클라우드 환경에서 보안 가시성 확보를 위한 자동화된 패킷 분류 및 처리기법

안명호* · 류미현**

*MHR Inc.

Near Realtime Packet Classification & Handling Mechanism for Visualized Security Management in Cloud Environments

Myong-ho Ahn* · Mi-hyeon Ryoo**

*MHR Inc

E-mail : james@netracloud.org, mhr.ryoo@gmail.com

요 약

컴퓨팅 패러다임이 클라우드 컴퓨팅으로 변화됨에 따라 보안 이슈가 더욱 더 중요하게 되었다. 컴퓨팅 플랫폼 서비스 제공자들은 Firewall, Identity Management 등을 제공하지만 클라우드 컴퓨팅 인프라는 사용자가 맘대로 제어하거나 새로운 장치들을 부착하여 사용할 수 없기 때문에 비교적 보안에 취약한 것이 사실이다. 이런 환경에서는 사용자 스스로 보안을 대비해야 하기 때문에 직관적인 방법으로 전체 네트워크 트래픽 상황을 가시적으로 조망할 수 있는 기법이 필요하다. 이를 위해서는 네트워크 패킷을 실시간으로 저장하고, 저장된 데이터를 준 실시간으로 분류할 수 있는 기술이 요구된다. 네트워크 패킷 분류에서 중요한 사항은 패킷 분류를 비지도 방식으로 사람의 개입 없이도 판단 기준을 지능적으로 생성하고 이를 통해 패킷을 스스로 판별할 수 있는 기술개발이 필요하다. 또한, 이를 위해서 Naive-Bayesian Classifier, Packet Chunking 등의 방법들을 활용해 사용자 개입없이 분류에 필요한 시그니쳐(Signature)를 탐색하고 이를 학습해 스스로 자동화된 패킷 분류를 실현할 수 있는 방안을 제시한다.

ABSTRACT

Paradigm shift to cloud computing has increased the importance of security. Even though public cloud computing providers such as Amazon, already provides security related service like firewall and identity management services, it is not suitable to protect data in cloud environments. Because in public cloud computing environments do not allow to use client's own security solution nor equipments. In this environments, user are supposed to do something to enhance security by their hands, so the needs of visualized security management arises. To implement visualized security management, developing near realtime data handling & packet classification mechanisms are crucial. The key technical challenges in packet classification is how to classify packet in the manner of unsupervised way without human interactions. To achieve the goal, this paper presents automated packet classification mechanism based on naive-bayesian and packet Chunking techniques, which can identify signature and does machine learning by itself without human intervention.

키워드

클라우드컴퓨팅, 어플리케이션, 보안, 패킷, 분류

Ⅰ. 서 론

[1]클라우드 컴퓨팅이 새로운 컴퓨팅 패러다임으로 자리매김을 함에 따라 IT 환경에도 많은 변화가 발생하게 되었다. 특히 보안 이슈는 사용자, 제공자 모두에게 민감한 부분이자 신뢰성의 문제이다. 더불어 IT 서비스개발에 대한 문화도 이전의 컴퓨팅 패러다임과 확연히 다르다.

이러한 새로운 개발문화는 [2]"DevOps"라는 새로운 용어로 표현되는데 이전에는 "개발 != 운영"이 서로 별개의 것으로 개발자와 운영자의 역할이 상호 분리가 되었다면, 클라우드 컴퓨팅환경에서는 "개발 = 운영"을 동일한 선상에서 처리하기 때문에 개발자가 운영자 없이 서비스개발과 운영의 전반에 모두 개입하는 것이 가능하다.

이러한 클라우드 컴퓨팅 환경에서는 DevOps 개발 문화에 적합한 새로운 보안 방법이 필요하다.

새로운 보안 방법은 1) 개발자의 부담을 덜어줄 수 있도록 자동화로 사람의 개입을 최소화해야 하며, 2) 직관적으로 네트워크 보안상황을 알 수 있는 네트워크 트래픽 시각화 기술, 3) 빠른 상황 파악과 대처를 위해 매우 빠른 처리 속도를 필요로 한다.

