논문 15-40-04-08 The Journal of Korean Institute of Communications and Information Sciences '15-04 Vol.40 No.04
http://dx.doi.org/10.7840/kics.2015.40.4.666

네트워크 트래픽 분석을 위한 Snort Content 규칙 자동 생성

심 규 석◆, 윤 성 호*, 이 수 강*, 김 성 민*, 정 우 석*, 김 명 섭°

Automatic Generation of Snort Content Rule for Network Traffic Analysis

Kyu-Seok Shim◆, Sung-Ho Yoon*, Su-Kang Lee*, Sung-Min Kim*, Woo-Suk Jung*, Myung-Sup Kim°

요 약

효과적인 네트워크 관리를 위해 응용 트래픽 분석의 중요성이 강조되고 있다. Snort는 트래픽 탐지를 위해 사용되는 보편적인 엔진으로써 기 정의된 규칙을 기반으로 트래픽을 차단하거나 로그를 기록한다. 하지만 Snort 규칙을 생성하기 위해서는 탐지 대상 트래픽을 전수 조사해야하기 때문에 많은 한계점이 존재할 뿐만 아니라 생성된 규칙의 정확성을 보장하기 어렵다. 본 논문에서는 순차 패턴 알고리즘을 활용하여 입력된 트래픽에서 최소 지지도를 만족하는 문자열을 찾는 방법을 제안한다. 또한, 추출된 문자열을 사용한 규칙을 입력 트래픽에 적용하여 트래픽에서 해당 문자열이 존재하는 위치 정보 및 헤더 정보를 추출한다. 이렇게 추출된 문자열과 위치정보, 그리고 헤더 정보를 조합하여 Snort 규칙을 자동 생성하는 방법을 제안한다. 생성된 규칙을 이용하여 다시 트래픽 분석을 실시했을 때 대부분의 응용이 97%이상 탐지되는 것을 확인하였다.

Key Words : Snort, Automatic, Sequence Pattern, Extraction, Network Traffic Detection

ABSTRACT

The importance of application traffic analysis for efficient network management has been emphasized continuously. Snort is a popular traffic analysis system which detects traffic matched to pre-defined signatures and perform various actions based on the rules. However, it is very difficult to get highly accurate signatures to meet various analysis purpose because it is very tedious and time-consuming work to search the entire traffic data manually or semi-automatically. In this paper, we propose a novel method to generate signatures in a fully automatic manner in the form of sort rule from raw packet data captured from network link or end-host. We use a sequence pattern algorithm to generate common substring satisfying the minimum support from traffic flow data. Also, we extract the location and header information of the signature which are the components of snort content rule. When we analyzed the proposed method to several application traffic data, the generated rule could detect more than 97 percentage of the traffic data.

※ 본 연구는 BK21플러스사업(No.T1300572) 및 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단 차세대정보컴퓨팅기술개발사업(2010-0020728) 및 2012년 정부(교육과학기술부)의 재원으로 한국연구재단(2012R1A1A2007483)의 지원을 받아 수행되었습니다.
◆ First Author : Department of Computer and Information Science, Korea University, kusuk007@korea.ac.kr, 학생회원
° Corresponding Author : Department of Computer and Information Science, Korea University, tmskim@korea.ac.kr, 종신회원
* Department of Computer and Information Science, Korea University, {(sungho_yoon, sukanglee, gogumiking, hary5832)@korea.ac.kr}
논문번호 : KICS2014-12-006 Received December 4, 2014; Revised February 13, 2015; Accepted April 6, 2015

Ⅰ. 서 론

네트워크 관리의 목적은 네트워크 자원을 최대한 활용하여 사용자에게 목적에 맞는 서비스를 신속하게 제공하는 것이다. 이를 위해 네트워크 관리자들은 적절한 네트워크 정책을 수립하여 관리 대상 네트워크에 적용한다[1,2]. 네트워크 정책은 특정 트래픽을 차단하거나 대역폭을 조절하는 방법으로 수행되기 때문에 트래픽의 발생 원천을 알아내는 트래픽 분석이 선행되어야 한다. 트래픽 분석은 응용을 대표하는 고유한 특성(시그니처, 규칙)을 사용하여 트래픽을 발생시킨 응용을 판별하는 것으로써, 분석 결과는 네트워크 정책뿐만 아니라, 용량계획(capacity planning), 네트워크 권한 설정 (network provisioning), 트래픽 엔지니어링 (traffic engineering), 고장 진단(fault diagnosis) 등과 같은 다양한 분야에서 활용된다.

Snort[3]는 보편적으로 사용하는 트래픽 탐지 엔진이다. 실제 상용 회사에서도 Snort엔진을 트래픽 분석 및 탐지 장비 개발에 활용할 만큼 네트워크 트래픽 분야에서 매우 보편적인 도구이다. Snort는 기 정의된 규칙(Rule)을 사용하여 대응되는 패킷을 탐지하고 규칙에 정의된 행위(Action)를 수행한다. 규칙은 패킷 단위로 적용되며, 패킷의 헤더 정보 (IP address, port, protocol)와 통계 정보(packet size), 그리고 페이로드 정보(content, pcre, offset)등을 사용한다. 본 논문에서는 헤더 정보와 페이로드 정보에 초점을 맞추어 자동 규칙 생성 방법을 제안한다.

일반적으로 규칙을 생성하기 위해서는 분석 대상 트래픽을 전수 조사하여 해당 트래픽에서만 관찰되는 공통된 특징을 찾는 작업을 수행한다. 하지만 이러한 방법은 규칙 생성 시간이 많이 소요되고, 추출하는 사람의 능력에 따라 생성된 규칙의 정확도가 가변적이라는 한계점이 존재한다. 이를 개선하기 위해 다양한 자동 규칙(시그니처) 생성 방법[4-9]들이 제안되었다. 하지만 기존 제안된 방법들은 특정 두 문자열 사이에서 공통된 부분 문자열을 찾는 알고리즘을 사용하였기 때문에 실제 트래픽에 적용하기에는 많은 한계가 존재한다. 적용할 트래픽의 순서를 정하거나 그룹화시키는 전처리 과정과 생성된 부분 문자열을 하나의 규칙으로 통합시키는 후처리 과정이 필요하다. 또한, 전 후처리 과정에서 사용되는 임계값(threshold)에 따라 생성된 규칙의 개수와 정확도가 변화하기 때문에 탐지 대상에 최적화된 임계값을 찾는 반복적인 실험 과정이 필요하다.

이를 개선하기 위해 본 논문에서는 순차 패턴 알고리즘[10,11]을 활용하여 공통된 문자열(content)을 자동 추출하는 방법을 제안한다. 순차 패턴 알고리즘은 최소 지지도를 만족하지 못하는 부분 content를 조기에 제외함으로써 입력 데이터의 크기에 큰 영향 없이 신속하게 공통된 content을 추출할 수 있다. 또한, 공통된 content을 찾기 위해 특정 두 문자열을 지정해야 했던 기존 방법과 달리 추출 대상 문자열 전체를 입력으로 받아 처리하기 때문에 추가적인 전 후처리 과정이 필요 없다. 본 논문에서는 content 추출뿐만 아니라 추출한 content를 입력 트래픽에 적용하여 트래픽에서 해당 content의 위치 정보(depth, offset)와 헤더 정보를 추가적으로 추출한다.

