지만, 많은 양의 패킷을 짧은 시간 내에 처리해야 하는 부담이 있다. 따라서, 이 문제를 해결하기 위하여 우리는 User Level에서 동작하는 네트워크 침입탐지시스템의 탐지 성능 향상을 위해 Kernel Level에서 동작하는 시스템을 제안한다. 실제로, kernel level에서 동작하는 네트워크 침입탐지시스템을 구현함으로써 패킷 분석 및 탐지 성능을 향상함을 확인하였다.

[Abstract]

The breakthrough in computer and network has facilitated a variety of information exchange. However, at the same time, malicious users and groups are attacking vulnerable systems. Intrusion Detection System(IDS) detects malicious behaviors through network packet analysis. However, it has a burden of processing a large amount of packets in a short time. Therefore, in order to solve these problem, we propose a network intrusion detection system that operates at kernel level to improve detection performance at user level. In fact, we confirmed that the network intrusion detection system implemented at kernel level improves packet analysis and detection performance.

색인어 : 정보보호, 침입탐지시스템, 침입방지시스템, 티디아이, 엔디아이에스

Key word : Information Protection, Intrusion Detection System, Intrusion Protection System, Transport Driver Interface, Network Driver Interface Specification

http://dx.doi.org/10.9728/dcs.2018.19.4.711

Received 22 March 2018; **Revised** 02 April 2018
Accepted 21 April 2018

***Corresponding Author; Keun Ho Ryu**

Tel: +82-43-267-2254
E-mail: khryu@dblab.chungbuk.ac.kr

Ⅰ. 서 론

IDS(Intrusion Detection System)는 일반적으로 시스템의 비정상적인 사용, 오용 및 남용 등을 실시간으로 탐지하는 보안솔루션으로, 전통적인 침입차단시스템(Firewall)이 탐지할 수 없는 모든 종류의 악의적인 네트워크 트래픽 및 사용을 탐지하기 위해 동작한다. IDS는 탐지 영역에 따라 네트워크 침입탐지시스템(Network Intrusion Detection System, NIDS)과 호스트 침입탐지시스템(Host Intrusion Detection System, HIDS)으로 구분된다[1]. NIDS는 네트워크 트래픽을 검사하고[2] 여러 호스트를 관찰하여 침입을 식별하는 독립된 플랫폼으로, 쉽게 설치할 수 있고, 강력한 탐지 기능이 있으므로 널리 사용된다. 반면, HIDS는 호스트에서 시스템 콜, 애플리케이션 로그, 사용자 로그인과 로그아웃 및 파일 시스템의 수정사항 등을 감시한다.

IDS는 탐지기법에 따라 오용(Misuse)기반 탐지와 비정상(Anomaly)기반 탐지로 구분된다. 오용기반 탐지기법은 알려진 악의적인 공격 또는 의도치 않은 동작에 대한 시그니쳐 정보를 바탕으로 공격을 탐지하기 때문에 Zero-day 공격을 탐지하지 못한다. 비정상기반 탐지기법은 정상적인 동작 및 행위로 정의되지 않은 모든 상황을 비정상으로 간주하여 탐지하기 때문에 오용 탐지기법과 달리, Zero-day 공격에 대한 탐지에 적합하다. 그러나, 정상 동작 및 행위를 판단할 수 있는 근거를 정의하기 위해 방대한 데이터가 요구되고, 학습을 통한 모델 수립에 어려움이 있다[3][4].

대부분의 IDS는 오용기반 탐지기법을 사용하고 있고, 이를 위해 시그니쳐 기반의 탐지규칙을 활용하여 침입 여부를 판별한다. 시그니쳐 기반 침입탐지는 악의적인 공격에 대한 정확한 정보를 갖고 있어 즉시 탐지가 가능하지만, 시그니쳐 규칙들의 계속되는 증가와 이에 따른 느린 탐지 속도, 오탐(fault-positive), DoS 공격에 대한 취약점들이 지적됐다[5][6]. 특히, 탐지규칙을 통합적으로 효율적으로 생성하고 관리하는 것이 매우 어려우므로 탐지규칙들이 중복되거나 유사한 탐지규칙들이 많고, 이로 인해 침입탐지시스템의 성능이 크게 저하되고 있다[7]. 이에 따라, 시그니쳐 기반의 탐지규칙을 효과적으로 수행하기 위해 시그니쳐를 고성능으로 처리하려는 다양한 연구가 시도되고 있으며[8], 기존 보안 시스템에서 사용하는 다양한 탐지규칙에 대해 정형화된 하나의 규칙을 개발함으로써 이기종 시스템에 이식하는데 소요되는 불필요한 학습 비용을 줄이고자 하는 연구도 수행되고 있다[9].