본 논문에서는 새로운 보안 방법에 핵심적인 역할을 하는 네트워크 패킷 자동분류 매커니즘을 소개하고자 한다. 제안하려는 매커니즘은 Unsupervised Learning 기법을 도입해, 기존의 패킷 분류 방법처럼 사용자가 패턴 혹은 시그니쳐를 제공할 필요 없이, 매커니즘 자체에서 필요한 시그니쳐를 스스로 추출하고, 이를 지속적으로 학습해 패킷 분류 성공률을 높이는 방법이다.

본 논문에서는 클라우드 컴퓨팅 환경에서 가장 많이 사용되고 있으며, 외부로 노출되어 보안성에 취약한 HTTP 프로토콜의 자동화된 패킷 분류 방법을 제시하고자 한다.

Ⅱ. 선행연구의 고찰

일반적으로 패킷 분류를 하는 방법은 크게 1) 패턴 혹은 시그니쳐 기반의 방식과 2) 학습기반의 분류 방식으로 나누어 볼 수 있다.

전자의 방식은 패턴의 특징을 추출하여 이를 시그니쳐로 등록해 패킷 분류에 해당 시그니쳐의 존재 유무를 확인하는 것으로 잘 정의된 시그니쳐가 있다면 빠른 속도로, 일정 수준 이상의 패킷 성공률을 보장해 줄 수 있어 가장 많이 활용되는 방법이다.

후자의 방식은 패킷에서 시그니쳐를 학습알고리즘에 의해 스스로 찾고 이를 이용해 패킷을 분류하는 방법으로 사용자가 패킷 분류에 필요한 시그니쳐를 직접 찾아서 이를 알고리즘에 적용시킬 필요가 없다. 이것은 패킷이 복잡하거나 시그니쳐를 찾는 것이 쉽지 않을 때 적용할 수 있다.

[3]Kim, Claffy은 포트번호 기반의 패킷 분류방법을 제안했다. 일반적으로 사용하는 포트번호와 프로토콜 타입을 매칭하여 분류하는 방식으로 포트 80을 사용하는 패킷은 HTTP로 분류하는 방식이다. 하지만 이런 방식은 포트번호가 동적으로 바뀌거나 혹은 보안상의 이유로 포트번호를 바꾸는 경우에는 효과적이지 않다.

[4]D. Adami는 DPI(Deep Packet Inspection)에 기반한 패킷 분류방식을 제안했다. 이 방식은 Regular Expression을 적용하는 방식과 유사하게 Pattern Matching으로 분류하는 방식으로 포트번호처럼 고정된 값에 기반한 것이 아니기 때문에 포트번호의 동적인 변동 등에 적용할 수 있다. 하지만 이 방식은 정확도가 50-70% 정도로 그리 높지 않기 때문에 실제 적용하기에는 무리가 있다.

[5]D. Barman은 Machine Learning 기반의 패킷 분류방법을 제안하였다. 이들은 [6]WEKA를 사용해 기본 설정 값만으로 패킷을 학습시키고, 이를 적용했기 때문에 파라메터 설정을 최적화하면 보다 높은 정확도를 기대할 수 있다. Machine Learning에 기반한 방법은 초기 학습을 위한 데이터셋의 선정에 따라 정확도가 좌우되고, 학습시간이 오래 걸리며 그리고 추후 업데이트가 어렵다는 점이 단점으로 작용한다.

Ⅲ. Automated Packet Classification by Packet Chunking and Unsupervised Learning

본 논문에서는 Packet Chunking과 Unsupervised Learning을 결합한 자동화된 패킷 분류방법을 제안한다. APCPCUL(Automated Packet Classification by Packet Chunking and Unsupervised Learning) 방법은 시그니쳐 방식과 Learning Machine 방법을 결합하여 사용하는 방법으로 시그니쳐를 자동으로 추출하고, 추출된 시그니쳐를 Naive-Bayesian으로 학습시켜 패킷을 분류한다. 본 논문의 방식은 정확도 높은 시그니쳐를 추출하기 위해 여러 HTTP 패킷들의 연관성을 분석해 HTTP Request, HTTP Response와 같이 하나의 HTTP Stream으로 묶을 수 있는 HTTP 패킷들을 찾아낸 다음, 이들을 Packet Chunk로 그루핑을 한다. 이렇게 묶여진 패킷들은 동일한 웹 어플리케이션에 의해 발생한 패킷이기 때문에 해당 웹 어플리케이션들에 대한 시그니쳐를 추출할 때 묶여진 Packet Chunk들에 포함되어 있는 HTTP Header 정보,