본 논문은 2장에서 관련 연구에 대해 조사하고, 3장에서 5단계로 진행되는 snort content 규칙 자동 생성 방법에 대해 설명한다. 4장에서는 실험 결과를 기술하고, 마지막으로 5장에서 결론과 향후 연구에 대해 언급한다.

Ⅱ. 관련 연구

네트워크 자원 효율화를 위한 트래픽 분류 및 탐지 방법은 다양한 방면으로 연구가 진행되고 있다. 트래픽 분류 방법은 트래픽을 분석 시 사용하는 트래픽 특징을 기준으로 포트 기반 분석과 페이로드 기반 분석, 통계 정보 기반 분석 그리고 행위 기반 분석 등 트래픽의 특징(시그니처)을 기준으로 분석하는 시그니처 기반 분석 방법이 있다.

포트 기반 분석은 Internet Assigned Number Authority(IANA)에서 지정한 포트 정보를 이용한 트래픽 분석방법이다. 포트 정보와 대응하는 서비스를 트래픽에서 분류할 수 있다. 적은 메모리 사용으로 매우 빠르게 분석 할 수 있는 장점을 가지지만, 오늘날 네트워크를 사용하는 많은 응용들은 방화벽 및 IPS 장비를 통과하기 위해 포트 번호를 임의로 설정하여 트래픽을 발생시키고, 포트사용자가 설정하거나 매 실행 시 임의의 포트번호를 사용하기 때문에 더 이상 포트 번호가 특정 서비스, 프로토콜을 의미하지 않는다.

이러한 문제를 해결하기 위해 페이로드 기반 트래픽 분석 방법이 연구되었다. 페이로드 기반 분석 방법은 패킷의 페이로드 내에서 응용마다 가지는 특정 스트링이 포함 유무를 통해 트래픽을 분석하는 방법이다. 현재의 네트워크 트래픽 분류 방법 중 트래픽의 페이로드를 직접 검사하기 때문에 가장 신뢰도 있는 분석방법이다.

하지만 페이로드 기반 분석 방법을 사용하기 위해

서는 분류 대상 응용 트래픽의 특징을 파악하고 시그니처를 생성하는 전처리 과정이 필수적이다. 페이로드 기반 시그니처는 패킷의 페이로드 내용을 확인하여, 공통적인 스트링을 찾아 시그니처로 생성하여 특정 응용을 분류할 수 있다. 그러나 시그니처의 생성 및 관리가 어렵고 최근 암호화된 트래픽이 증가하고 보안 기술이 발전하면서 네트워크 관리자가 직접 눈으로 공통 스트링을 찾기에는 많은 어려움이 존재한다.

따라서 페이로드 시그니처 자동 추출 방법론이 연구되고 있다. 기존 페이로드 시그니처 자동 추출 방법으로는 LCS(Longest Common String) 또는 Smith-Waterman 알고리즘을 이용하여 공통적인 스트링을 추출하는 연구가 진행되고 있다.

LASER (LCS-based Application Signature ExtRaction)은 LCS(Longest Common String) 알고리즘을 응용 트래픽 시그니처 추출 목적에 맞게 변형한 알고리즘이다[6]. LASER 알고리즘을 이용한 시그니처 추출 방법은 두 개의 스트링을 비교하는 Matrix에서 Backtracking을 이용하여 공통 문자열을 찾는 방법이다. 따라서 시간 복잡도와 계산 복잡도가 높은 단점이 있다.

Smith-Waterman 알고리즘도 시그니처 추출에 많이 사용된다[5,8]. Smith-Waterman 알고리즘은 본래 DNA의 유사도를 판단하는 목적에 발표된 알고리즘이다. 이 방법 또한 두 개의 스트링을 비교하여 Matrix에 표시하고 다시 Backtracking으로 최대, 최적의 시그니처를 찾아내는 작업을 한다. 위의 LCS 알고리즘과 비교했을 때 가장 큰 차이는 Backtracking 방법이다. LCS 방법보다는 시간 복잡도와 계산 복잡도가 적지만 아직도 상당한 소요시간과 계산과정을 거쳐야한다.

다른 형태의 시그니처 자동 추출 방법인 Autosig[7]는 시그니처의 가능성이 있는 모든 공통 문자열을 추출하고 추출된 문자열을 구조화 하여 시그니처를 생성하는 방법이다. 이 방법은 가능성 있는 모든 공통 문자열을 추출 할 때, 너무 많은 문자열이 계산과정에 포함된다는 것이다. 예를 들면, 20개의 문자로 되어 있는 문자열에서 길이가 4인 문자열을 추출한다면 16개의 부분 문자열이 추출되고, 16개의 문자열은 모두 계산과정에 포함된다. 따라서 추출된 부분 문자열의 개수가 많고 범위가 넓기 때문에 메모리 사용률 및 처리 시간의 단점을 가지고 있다.

기존 방법들은 특정 두 문자열을 비교하여 실제 트래픽에 적용하기에는 많은 한계가 존재한다. 적용할 트래픽의 순서를 정하거나 그룹화시키는 전처리 과정과 생성된 부분 문자열을 하나의 규칙으로 통합시키는 후처리 과정이 필요하다. 또한, 생성하는 환경이 다르다면 생성되는 시그니처가 달라질 수 있기 때문에 신뢰도가 떨어진다. 이와 같은 단점을 해결하기 위해 본 논문에서는 특정 응용에서 공통 문자열을 순차 패턴 알고리즘을 이용하여 자동으로 추출하고, 추출된 공통 문자열을 이용하여 Snort content 규칙을 생성하는 방법을 제안한다.

본 논문에서 제안하는 순차 패턴 알고리즘을 이용한 자동 Snort content 규칙 생성 방법은 언급된 단점을 해결할 수 있다. 본 논문에서 제안하는 방법은 한 단어가 최소 지지도를 만족하지 않으면 제외한 후에 공통 스트링을 생성하기 때문에 시간복잡도 및 공간 복잡도가 입력데이터 크기의 영향이 크지 않다. 또한 입력 데이터가 두 개의 스트링이 아닌 모든 스트링이 입력된 후에 계산되기 때문에 전처리 과정 및 후처리 과정이 필요 없다. 방법론에 대한 자세한 설명은 Ⅲ장 Content 규 추출 알고리즘에서 기술한다.

Ⅲ. Content 규칙 추출 알고리즘

본 장에서는 순차 패턴 알고리즘을 사용하여 Snort content 규칙을 자동 생성하는 방법을 설명한다. Snort 규칙은 다양한 구성 요소를 표기할 수 있지만, 본 논문에서는 그림 1과 같이 헤더 정보, 페이로드 정보만으로 구성되는 규칙을 대상으로 한다.