본 연구는 시스템의 User level에서 동작하는 NIDS의 패킷 처리 성능을 개선하고자, Kernel level에서 동작하는 네트워크 침입탐지시스템을 제안 및 구현하였다. 구현된 KNIDS(Kernel based Network Intrusion Detection System)는 NIDS의 기본 기능을 갖도록 설계하였고, User level에서 동작하는 NIDS와 달리 커널에서 동작함으로써 더욱 빠르게 많은 양의 패턴을 분석 및 탐지할 수 있도록 구현하였다.

Ⅱ. 관련 연구

2-1 Snort

Snort는 오픈소스 기반의 네트워크 침입탐지시스템(NIDS)으로 1998년 Martion Roesch에 의해 처음 개발되어 지금까지 가장 많이 사용하는 보안 시스템 중의 하나이다[10][11][12]. Snort는 오픈소스로 개발된 대표적인 네트워크 침입탐지시스템으로 사실상의 표준(de facto standard)으로 분류될 정도로 많은 사람이 사용하고 있다[3]. 실제로 보안 관련 소프트웨어 중 Snort는 sectools.org에서 제공하는 보안 시스템의 인기 순위 100위 중, 5위에 랭크 되어 있으며, 침입 탐지시스템 중에서는 가장 높은 순위의 보안 시스템으로 평가받고 있다[13].

Snort는 내부 규칙을 사용하여 네트워크 트래픽을 통해 수집된 패킷 정보를 비교 분석하여 악성코드의 전송 여부를 탐지할 수 있다[14]. 또한, 프로토콜 분석, 콘텐츠 비교 및 검색 기능을 하며 다양한 공격과 스캔(Stealth Port Scan, Buffer Overflow, SMB Scan, OS Finger Printing, CGI Attack etc)을 탐지할 수 있다[15]. Snort는 크게 세 가지 Mode(Network Intrusion Detection, Sniffer, Packet Logger)로 다양한 설정이 가능하며 Sniffer Mode에서는 네트워크에서 패킷을 읽어 들여 출력한다. Packet Logger Mode에서는 패킷을 저장매체에 로그 형태로 저장한다. Network Intrusion Detection Mode는 사용자에 의해 설정된 규칙을 갖고 네트워크 통신량을 관찰하며 분석을 수행한다. Snort의 내부 동작 단계의 구성을 살펴보면 패킷 스니퍼, 전처리기, 탐지 엔진, 로깅/경고, 로그 파일/데이터베이스로 구성되어 있다. 패킷 스니퍼는 네트워크로부터 패킷을 받아들이며 이 패킷을 전처리기에서 Snort의 탐지 엔진에 도달하기 전에 악의적인 패킷인지 올바른 패킷인지 구별하는 과정을 거친다. 전처리기를 통과한 패킷은 탐지 엔진을 통해 사용자에 의하여 설정된 보안규칙을 가지고 비교하여 악의적인 침입을 탐지한다. 마지막으로 탐지 엔진에서 나온 결과를 바탕으로 보안관리자에게 경보 및 로그를 기록하도록 하여 로그 파일과 데이터베이스의 형태로 탐지 기록을 저장하도록 한다[16].

2-2 User Level의 NIDS

네트워크 침입탐지시스템(NIDS)은 네트워크 통신량을 감시하여 서비스 거부 공격, 포트 스캔, 컴퓨터 크랙 등과 같은 악의적인 동작을 수행하는 패킷을 탐지하는 시스템이다. 이를 위해, NIDS는 모든 수신 패킷을 확인하고, 의심스러운 패턴을 찾는다. 예를 들어, 다수의 TCP 연결 요청을 통해 다양한 다른 포트를 연결하려는 시도가 발견될 경우, 이 행위는 누군가 포트 스캔을 시도하고 있다고 추측할 수 있다. 또한, 다양한 악의적인 셸 코드를 찾는 작업도 수행한다. 네트워크 침입탐지시스템은 종종 다른 시스템과 협동하여 동작한다. 예를 들어, 크래커에 의해 사용된 컴퓨터의 IP 리스트를 침입차단시스템의 블랙리스트에 업데이트하기도 한다. NIDS 제품으로는 Black ICE Defender, CheckPointe RealSecure, Cisco Secure IDS 등이 있으며, 오픈코드 기반의 Snort가 가장 많이 사용되고 있고[16], 엔

진도 타제품에 비교해 가벼우므로 본 연구의 비교 대상 제품으로 활용하였다.