그림 1에서 기술한 규칙 예시의 의미는 프로토콜은 TCP, 목적지 포트 번호는 80을 사용하는 패킷 중에서 "cgi-bin/phf"이란 content가 페이로드 4번째 바이트와 30번째 바이트 사이에 위치하는 경우 알림을 울리라는 의미이다.

예시와 같이 Snort 규칙을 자동 생성하기 위하여 그림 2와 같은 과정을 수행한다. 최초 호스트 별로 분석 대상 응용 및 서비스, 혹은 악성 코드가 발생트래픽을 수집한다. 단일 플로우에서 전송 방향이 같은 패킷들을 조합하여 하나의 sequence를 구성한다. sequence는 content를 추출하기 위한 순차 패턴 알고리즘의 입력으로 사용된다

```
Action Protocol SrcIP SrcPort → DstIP DstPort (Paylaod)
alert tcp any any → any 80 (content:"cgi-bin/phf"; offset:4; depth:20;)
```

그림 1. Snort 규칙과 예
Fig. 1. Snort Rule and example

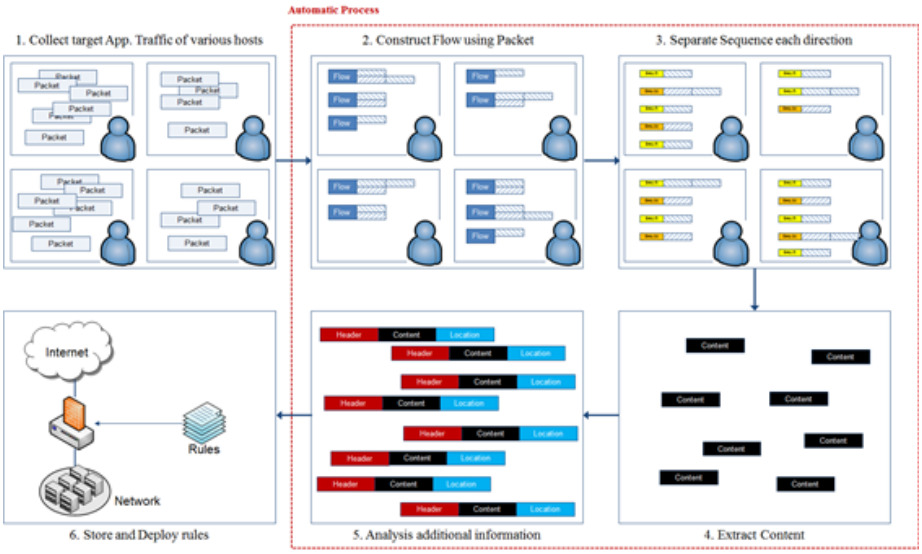

그림. 2. Snort content 규칙 자동 생성 흐름도
Fig. 2. The flow to automatically generate Snort content rule

순차 패턴 알고리즘은 입력 받은 sequence에서 길이가 1인 후보 content를 시작으로 길이를 증가시키면서 후보 content를 찾고 최종적으로 일정 수준이상의 지지도를 가지는 content를 추출한다. 단순히 content만을 규칙으로 사용할 경우 오탐(탐지 대상이 아닌 트래픽을 탐지)의 가능성이 높기 때문에 추가적인 정보를 분석하여 규칙에 기술한다. 본 논문에서 사용한 추가 정보는 헤더 정보와 content의 위치 정보이다. 추출된 content를 입력 트래픽에 적용하여 해당 content가 매칭되는 트래픽을 그룹화하고 해당 그룹의 공통된 헤더 정보와 위치 정보를 분석한다. 최종적으로 생성된 Snort 규칙은 Snort엔진이 탑재된 네트워크 장비에 적용한다. 본 논문에서는 수집된 트래픽에서 Snort content 규칙을 생성하는 단계를 대상으로 자동화 방법을 제안한다. 이어지는 장에서는 각 부분에 대한 자세한 내용을 설명한다.

3.1 트래픽 수집 단계

트래픽 수집은 규칙 생성을 위한 첫 단계이다. 트래픽 수집을 하기 위해서는 탐지 대상을 결정해야한다. 탐지 대상은 응용, 서비스, 공격, 악성 코드등 네트워크 관리 목적에 따라 매우 다양하다. 탐지 대상이 결정되면 탐지 대상에서 발생하는 트래픽을 수집한다. 트래픽을 발생시키는 호스트에서 직접 트래픽을 수집할 경우 wireshark[14], tcpdump[15]와 같은 네트워크 트래픽 수집 도구를 사용한다. 네트워크 전체 트래픽을 수집할 경우 스위치의 미러닝 기능이나 탭 장비를 사용하여 수집한다. 수식 (1), (2)는 수집한 패킷의 형태를 나타낸다. PacketSet은 패킷들의 집합을 의미하고 단일 패킷 P는 호스트 ID, 소스 IP 주소, 소스 포트 번호, Layer4 프로토콜, 목적지 IP 주소, 목적지 포트 번호, 그리고 페이로드로 구성된다. 특히, payload는 연속된 문자들로 구성되며 본 논문에서 자동 생성하는 content는 페이로드의 부분 문자열을 의미한다.

$$PacketSet = \{P_1, P_2, \ \ldots, P_p\} \tag{1}$$

$$P_i = \begin{cases} host_{id}, srcIP, srcPort, L4Prot, dstIP, dstPor \\ payload = \langle a_1 a_2 a_3 \ldots a_a \rangle \end{cases} \tag{2}$$

규칙 생성을 위한 트래픽 수집에서는 탐지 대상 트래픽 만을 수집해야 하기 때문에 개별 호스트에서 트래픽을 수집하는 것이 생성된 규칙의 정확성을 높이는 측면에서 권장한다. 본 논문에서 사용하는 지지도는 입력 트래픽을 발생시킨 호스트를 기준으로 하기 때문에 최소 2개 이상의 호스트에서 트래픽을 수집해야 한다. 하지만 실제 트래픽 수집 환경에서 여러 호스트의 트래픽을 수집하는 것은 매우 번거롭고 불가능한 경우가 있기 때문에 동일한 호스트에서 수집한 트래픽을 여러 파일에 나누어 저장하고 지지도를 호스트 기준이 아닌 입력 파일 기준으로 계산하는 방법

도 적용할 수 있다. 즉, 트래픽 수집의 환경에 따라 지지도를 계산하는 기준이 변화할 수 있다. 자세한 설명은 4절 Content 추출 단계에서 설명한다.

3.2 플로우 구성 단계

수집된 패킷 집합 트래픽을 플로우로 구성한다. 본 논문에서 사용한 플로우는 수식 (3), (4)와 같이 5-tuple이 동일한 패킷의 집합이다. 단, source측과 destination 측이 대칭되는 플로우는 하나의 플로우로 구성하고 각 패킷 집합에 전송 방향(forward, backward)을 기입한다. 즉, 본 논문에서 정의한 플로우는 5-tuple이 동일한 패킷 집합과 이와 대칭 패킷 집합을 포함하는 양방향 플로우이다. $host_{id}$는 지지도를 계산하기 위해 어떤 호스트에서 해당 플로우가 수집이 되었는지 명시해준다.

$$F = \{F_1, F_2, \ \ldots, F_f\} \tag{3}$$

$$F_i = \left\{ \begin{matrix} host\_id, srcIP,\ srcPort,\ L4Prot,\ dstIP,\ dstPort, \\ forward = \{P_1, P_2, \ldots, P_x | P_{1.5tupe} \ldots = P_{x.5tupe}\}, \\ backward = \{P_1, P_2, \ldots, P_y | P_{1.5tupe} \ldots = P_{y.5tupe}\} \\ | forward.srcIP = backward.dstIP, \\ forward.srcPort = backward.dstPort, \\ forward.L4Prot = backward.L4Prot, \\ forward.dstIP = backward.srctIP, \\ forward.dstPort = backward.srcPort \end{matrix} \right\} \tag{4}$$

패킷을 플로우로 구성하는 이유는 비록 snort가 패킷 단위에 적용되기는 하나 네트워크 특성상 단일 메시지가 여러 패킷으로 나누어 전송되는 경우(패킷 단편화)가 발생한다. 따라서 단일 플로우를 구성하는 패킷들을 전송 방향 별로 구분하여 페이로드를 합치면 메시지의 끊김 없이 실제 전송된 페이로드 메시지를 확인할 수 있다.

3.3 Sequence 구성 단계

플로우의 순방향(forward), 역방향(backward)으로 구분된 패킷들의 페이로드만을 추출하여 하나의 sequence를 만든다. 만약 플로우가 양방향 통신 패킷들로 구성되었다면, 2개의 sequence가 생성되고 단 방향 통신 트래픽이면 1개의 sequence가 생성된다. SeqeunceSet은 수식 (5)와 같이 여러 sequences(S)들로 구성되며, 하나의 sequence는 수식(6)과 같이 호스트ID와 문자열 $\langle a_1 a_2 a_3 \ldots a_n \rangle$로 구성된다.

$$SequenceSet = \{S_1, S_2, \ \ldots, S_s\} \tag{5}$$

$$S_i = \{host\_id, \langle a_1 a_2 a_3 \ldots a_n \rangle\} \tag{6}$$

수식 (5),(6)과 같이 구성된 sequence에 호스트ID를 기입한다. 이는 다음 단계인 content 추출 단계에서 지지도 계산을 위해 사용된다. 만약 지지도 계산을 파일 기준으로 계산할 경우 파일ID를 기입한다.

3.4 Content 추출 단계

content 추출 단계에서는 sequence 집합과 최소 지지도를 입력 받아 최소 지지도를 만족하는 content를 추출한다. 본 알고리즘은 대용량 데이터 베이스에서 순차 패턴을 찾는 Apriorior 알고리즘[16]을 content 추출 환경에 맞게 개선하였다. 알고리즘의 출력인 ContentSet은 수식 (7)과 같이 여러 content(C)들로 구성되며, 하나의 content는 수식 (8)과 같이 sequence 문자열의 연속된 부분 문자열이다. Algorithm 1, 2는 입력 받은 sequence 집합으로부터 기 정의된 최소 지지도를 만족하는 content 집합을 출력하는 방법을 나타낸다.

Algorithm 1과 같이 content 추출 알고리즘이 수행되면, 입력 받은 sequence 집합의 모든 sequence에서 길이 1인 content를 추출하여 길이 1 content 집합 L_1에 저장한다(Algo.1 Line: 1~5), 길이 1인 content를 시작으로 길이를 1씩 늘려가며 모든 길이의 content를 추출하여 자신의 길이 content 집합 L_k에 저장한다(Algo.1 Line: 6~20).

$$ContentSet = \{C_1, C_2, \ \ldots, C_c\} \tag{7}$$

$$C_i = \{\langle a_x a_{x+1} \ldots a_y \rangle | 1 \le x \le y \le n, \} \tag{8}$$

$$Support = \frac{Number\ of\ support\ hosts}{Total\ number\ of\ hosts} \tag{9}$$

단, 새로 생성된 집합 L_{k-1}의 모든 content 중, 입력 받은 최소 지지도를 만족하지 않는 content는 삭제한다(Algo.1 Line: 8~17). 최소 지지도를 만족하지 못하는 content는 content 추출 자격을 만족하지 못할 뿐만 아니라 해당 content를 확장한 content에서도 최소 지지도를 만족하지 않기 때문이다.

최소 지지도를 만족하지 못하는 content가 삭제된 집합 L_{k-1}의 content는 집합 L_k을 생성하는데 사용된다(Algo.1 Line: 18). 이때 사용하는 방법은 Algorithm 2에 기술된 방법이다. 입력 받은 집합

Input : SequenceSet = $\{S_1, S_2, ..., S_s\}$, minsupp
Output : ContentSet = $\{C_1, C_2, ..., C_c\}$

contentExtractor(SequenceSet, minsupp)
1: **foreach** sequence S in the SequenceSet **do**
2: **foreach** character a in the sequence S **do**
3: $L_1 = L_1 \cup a$;
4: **end**
5: **end**
6: k=2;
7: **while** $L_{k-1} = \varnothing$ do
8: **foreach** content c in the L_{k-1} **do** // delete under minsupp
9: **for** $i = 1$ to s **do**
10: **if**(S_i $include$ c)**then**
11: count = count + 1;
12: **end**
13: **end**
14: **if** ($(count/s) < minsupp$) **then**
15: $L_{k-1} = L_{k-1} - C$;
16: **end**
17: **end**
18: L_k=***candidate_gen***(L_{k-1}) //extract length-k content
19: k++;
20: **end**
21: $ContentSet = \forall L_k$
22: ***deleteSunset***(ContentSet);
23: **return** ContentSet

***Notation** : L_k : lengh k content set

알고리즘.1. Content 추출 알고리즘
Algorithm.1. The content extract Algorithm

Input : L_{k-1}
Output : L_k

candidate_gen(L_{k-1})
1: **foreach** content p in L_{k-1} **do**
2: **foreach** content q in L_{k-1} **do**
3: **if** ($(p.a_2 = q.a_1)$ && $(p.a_3 = q.a_2)$ ... && $(p.a_{k-1} = q.a_{k-2})$) **then**
4: $L_k = L_k \cup \langle p.a_1, p.a_2, ..., p.a_{k-1}, q.a_{k-1} \rangle$;
5: **end**
6: **end**
7: **end**
8: **return** L_k

***Notation** : L_k : lengh k content set

알고리즘.2. 후보 Content 추출 알고리즘
Algorithm.2. The subcontent extract Algorithm

L_{k-1} 의 content들을 비교하여 집합 L_k의 content를 생성한다. 집합 L_{k-1}의 content를 조합하여 집합 L_k의 content를 생성하는 것은 첫 문자를 제외한 길이 k-2 content와 마지막 문자를 제외한 길이 k-2 content가 동일한 집합 L_{k-1}의 content들끼리 가능하다 (Algo.2 Line: 1~7). 예를 들어 집합 L_4의 content인 “abcd″와 ”bcde″는 “a″를 제외한 ”bcd“와 ”e″를 제외한 “bcd″가 같기 때문에 집합 L_5의 content ”abcde″를 생성할 수 있다.