[그림 1]과 같이, Snort는 pcap 라이브러리를 사용하여 promiscuous 모드로 설정된 네트워크 카드(NIC)를 통해서 네트워크 패킷을 캡처한다[17]. 이 모드의 장점은 모든 네트워크 패킷을 관찰할 수 있고 이를 통해, 네트워크 패킷을 방해하지 않고, IN/OUT 패킷을 도청할 수 있으며, 네트워크 성능 또한 크게 영향을 주지 않는다. 네트워크 패킷은 Promiscuous mode를 통해서 Kernel Level, User Level의 응용프로그램에 전달된다[18]. 패킷은 Kernel Level에서 User Level에게 버퍼를 카피하거나 커널 공간과 사용자 공간의 공유 메모리를 통해 전달된다.

Snort는 pcap을 통해 패킷을 수집할 때, Kernel Level의 BPF(Berkeley Packet Filter)를 통해서 패킷 정보를 얻는다. Snort는 수집된 패킷과 정의된 패턴을 비교하여 비정상적인 패킷이 있는지 확인하고, 각 패턴에 따라 경고 (alert), 차단(drop), 허용(pass) 등을 수행하지만, promiscuous 모드에서 네트워크 패킷을 차단하지 못하는 단점이 있다.

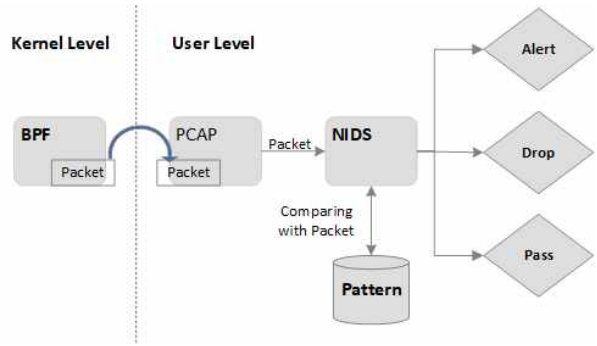

그림 1. 전통적 방식의 네트워크 침입탐지시스템 구조
Fig. 1. Traditional NIDS Structure

Ⅲ. 제안하는 Kenel Level의 NIDS

3-1 개요

MS Windows 플랫폼은 Kernel 모드에서 동작하는 TDI(Trasnsport Driver Interface)와 NDIS (Network Driver Interface Specification) Driver를 제공하고 이를 통해 네트워크 트래픽과 관련된 정보를 수집 및 차단할 수 있다. 이 중, NDIS 드라이버는 운영체제를 통해 송수신되는 모든 패킷을 Kernel Level에서 수집, 분석 및 허용/차단할 수 있는 개발 환경을 제공한다. Snort와 같은 User level의 NIDS에서 네트워크 패킷을 수집 및 분석할 경우, 많은 양의 네트워크 패킷를 처리하기 위해 부하가 많이 걸리며 이로 인한 패킷 펜딩과 같은 문제와 시간 낭비를 초래할 수 있다[19]. 만일, Kernel Level의 NDIS가 패킷을 캡처하고 User Level의 NIDS에게 패킷 정보를 전달할 경우, 많은 처리 수행 시간이 소요된다[5][20]. 따라서, [그림 2]와 같이, NDIS가 패킷을 캡처해서 직접 Kernel Level에 있는 NIDS에게 직접 전달할 수 있다면, 패킷 처리 속도가 응용 프로그램상에 전달하는 것보다 더욱더 빠르게 처리될 수 있으며, NDIS모듈은 하위 물리적인 네트워크 계층에서 상위 응용계층까지 처리할 수 있으므로 패킷에 대한 허용 및 차단 기능을 쉽게 구현할 수 있다.

본 연구에서 제안한, 윈도우즈 NDIS를 활용한 커널 기반 네트워크 침입탐지시스템 (Kernel based Network Intrusion Detection System, KNIDS) 구현은 빠른 처리 속도 및 패킷을 제어하는 장점이 있지만, 시스템 구현이 복잡하고 어렵다는 단점도 존재한다.