위와 같은 방법으로 길이를 1씩 증가 시키면서 더 이상 새로운 content가 추출되지 않을 때까지 content 추출과 지지도 미만 content 삭제를 반복한다. 추출의 마지막 단계로써 추출된 모든 길이의 content 포함관계를 확인하고, 만약 포함 관계에 있는 content가 발견되면 해당 content를 집합에서 삭제한다(Algo.1 Line: 22). 최종적으로 생성된 content 집합을 다음 단계로 전달한다.

표 1은 content 추출 알고리즘 수행 과정 예시를 나타낸다. 위의 예시에서는 3명의 호스트에서 발생한 트래픽으로 구성된 SequenceSet과 최소 지지도 0.6을 입력받는다. SequenceSet 은 4개의 sequence로 구성되어있다. 최소 지지도가 0.6이라는 의미는 전체 호스트 수가 3이기 때문에 최소 2명의 호스트에서 발생된 트래픽에 해당 content가 관찰되어야 한다는 의미이다. 알고리즘이 수행되면 길이 1인 모든 content를 추출한다. 길이 1인 content($\langle C \rangle, \langle M \rangle, \langle D \rangle$) 모두 최소 지지도 0.6을 만족하기 때문에 길이 2 content 생성에 사용된다. 길이 2 content 생성 후, 최소 지지도를 계산하고 만족하지 못하는 content 생성 후, 최소 지지도를 만족하는 길이 2 content($\langle CM \rangle, \langle MD \rangle$)를 사용하여 길이 3 content를 추출한다. 최소 지지도를 만족하는 길이 3 content의 개수가 1 이기 때문에 더 이상 content의 길이를 늘리는 것은 불가능하다. 따라서, content 추출을 종료한다. 추출 종료 후, 포함관계가 있는 $\langle C \rangle, \langle M \rangle, \langle D \rangle$는 contentSet에서 삭제되고 최종적으로 $\langle CMD \rangle$를 다음 단계로 전달한다.

3.5 추가 정보 분석 단계

앞선 단계에서 추출된 content 정보만을 사용하여 Snort 규칙을 작성할 경우, 오탐의 가능성이 높다. 즉, 추출된 content의 길이가 너무 짧을 경우, 탐지 대상이 아닌 트래픽에서 해당 규칙이 적용될 수 있다. 그림 3,4와 같이 content 정보만을 사용한 규칙과 그렇지 않은 규칙은 큰 차이를 보인다. 그림 3의 예는 패킷

표 1. Content 추출 알고리즘 수행 과정 예시
Table 1. The example that process to content extract algorithm

$SequnceSet = \{S_1, S_2, S_3, S_4\}$	$minsupp$
$S_1 = \{host_1, \langle .... CMD .... \rangle\}$ $S_2 = \{host_1, \langle .... CD .... \rangle\}$ $S_3 = \{host_2, \langle ..... CMD .... \rangle\}$ $S_4 = \{host_3, \langle CM .... \rangle\}$	0.6

L_1

Length 1 content	Support
$\langle C \rangle$	1.00
$\langle M \rangle$	1.00
$\langle D \rangle$	0.67

L_2

Length 2 content	Support
$\langle CC \rangle$	0.00
$\langle CM \rangle$	1.00
$\langle CD \rangle$	0.34
$\langle MC \rangle$	0.00
$\langle MM \rangle$	0.00
$\langle MD \rangle$	0.67
$\langle DC \rangle$	0.00
$\langle DM \rangle$	0.00
$\langle DD \rangle$	0.00

L_3

Length 3 content	Support
$\langle CMD \rangle$	0.67

페이로드 전체를 검사하면서 해당 content가 존재하는지를 검사한다. 하지만, 그림 4는 TCP 프로토콜을 사용하고, 목적지 IP가 111.222.333.0/24, 포트 번호는 80을 사용하여 전송하는 패킷 중, 해당 content가 페이로드의 4번째 바이트부터 20번째 바이트 사이에 존재하는 지를 검사한다. content의 위치정보와 헤더정보를 포함시킴으로써 규칙의 오탐 가능성을 낮출 수 있을 뿐만 아니라 상대적으로 수행 오버헤드가 큰 페이로드 검사량을 줄임으로써 시스템 성능 향상에도 도움을 준다.

추출한 content의 위치 정보를 분석하기 위해 트래픽 수집 단계에서 생성한 패킷 데이터를 사용한다. Snort는 패킷 단위로 동작하기 때문에 content의 위치는 sequence가 아닌 실제 패킷 페이로드 내의 위치로 분석해야 한다.

```
alert any any any → any any (content: "CMD";)
```

그림 3. Snort 규칙 예(content 정보만 포함)
Fig. 3. The example of Snort rule (only content information)

```
alert tcp any any → 111.222.333.0/24 80 (content: "CMD"; offset:4; depth:20;)
```

그림 4. Snort 규칙 예(content, 위치, 헤더 정보 포함)
Fig. 4. The example of Snort rule (include content, location and header information)

Algorithm 3 은 content 와 packetSet이 주어졌을 때, content의 위치 정보를 분석하는 과정을 나타낸다. 본 알고리즘의 출력인 offset은 해당 content가 packetSet의 패킷에 매칭 될 때, 매칭 시작 위치의 최소 바이트 위치를 의미하고 depth는 매칭 종료 위치의 최대 바이트 위치를 의미한다. 즉, 해당 content가 패킷에 매칭 될 때, 페이로드의 offset과 depth사이에서만 매칭된다는 의미이다.

Input : content, packetSet = $\{P_1, P_2, ..., P_p\}$
Output : offset, depth

```
analysisContentLocation(content, packetSet)
1:  offset = MAX_PACKET_SIZE;
2:  depth = 0;
3:  foreach packet t in packetSet do
4:    if(t.isMatchContent(content))then
5:      offset=min(offset, t.getStartMatch(content));
6:      depth=max(depth, t.getEndMatch(content));
7:    end
8:  end
9:  return offset, depth;
```

*Notation : offset: min byte point beginning match, depth: max byte point ending match

알고리즘. 3. 위치 정보 추출 알고리즘
Algorithm. 3. The locate information extract Algorithm

최초 offset은 패킷의 최대 크기로 depth는 0으로 초기화 한다(Algo.3 Line: 1~2). 그리고 packetSet의 모든 패킷을 순회하며 offset과 depth를 조정한다. 입력으로 받은 content와 패킷의 매칭 여부를 확인하고, 만약 매칭이 된다면 시작 바이트 위치를 얻어와 현재 offset과 비교한다. 만약, 현재 offset보다 작은 값이면 해당 값은 현재 offset으로 변경한다. depth의 경우 종료 바이트 위치를 얻어와 현재 depth보다 큰 값이면 해당 값을 현재 depth로 변경한다(Algo.3 Line: 4~6). 최종 결정된 offset과 depth값을 content 규칙에 추가한다.