3-2 Kernel Level의 NIDS 구조

Kernel Level의 NIDS는 네트워크 카드, NDIS, KNIDS, 탐지 패턴 DB 및 사용자 프로그램으로 구성된다. NIC 카드와 사용자 프로그램은 구현 대상이 아니며, 본 연구를 통해서 구현된 부분은 NDIS, KNIDS 및 패턴 DB 부분이다. [그림 2]와 같이, User Level의 응용프로그램(ex. Browser)은 인터넷을 통해 특정 홈페이지에 접속하기 위해 네트워크 연결을 시도한다. NDIS는 해당 패킷 정보를 네트워크 카드에 전달하고, 응답을 기다린다. 네트워크 카드를 통해 패킷 정보가 전달될 경우, 해당 정보는 Kernel Level의 NDIS를 통해 KNIDS에게 전달되고, 악성 패킷인지 정상 패킷인지에 대한 결과를 기다린다. KNIDS는 수집된 패킷을 패턴과 비교하여 어떻게 처리할 것인지 결정하고, 결과를 NDIS에게 보낸다. 만일, NDIS가 KNIDS의 패킷 통과 요청을 받을 경우, 비로소 User Level의 응용프로그램은 특정 홈페이지에 접속할 수 있다.

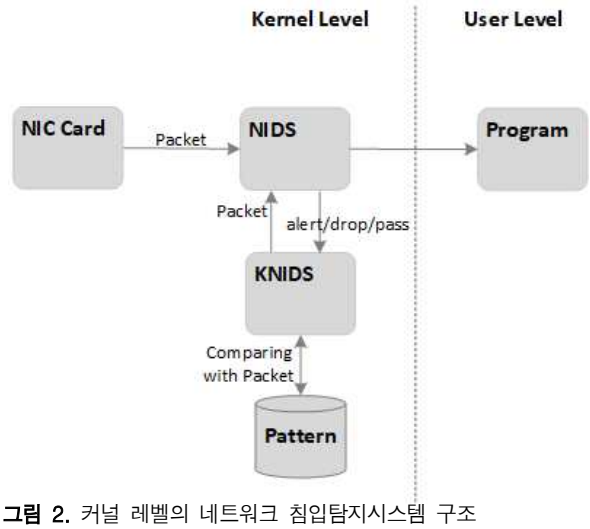

그림 2. 커널 레벨의 네트워크 침입탐지시스템 구조
Fig. 2. NIDS Structure of Kernel Level

3-3 Kernel Level의 NIDS 침입탐지

Snort는 일반적으로, 네트워크 카드를 통해 수신되는 패킷은 인터넷 프로토콜을 기반으로 동작하고, Authentication Packet과 Data Packet으로 구분되며, Authentication Packet을 통해 세

션을 설정한 후, Data Packet을 통해 상호 전달하고자 하는 정보를 제공한다. NDIS는 네트워크 카드를 통해 송수신되는 모든 패킷을 수집하여 패킷의 헤더를 차례로 분리하여 원하는 정보를 수집한 후, 응용프로그램에 관련 정보를 제공한다.

KNIDS는 NDIS가 파싱하여 분석한 정보를 응용프로그램에 바로 전달하지 않고, KNIDS 분석 모듈을 통과 후, 처리된 정보를 반환하며, NDIS를 통해 응용프로그램에 전달하는 방식이어서 악의적인 목적의 패킷과 데이터를 탐지 및 차단할 수 있다. KNIDS는 NDIS의 네트워크 파서를 통해 패킷의 헤더 정보를 전달받는다. 헤더 정보는 해당 패킷의 프로토콜 정보, 포트 정보 및 데이터 크기 등이 기록되어 있다. 패킷 헤더 정보에 따라 비교 검색할 규칙 그룹을 매핑한다. 예를 들어, 규칙 그룹에는 TCP 그룹, IP 그룹으로 구분되어 있고, 모든 패킷을 모든 규칙과 패턴매칭 하는 것보다는 TCP 패킷일 경우, TCP 규칙 그룹만 검색 및 비교하고, IP 패킷일 경우, IP 규칙 그룹만 검색하면 더욱 효율적으로 검색할 수 있다. 패킷의 비교 대상 규칙 그룹이 설정되면, 빠른 패턴매칭을 통해 두 번째 검색을 수행한다. 빠른 패턴매칭에서는 규칙의 패턴 전체를 비교 분석하는 것보다는 비교 대상 패킷 정보 중, 일부가 규칙 그룹의 패턴 정보와 일치하는지 확인하고, 만일 일치하지 않으면, 통과시키고, 일치할 경우 특정 플래그를 붙여 마지막 필터링 단계로 전달한다. 마지막 필터링 단계에서는 빠른 패턴매칭 모듈을 통해 전달된 패킷 정보를 특정 규칙 그룹의 모든 규칙 리스트와 직접 비교 분석하여 악의적인 패킷을 탐지 및 차단한다.