추출한 content의 헤더 정보를 분석하기 위해 앞서 설명한 위치 정보 분석 단계와 유사한 과정을 수행한다. packetSet의 모든 패킷을 순회하며, 해당 content와 매칭 여부를 확인한다. 만약 매칭이 된다면 해당 패킷의 헤더 정보를 저장한다. 모든 패킷을 검사한 후, 저장된 헤더 정보가 고유한 한가지 값을 가지는 경우, 해당 헤더 정보를 content 규칙에 추가한다. 단, IP의 경우 CIDR값을 32, 24, 16 순으로 감소시키면서 고유한 값이 추출될 때까지 반복한다. 즉, CIDR값이 32인, D클래스 IP로 고유한 값을 찾는 것을 시도하고, 만약 찾지 못하면, CIDR값을 24로 적용하여 C클래스 IP를 찾는다. 예를 들어 해당 content가 매칭되는 Destination IP가 CIDR 32로 “111.222.333.1/32”과 “111.222.333.2/32”이 추출되면, CIDR 24로 설정하고 “111.222.333.0/24”를 추출한다.

Ⅳ. 실험 및 결과

본 장에서는 제안한 알고리즘을 사용하여 Snort content 규칙을 자동 생성한 결과와 해당 규칙을 사용한 Snort 분석 결과를 나타낸다. 또한 최소 지지도, 패킷 개수에 대한 조건을 다양하게 변경하여 결과를 생성하고, 결과를 토대로 본 논문에서 제안한 방법의 장점을 제안한다.

실험은 대표적인 인터넷 응용, 서비스 6종을 선정하여, 해당 응용을 사용할 때 발생하는 트래픽을 수집하였다. 실험 결과의 타당성을 높이기 위해 5대의 서로 다른 호스트에서 트래픽을 수집하였다. 실험에 사용된 트래픽은 트래픽 수집 도구인 Netmonitor와 WireShark[14]를 사용하여 수집한 트래픽을 사용한다. 특히, Netmonitor는 프로세스 별로 트래픽 수집이 가능하기 때문에 특정 응용의 트래픽을 수집해야하는 경우 가장 많이 사용하는 수집 도구이다. 수집된 트래픽을 바이너리 형태의 패킷으로 변환하여, Sequence를 추출한다.

다른 방법론과 비교할 때, LASER[6]와 Smith-Waterman[5,8]방법은 두 문자열의 비교 테이블을 계속해서 만들어야 하는 전처리 과정이 필요하다. 다음과 같은 전처리 과정은 엄청난 오버헤드를 차지할 수 있다. 예를 들면, 10개의 문자열(페이로드)에 대해 공통 문자열을 찾을 때, 45개의 테이블이 만들어야한다. 45개의 테이블에서 공통 문자열을 찾는 백트레킹 작업이 필요하고, 문자열들의 공통 문자열이 존재한다고 가정했을 때, 45개의 같은 문자열이 출력된다. 공통 문자열이 중복되는 문제를 해결해주는 후처리과정 또한 필요하다.

Autosig[7]는 각 문자열을 비교하며 테이블을 만드는 전처리 작업이 존재하지 않지만, 두 개의 문자열을 비교하는 방법이기 때문에 마찬가지로 10개의 문자열이 존재할 때, 45번의 공통 문자열을 찾는 작업이 필요하다. 또한 Autosig방법은 각 문자열에서 윈도우 단위로 나누고, 각 윈도우에서 싱글단위로 나누어 공통 문자열을 찾는다. 따라서 윈도우 및 싱글의 단위를 어떻게 설정해 주는 지에 따라 출력되는 시그니쳐의 결과가 다르기 때문에 객관적인 시그니쳐를 얻을 수 없다.

하지만 본 방법은 10개의 문자열(페이로드)가 있을 때, 10개의 sequence를 바탕으로 공통 문자열을 찾아가는 과정이기 때문에 시스템에 부하를 일으키지 않는다. 또한 사용자가 임의로 정해주는 값은 최소 지지도 값 이외에 없기 때문에 객관적인 시그니쳐를 기대할 수 있다. 가장 큰 차이점 및 장점은 본 방법은 두 개의 문자열을 비교하며 공통 문자열을 찾는 방법이 아닌, 문자열 전체를 비교하며 공통 문자열을 찾는 방법이다. 따라서 10개의 문자열이 존재할 때 45번의 공통 문자열 작업이 필요 없을 뿐만 아니라 두 개의 문자열을 비교하여 출력된 공통 문자열을 따로 후처리하는 과정이 필요 없다.

본 실험에서 분석률은 패킷단위와 플로우 단위로 나누어 나타낸다. 패킷 단위는 각 패킷마다 분석결과를 내어 전체 패킷 중 분석된 패킷의 분석률을 의미한다(Snort결과). 플로우 단위는 하나의 플로우 중 한 개의 패킷이 분석되면 해당 플로우의 속한 패킷 전체가 분석된 플로우의 분석률을 의미한다. 예를 들면 패킷 단위의 분석률은 100개의 패킷 중 80개에 해당하는 규칙이 포함되어 있으면 분석률은 80%를 나타낸다. 플로우 단위의 분석률은 10개의 플로우가 각 10개의 패킷을 포함하고 있을 때, 플로우 당 패킷 하나에서 해당 룰을 포함되어 있으면 플로우 분석률은 100%,

표 2. 생성된 규칙 개수, 분석률 및 예시
Table 2. The number of Generated rules, completeness and example