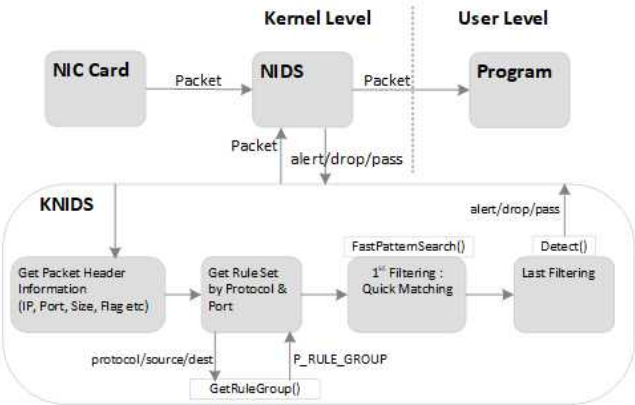

그림 3. 네트워크 침입탐지시스템의 침입탐지 과정
Fig. 3. Intrusion Detection Process of KNIDS

1) 규칙 그룹 찾기

[그림 3]에서 보여준 KNIDS의 침입탐지 규칙 그룹 검색 알고리즘은 [그림 4]와 같다. 침입탐지 규칙 그룹 검색 알고리즘은 KNIDS가 호출한 GetRuleGroup()의 입력값 즉, 프로토콜, 소스 포트, 목적지 포트에 대하여 3가지 변수(소스 규칙 그룹, 목적지 규칙 그룹, 일반 규칙 그룹)를 반환한다. 참고로, 반환된 3가지 변숫값은 모두가 값을 가지지 않고 그중 일부는 NULL이 될 수도 있다. 규칙 그룹을 찾는 순서는 아래와 같다.

가) 입력된 패킷에 대한 프로토콜 그룹 선택
나) 소스 및 목적지 포트검색을 통해 프로토콜 그룹에서 대응되는 규칙 그룹 선택
다) 찾은 특정 규칙 그룹과 임의 규칙 그룹 반환

```
Algorithm Group Search for Intrusion Detection Rule
Input: Network Protocols
Output: TCP/IP/UDP/ICMP Protocols, TCP Source/Destination Port, TCP Any Port, UDP Source Port,
ICMP Any Port, IP Any Port
Procedure {
        switch (protocol)
          {
          case IPPROTO_TCP: //TCP 프로토콜 경우
            srcPortGroup = patternConf_->tcpSrcPortGroup; //TCP소스 포트 그룹
            dstPortGroup = patternConf_->tcpDstPortGroup; //TCP목적지 포트 그룹
            genericPortGroup = patternConf_->anyPortGroup->tcpAnyAny; //TCP임의 포트 그룹
            break;
          case IPPROTO_UDP: //UDP 프로토콜 경우
            srcPortGroup = patternConf_->udpSrcPortGroup;
        break;
          case IPPROTO_ICMP: //ICMP 프로토콜 경우에는 포트 상관 없이 임의 포트 그룹만 사용
            genericPortGroup = patternConf_->anyPortGroup->icmpAnyAny;
            break;
          default:
            genericPortGroup = patternConf_->anyPortGroup->ipAnyAny;
            break;
          }
        }
End
```

그림 4. 네트워크 침입탐지시스템의 탐지 그룹 검색 알고리즘
Fig. 4. Searching Algorithm for Detection Group of KNIDS

2) 빠른 패턴 매칭

침입탐지 규칙 그룹을 검색한 KNIDS는 다음 단계로, 빠른 패턴매칭 기능을 [그림 5]의 알고리즘과 같이 수행한다. 빠른 패턴매칭 모듈은 다음과 같은 순서로 동작한다.

가) 수신 패킷의 데이터를 빠른 패턴매칭으로 비교
나) 입력데이터가 빠른 패턴매칭과 불일치하면 패스
다) 입력데이터가 빠른 패턴매칭과 일치하면 매치 플래그를 설정하고, 마지막 필터링 단계로 전달

빠른 패턴매칭 모듈은 KNIDS의 첫 번째 필터링으로, KNIDS는 필터링할 때 패킷이 어떤 규칙의 빠른 패턴과 일치하더라도 그 규칙에 따라 조치하는 것은 의미가 없다. 그 이유는 하나의 규칙 패턴들은 빠른 패턴 이외에 다른 패턴도 갖고 있기 때문이다. 예를 들어; alert tcp $EXTERNAL_NET any -> $HOME_NET 5800(content:"GET"; depth:4; isdataat: 1029,relative; content:!"|0A|"; within: 1024; sid:17708; rev:2;) 규칙[21]에서 문자열 'GET'는 그 규칙의 제일 긴 문자열로, 빠른 패턴이다. 비록, 한 패킷의 데이터는 'GET'를 포함하지만, 이 패킷이 규칙을 만족한다고는 할 수가 없다. 이 규칙에는 'isdataat: 1029, relative; content:!"|0A|"; within:1024;' 와 같이 다른 패턴도 있기 때문이다. 참고로, 'isdataat:1029, relative; content:!"|0A|"; within: 1024;'규칙은 1029에 데이터가 있는지 (isdataat:1029) 확인하고, 이 위치(1029)에서부터 이후의 1024byte 위치인 2053 (1029+1024=2053)까지의 위치 이내에는 '0A'가 없어야 한다(content:!"|0A|"; within:1024)는 의미이다. KNIDS는 우선 빠른 패턴을 검사하여 일치하는 규칙에 대한 매칭 플래그를 설정하고, 매칭 플래그의 값이 true면 마지막 탐지 단계로 넘어가고, false면 패킷을 pass 시킨다. 또한, 빠른 패턴

매칭 과정에서는 callback()을 사용해서 일치하는 규칙 항목에 대한 플래그를 설정한다. 각 규칙 항목에 있는 부울 매개 변수 'fpMatched'는 이 단계에서 설정하고 마지막 탐지단계에서는 확인한다.

```
Algorithm for Final Checking of Group Rule in KNIDS
Input: Rule Group Patterns, TCP Direction, TCP State
Output: False, Is Matched, Not Matched
Procedure
Begin {
        BOOL Detect(P_RULE_GROUP ruleGroup, CHAR* data, USHORT dataLen,
                    TCP_DIRECTION tcpDirection, TCP_STATE tcpState)
        {
          BOOL isMatch = FALSE;
          if (!ruleGroup)
          {
            return FALSE;
          }

          isMatch = RealDetect(ruleGroup->noContentRuleList, data, dataLen, tcpDirection,
                    tcpState);

          if (!isMatch)
          {
            isMatch = RealDetect(ruleGroup->caseMatchRuleList, data, dataLen, tcpDirection,
                      tcpState);

            if (!isMatch)
            {
              isMatch = RealDetect(ruleGroup->noCaseruleList, data, dataLen, tcpDirection,
                        tcpState);
            }
          }

          return isMatch;
        }
}
End
```

그림 5. 네트워크 침입탐지시스템의 빠른 패턴 매칭 알고리즘
Fig. 5. Quick Pattern Matching Algorithm of KNIDS

3-4 제안한 KNIDS의 패킷 처리 성능

[표 1]과 같이, 같은 패킷과 규칙을 사용한 경우에 NIDS와 KNIDS의 성능을 비교하였다. 패킷 수(2,568)는 특정 웹 포털 사이트에 접속할 때 input/output 되는 총 패킷 수이다.