Name	Rules	Completeness			Rule Example
		Flow	Pkt	Byte	
dropbox	16	91.75%	98.92%	99.96%	alert tcp any 443 → 163.xxx.xxx.0/24 any (sid: 1005195; conten:"|16||03||01||00|1|02||00||00|-|03||01|"; offset:0; depth:11;) alert tcp 163.152.219.0/24 any → any 443 (sid: 1005681; conten:"|16||03||01||00|A|01||00||00|=|03||01|T"; offset:0; depth:12;)
facebook	46	82.73%	97.61%	98.68%	alert tcp any 443 → 163.xxx.xxx.xxx any (sid: 1009836; conten:"|16||03||03||00|^|02||00||00|Z|03||03|"; offset:0; depth:11;) alert tcp 31.13.0.0/16 443 → 163.xxx.xxx.xxx any (sid: 1197426; conten:"|00||c0|+|00||00|2|00||00||00||00||ff||01||00||01||00||00||0b||00||04||03||00||01||02||00|#|00||00|3t|00||19||08|spdy/3.1|06|spdy/3|08|http/1.1|16||03||03||0b||0f||0b||00||0b||0b||00||0b||08||00||04||a6|0|82||04||a2|0|82||03||8a||a0||03||02||01||02||02||10||09||15||ff|Py|ba||a0||ee||01|Z6*|96||e7|T|ff|0|0d||06||09|*|86|H|86||f7||0d||01||01||05||05||00|0f1|0b|0|09||06||03|U|04||06||13||02|US1|15|0|13||06||03|U|04||0a||13||0c|DigiCert Inc1|19|"; offset:43; depth:200;)
skype	112	52.06%	51.82%	63.86%	alert tcp 163.xxx.xxx.0/24 any → any 443 (sid: 1004993; conten:"|c0||13||c0||14||c0|"; offset:54; depth:97;) alert tcp 163.xxx.xxx.0/24 any → any 443 (sid: 1004996; conten:"|c0||09||c0||0a||00|"; offset:58; depth:109;)
torrent	391	89.16%	97.94%	96.90%	alert udp any any → any any (sid: 1033594; conten:"d1:ad2:id20:"; offset:0; depth:12;) alert udp any any → any any (sid: 1183071; conten:"|00||00||00||10||00|"; offset:10; depth:119
tvpot	276	97.96%	99.57%	99.54%	alert tcp any 80 → 163.xxx.xxx.xxx any (sid: 1161279; conten:"HTTP/1.1 206 Partial Content|0d||0a|Cache-Control: max-age=1800|0d||0a|Etag: "; offset:0; depth:65;) alert tcp any any → 163.xxx.xxx.xxx any (sid: 1011360; conten:"9|00||00|T|00||b0|"; offset:0; depth:6;)
youtube	349	95.44%	99.76%	99.82%	alert tcp 163.xxx.xxx.xxx any → any 80 (sid: 1359554; conten:"&fexp=3300113%2C3300133%2C3300137%2C3300161%2C3310366%2C3310705%2C900245%2C908580%2C924613%2C9"; offset:61; depth:190;) alert tcp any 80 → 163.xxx.xxx.xxx any (sid: 1190861; conten:"3 GMT|0d||0a|Cache-Control: private, max-age=21"; offset:133; depth:200;)

패킷 분석률도 100%, 바이트 분석률도 100%로 나타낸다. 각 표에서는 패캣 단위의 분석 결과는 의미하는바가 적기 때문에 플로우 단위의 분석 결과를 나타낸다.

표 2는 최소 지지도를 1로 설정하고 플로우 당 패킷을 하나만 분석했을 때 각 응용별 생성된 규칙의 개수, 생성된 규칙의 분석률과 규칙의 예를 보여준다. 최소 지지도가 1인 것은 모든 호스트 또는 모든 파일에서 나타난 규칙을 의미한다. 대부분의 규칙이 네트워크 사용자가 눈으로 판단하기 어려운 규칙이지만 분석률은 Skype를 제외한 나머지응용의 Byte는 평균적으로 97%를 유지한다. 본 수치는 각 응용으로부터 발생된 트래픽 중 97%를 해당 시스템에서 생성된 룰로 분석할 수 있는 트래픽 양에 대한 비율이다. 분석 안 된 트래픽을 위해 최소지지도 또는 패킷 개수를 변경하며, 최상의 규칙 개수와 분석률을 위해 최적의 조건을 찾는 실험이 필요하다.

최적의 최소지지도를 평가하기 위해, 규칙 생성 시간 및 분석률을 비교해야한다. 최소지지도를 지나치게 낮추면 분석률은 증가하지만, 규칙 생성 시간이 증가한다. 반대로 최소지지도를 지나치게 높이면 규칙 생성 시간은 짧아지지만 분석률은 낮아진다. 따라서 실험은 최적의 최소 지지도를 찾고, 최적의 지지도에 이은 패킷 개수를 평가한다.

표 3은 최소 지지도 변화에 따른 Snort content 규칙 생성 개수와 Byte 분석률을 나타낸 것이다. 최소지지도가 낮아질수록 발생하는 content 규칙의 개수와 분석률은 증가한다. 그러나 규칙의 개수가 많아 지는 만큼 규칙 생성 시간과 분석 시간이 많이 소비된다. 최소지지도가 1에서 0.8사이의 변화를 관찰했을 때 분석률이 증가한다. 하지만 0.8에서 0.6사이의 변화했을 때 분석률 증가는 미비하고 규칙 생성 시간만 증가한다. 따라서 본 결과의 분석 시간에 따라 최적의 최

표 3. 최소지지도에 따른 규칙 개수 및 분석률
Table 3. The number of rules and completeness according to minimum support

	minsupp = 1		minsupp = 0.8		minsupp = 0.6	
	Rule	Completeness	Rule	Completeness	Rule	Completeness
Dropbox	16	99.96%	17	99.96%	24	99.96%
Facebook	46	98.68%	68	99.96%	118	99.96%
Skype	112	63.86%	213	69.00%	298	70.36%
Torrent(1)	555	96.90%	1,582	96.95%	3,801	99.94%
Torrent(2)	391	98.33%	1,131	98.36%	2,399	98.36%
Tvpot	276	99.54%	668	99.54%	1,429	99.54%
Youtube	349	99.82%	923	99.98%	2,075	99.98%

표 4. 패킷 개수에 따른 규칙 개수 및 분석률 (minsupp=0.8)
Table 4. The number of Rules and Completeness according to the number of packets

	packet = 1		packet = 2		packet = 3		packet=unlimited	
	Rule	Completeness	Rule	Completeness	Rule	Completeness	Rule	Completeness
Dropbox	16	99.96%	19	99.96%	20	99.96%	21	99.96%
Facebook	46	98.68%	58	99.97%	67	99.97%	98	99.97%
Skype	112	63.86%	148	65.09%	168	65.18%	188	65.26%
Torrent(1)	555	96.90%	595	96.90%	620	96.90%	758	99.87%
Torrent(2)	391	98.31%	433	98.31%	452	98.31%	488	98.33%
Tvpot	276	99.16%	367	99.54%	527	99.54%	1,530	99.54%
Youtube	349	99.82%	614	99.82%	830	99.98%	1,271	99.98%

소 만족도를 결정할 수 있다. 따라서 최적의 최소지지도를 0.8로 정한다.

그림 5는 하나의 응용(tvpot)을 대상으로 최소 지지도에 따른 규칙 생성 시간과 생성된 규칙으로 트래픽을 분석할 때의 분석률을 나타낸 그림이다. 최소 지지도가 1일 때 생성된 규칙은 약 80개 정도이고, 최소지지도가 0.8일 때 생성된 규칙은 약 120개 정도이다. 대략 40개의 규칙이 더 생성되었다. 분석률 또한 최소지지도가 1일 때보다 0.8일 때 증가한다. 따라서 응용을 실행할 때마다 필수적으로 나오지 않는 규칙도 본 논문에서 제안한 방법은 최소 지지도를 적합하게 설정하면 생성할 수 있다는 장점이 있다.

본 시스템의 효율성을 계산했을 때, 규칙이 많이 생성되면 규칙이 생성되는데 소요되는 시간도 증가하지만, 생성된 규칙으로 네트워크에 적용시키는 작업도 증가한다. 따라서 최소의 개수로 규칙을 생성하고, 최대의 분석률을 얻을 수 있는 최소지지도 및 패킷개수를 정해야한다.