표 1. 패킷 처리 결과
Table 1. Result of Packet Processing

Division	1st Test		2nd Test	
	NIDS	KNIDS	NIDS	KNIDS
Packet Number	2,568	2,568	2,568	2,568
Rule Number	1,075	1,075	2,362	2,362
Initial Time(ms)	393	796	6,150	6,656
Memory(MB)	30	33	193	233
Total P. T.	259	48	399	78
Average P. T.	0.101	0.019	0.155	0.030
P. M.	30	33	193	233

※ P.T. = Process Time(ms), P.M. = Process Memory(MB)

NIDS와 KNIDS가 사용하는 규칙은 Snort의 취약성 연구팀에 의해 인증된 규칙이고, 규칙 구조 파일(예, snort.conf)에서 정의된다. 각 규칙 파일은 다수의 규칙이 포함되어 있다. Snort는 총 53개 규칙 파일이 있고, 2,362개 규칙이 들어 있다. 즉, 53개 규칙 파일의 규칙을 모두 적재하면 규칙 수는 2,362이다. 53개 규칙 파일 중의 33개 파일을 적재하면 2,362의 절반인 약 1,000개(1,075) 규칙이 존재한다. [표 1]에서 규칙 수가 2,362개인 경우와 그 절반인 1,075개 일 경우를 각각 테스트 및 비교 분석하였다. KNIDS는 메모리와 초기화 시간을 NDIS 보다 더 사용하는 반면, 처리 속도가 더 빠르고, 패턴매칭 규칙의 수가 많아질수록 초기화 시간, 메모리 사용량 및 처리시간을 많이 사용하는 것으로 분석되었다.

Ⅴ. 결 론

유무선 네트워크 환경의 폭발적인 발전과 더불어, 컴퓨터 하드웨어 및 소프트웨어의 기술 발달로 악의적인 목적의 Malware 역시 기하급수적으로 늘어나, 매일 100만 개가 넘는 악성코드가 생성 및 배포, 탐지되고 있다. 따라서, 이를 정확히 짧은 시간 내에 탐지하기 위해 다양한 수단이 개발되고 있다.

그러나 Snort, Suricata[22]와 같이 User Level에서 동작하는 NIDS는 응용프로그램 상에서 패킷을 관찰, 분석 및 탐지하는 기능을 제공함으로 대량의 패킷을 분석할 때 많은 탐지 시간이 소요되는 문제가 있다. 이 문제를 해결하기 위하여 우리는 User Level에서 동작하는 네트워크 침입탐지시스템의 탐지 성능 향상을 위해 Kernel Level에서 동작하는 네트워크 침입탐지시스템을 제안하였다. 이 논문에서 제안한 Kernel Level에서 동작하는 NIDS가 User Level에서 동작하는 NIDS에 비해 메모리 초기화 시간과 메모리양을 좀 더 필요로 하지만, 패킷에 대해 처리 속도는 5배 이상 향상됨을 실험을 통해 확인하였다. 또한, 기존 NIDS와 달리 악의적인 행위에 대한 탐지 시, 해당 패킷을 즉시 차단함으로써 침입차단시스템과 연동하여 패킷을 차단하는 불편함을 해소할 수 있다. 추가로, 향후 KNIDS에 대한 메모리 누수 및 효율적인 메모리 사용에 관한 연구가 추가로 필요할 것이다.

참고문헌

[1] Wikiphedia, Intrusion Detection System[Internet], Available: https://en.wikipedia.org/wiki/Intrusion_detection_system.

[2] Snort[Internet], Available : http://www.snort.org.

[3] S. Chakrabarti, M. Chakraborty, and I. Mukhopadhyay, "Study of snort-based IDS", Proceedings of the International Conference and Workshop on Emerging Trends in Technology, ACM, 2010.

[4] Jay Beale, James C, Foster Jeffery Posluns, Brian Caswell, "Snort 2.0 Magic Box", Acorn, 2003.

[5] Myeong-Ki Jeong, Seong-Jin Ahn, Won-Hyung Park, "A Comparative Study on Function and Performance of Snort and Suricata", *The Journal of Information and Security*, Vol. 14, No 5, pp.3-8, Sep 2014.

[6] Yong-Sik Jeon, "Cost-Based Optimizer Detection Tree configuration plan for performance improvement of Signature-Based IDS", M.S. dissertation, Korea , Seoul, 2017.05.

[7] Snort, Snort User Manual[Internet], Available : http://manual-snort-org.s3-wesite-us-east-1.

[8] In-Kyoung Kim, Eul-Gyu Im, “A Study on the Analysis Rule for Network Intrusion Detection System using Snort”, *The Journal of Korean Institute of Communications and Information Sciences*, Vol. 2011, No. 6, pp. 656-658, Jun 2011.

[9] Ji-yong Han, In-bok Lee, Jung-Hee Han, “Accelerating PCRE Performance of Signature-based IDS”, *The Journal of Korean Instityte of Information Scientistics and Engineering : System and Theory*, Vol. 40, No. 2, pp. 53-60, Feb 2013.

[10] M. Alicherry, M. Muthuprasanna, and V. Kumar, “High speed pattern matching for network IDS/IPS”, *Proceedings of the 2006 14th IEEE International Conference on. IEEE,* Santa Barbara, CA, Feb 2006.

[11] M. Roesch, “Snort : Light weight Intrusion Detection for Networks”, *Proceedings of LISA ‘99:13th Systems Administration Conference*, Seattle, WA, Nov 1999.

[12] Kil-Ho Lee, “A Study of Network Intrusion Detection System using Snort”, M.S., Gyeongsang, Aug 2017.

[13] Security Tools, Security Tool Top 100[Internet], Available : http://www.sectools.org

[14] Ho-Sung Jo, Sung-Il Oh, In-Bok Lee, Hee-Jin Park, Joong-Chae Na, “Development and Application of a Similarity Analysis Program for Snort-based Detection Rules”, *The Journal of Korean Institute of Next Generation Computing*, Vol.11, No.1, pp. 32-43, Feb 2015.

[15] K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems(IDPS)”, NIST, Gaithersburg, MD, Special Publication 800-94, Feb 2007.

[16] Seok-Jin Ug, Moon-Seok Choi, Ji-Myung Kim, Jong-Soon Park, “A Comparative Study on Performance of Open Source IDS/IPS Snort and Suricata”, *The Journal of Korea Society of Digital Industry and Information Management,* Vol. 12, No. 1, pp. 89-95, Mar 2016.

[17] Yong-Woo Jung, “A Study on Normalized Rules of Security System”, M.S. dissertation, Soongsil, Seoul, Jun 2017.

[18] Keon-Woong Kong, Yong-gwan Won, “Implementation of Encrypted Mail Program using SMTP and POP3”, *the Journal of Digital Contents Society*, Vol. 18, No. 7, pp.1403-1409, Nov 2017.

[19] Doo-Won Sik, “A Study on the False Positive detection method of Intrusion Prevention System Using SVM”, M.S. dissertation, Sungkyunkwan, Seoul, Apr 2017.

[20] Dong-Hee Han, “A Study on the Method for Selecting Snort Intrusion Detection Rules for Improvement of Efficiency and Reduction of False Positive”, M.S. dissertation, Korea, Seoul, 2015.12.

[21] Kedar Namjoshi, Girija Narlikar, "Robust and Fast Pattern Matching for IDS", 2010 Proceedings IEEE, 2010.03.

[22] Suricata[Internet], Available: http://www.suricata-ids.org.

김의탁(Eui-Tak Kim)

1997년 : 대전대학교 컴퓨터공학과 (공학사)
1999년 : 대전대학교 대학원 컴퓨터공학과 (공학석사)
2018년 현재 : 충북대학교 대학원 전자계산학과 (박사수료)

2001년~2005년 : ㈜아이언마스크 정보보호연구소 기술이사
2005년~2010년 : 티에스온넷(주) 정보보호연구소 부장
2010년~현 재 : ㈜하우리 기술연구소 연구소장
※관심분야 : 악성코드 분석 및 탐지(Anti-Virus, Reverse Engineering), 데이터 마이닝(Data Mining), 클라우드 컴퓨팅 (Cloud Computing), 핀테크 보안(FinTech Security), 인공지능(AI), 접근통제시스템(Access Control System)

류근호(Keun Ho Ryu)

1976년 : 숭실대학교 전자계산학과 (공학사)
1980년 : 연세대학교 대학원 전자계산학과 (공학석사)
1988년 : 연세대학교 대학원 전자계산학과 (공학박사)

1976년~1986년 : 육군 3군수 지원사 전산실 (ROTC 장교), 한국전자통신연구원 (연구원), 한국방송통신대 전산학과(조교수)
1989년~1991년 : 미국 Univ. of Arizona Research Staff (TempIS 연구원, Temporal DB)
1986년~현 재 : 충북대학교 전자정보대학 소프트웨어학과 (교수)
※관심분야 : 시간데이터베이스, 시공간 데이터베이스, Temporal GIS, 지식기반 정보검색, 데이터마이, 데이터보안, 바이오메디칼 및 바이오인포매틱스