표 4는 최적의 최소 지지도를 0.8로 설정 후 하나의 플로우에서 분석할 패킷 개수 변화에 따른 Snort content 규칙 생성 개수를 나타낸 것이다. 즉, 개수가 1개라면 하나의 플로우에서 forward 패킷 1개와 backward 패킷 1개의 내용으로 규칙을 생성한다.

패킷의 개수가 증가해도 대부분의 규칙이 첫 번째 패킷에서만 생성되기 때문에 분석률은 증가하지 않는다. 그러나 생성되는 규칙의 개수는 증가한다. 따라서 본 실험의 최적 패킷 개수는 1개로 정한다.

결론적으로 본 논문에서 제안한 시스템을 적용함으로써 분석률은 대부분의 응용에서 98%이상 분석결과를 보였으며, 최소지지도 및 패킷개수를 변화하여 기존에 추출할 수 없었던 규칙을 생성할 수 있는 결과를 확인하였다.

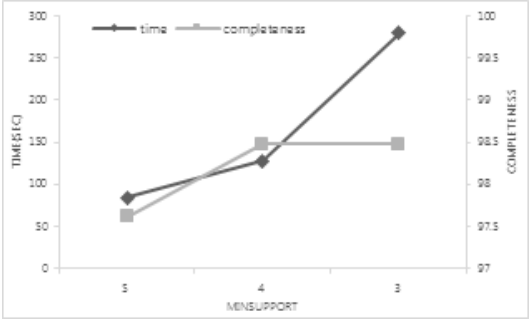

그림 5. 최소지지도에 따른 규칙 생성 시간 및 분석률 변화
Fig. 5. The completeness and the time of rule generation according to minium support

V. 결 론

본 논문에서는 순차 패턴 알고리즘을 사용하여 Snort content 규칙 생성 방법을 제안하였다. 입력된 트래픽에서 관찰되는 공통된 문자열(content)을 추출할 뿐만 아니라 해당 content의 위치 정보 및 헤더 정보를 추가하여 좀 더 정확한 규칙을 생성할 수 있었다. 대표적인 인터넷 응용 6종에 적용하여 제안한 방법의 타당성을 검증하였다.

본 논문에서 제안한 시스템으로 규칙을 생성하고, 생성된 규칙으로 트래픽을 분석했을 때, 특정응용을 제외한 대부분의 응용은 98%이상을 분석할 수 있었다. 또한 최적의 최소 지지도를 변경하여 기존 시그니처 자동 생성 방법으로는 추출할 수 없었던 규칙을 생성할 수 있었다.

향후 연구로는 현재 생성되는 규칙 중 중복되어 나타나는 규칙이 존재하는데 이러한 중복 규칙을 표현하기위해 단일 content가 아닌 다중 content로 구성된 규칙을 자동 생성하는 방법을 개발할 예정이다. 또한 본 시스템을 효율적으로 실시간에 적용하기 위해 더욱더 신속하게 규칙 생성하는 방법을 연구해야 한다.

References

[1] Y. Wang, Y. Xiang, W. L. Zhou, and S. Z. Yu, "Generating regular expression signatures for network traffic classification in trusted network management," *J. Netw. Comput. Appl.,* vol. 35, pp. 992-1000, May 2012.

[2] B. Park, Y. Won, J. Chung, M. S. Kim, and J. W. K. Hong, "Fine-grained traffic classification based on functional separation," *Int. J. Netw. Management,* vol. 23, pp. 350-381, Sept. 2013.

[3] *snort*. Available: https://www.snort.org/

[4] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection," in *USENIX Security Symp.*, vol. 286, 2004.

[5] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," *IEEE Symp. Security and Privacy,* pp. 226-241, 2005.

[6] B.-C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, "Towards automated application signature generation for traffic identification," *IEEE Network Operations and Management Symp. (NOMS 2008)*, pp. 160-167, 2008.

[7] M. Ye, K. Xu, J. Wu, and H. Po, "Autosig-automatically generating signatures for applications," *IEEE Int. Conf. Computer and Inf. Technol.(CIT'09),* pp. 104-109, 2009.

[8] X. Feng, X. Huang, X. Tian, and Y. Ma, "Automatic traffic signature extraction based on smith-waterman algorithm for traffic classification," *IEEE Int. Conf. Broadband Netw. Multimedia Technol. (IC-BNMT),* pp. 154-158, 2010.

[9] C. Mu, X.-h. Huang, X. Tian, Y. Ma, and J.-l. Qi, "Automatic traffic signature extraction based on fixed bit offset algorithm for traffic classification," *The J. China Universities of Posts and Telecommun.,* vol. 18, pp. 79-85, 2011.

[10] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. VLDB*, pp. 487-499, 1994.

[11] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. Eleventh Int. Conf. Data Eng.,* pp. 3-14, *1995.*

[12] C. S. Park, J. S. Park, and M. S. Kim, "Automatic payload signature generation system," *J. KICS*, vol. 38B, no. 08, pp. 615-622, Aug. 2013.

[13] S. H. Yoon, J. S. Park, H. M. An, and M. S. Kim, "Traffic behavior signature extraction using sequence pattern algorithm," in *Proc. KICS Int. Conf. Commun. (KICS ICC 2014)*, pp. 996-997, Jeju Island, Korea, Jun. 2014.

[14] https://www.wireshark.org

[15] https://www.tcpdump.org

[16] R. Srikant, Q. Vu, and R. Agrawal, "Mining association rules with item constraints," *KDD,* vol. 97, pp. 67-73, 1997.

[17] J. H. Park, J. S. Park, and M. S. Kim, "Processing speed improvement of HTTP traffic classification based on hierarchical structure of signature" *J. KICS*, vol. 39B, no. 04, pp. 191-199, Apr. 2014.

심 규 석 (Kyu-Seok Shim)

2014년 : 고려대학교 컴퓨터 정보학과 졸업
2014년~현재 : 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

김 성 민 (Sung-Min Kim)

2014년 : 고려대학교 컴퓨터 정보학과 졸업
2014년~현재 : 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 과니 및 보안, 트래픽 모니터링 및 분석

윤 성 호 (Sung-Ho Yoon)

2009년 : 고려대학교 컴퓨터 정보학과 졸업
2011년 : 고려대학교 컴퓨터정보학과 석사
2011년~현재 : 고려대학교 컴퓨터 정보학과 박사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

정 우 석 (Woo-Suk Jung)

2015년 : 고려대학교 컴퓨터 정보학과 졸업
2015년~현재 : 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

이 수 강 (Su-Kang Lee)

2014년 : 고려대학교 컴퓨터 정보학과 졸업
2014년~현재 : 고려대학교 컴퓨터정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 트래픽 분류

김 명 섭 (Myung-Sup Kim)

1998년 : 포항공과대학교 전자계산학과 졸업
2000년 : 포항공과대학교 컴퓨터공학과 석사
2004년 : 포항공과대학교 컴퓨터공학과 박사
2006년 : Post-Doc. Dept. of ECE, Univ. of Toronto, Canada
2006년~현재 : 고려대학교 컴퓨터정보학과 부교수
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크