TCP/IP 정보를 모두 활용하는 것이 가능해진다. 기존의 패킷분류방식에서는 하나의 패킷을 대상으로 시그니쳐를 추출해야 하기 때문에 해당 패킷의 특성을 명확히 나타내줄 수 있는 값에 대한 신뢰도의 평가가 쉽지 않았다. 하지만 제안방식에는 추출된 시그니쳐 후보군의 특성이 Packet Chunk에 모두 나타나는 것인지 아닌지를 파악할 수 있고 이를 확률적으로 처리할 수 있어 정확도를 높일 수 있다.

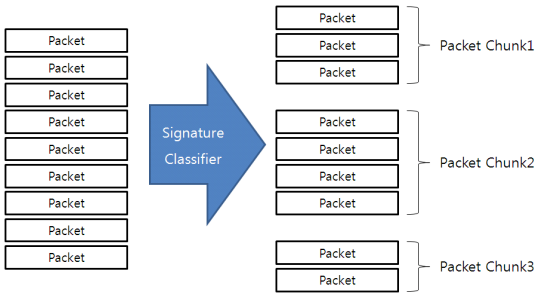

그림 1. Packet Chunking 개념도

Packet Chunk를 만들고 난 후에 시그니쳐를 추출하면 이 시그니쳐를 Naive-Bayesian Classifier에 학습을 시켜야 하는데, 학습을 위한 데이터셋의 신뢰도를 높이기 위해 모든 패킷을 학습시키지 않고 시그니쳐의 정확도가 높게 보장되는 패킷을 선택한다. HTTP Request 패킷은 host, referer값을 가지고 있어 Naive-Bayesian Classifier 학습에 좋은 데이터가 될 수 있다. 이렇게 학습된 Classifier는 Packet Chunk에 적용되어 실제 분류할 패킷에 적용해 패킷을 분류한다.

제안방법의 장점은 다음과 같다.

- Unsupervised Learning으로 사용자가 별도의 데이터셋을 제공할 필요 없음
- 실시간으로 Packet Classifier를 학습시킬 수 있어 새로운 패킷을 자동으로 학습할 수 있음
- Naive-Bayesian 기반으로 동작속도가 기존의 Machine Learning에 비해 빠름
- Payload 정보 없이 Header, TCP/IP 정보만으로 학습할 수 있어 암호화된 HTTP 프로토콜 분석에도 사용할 수 있음
- 연관된 HTTP 패킷을 Packet Chunk로 묶고, 묶여진 Chunk에서 시그니쳐를 추출하여 보다 신뢰성 높은 시그니쳐를 생성할 수 있음

1. APCPCUL의 처리 흐름도

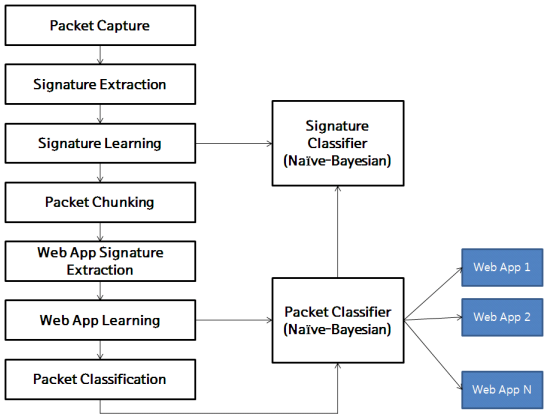

그림 2. 패킷분류 처리흐름도

APCPCUL에서 패킷분류를 위한 단계는 위의 그림과 같이 7단계로 나눌 수 있다.

Packet Capture 단계에서는 학습과 분류에 사용할 패킷을 수집한다.

Signature Extraction 단계에서는 주어진 패킷들을 분석하여 Packet Chunk를 구성하기 위해 필요한 특성을 가지고 있는 패킷들을 추출하고 이들 패킷에서 시그니쳐를 찾는 단계이다.

Signature Learning 단계는 추출된 시그니쳐들을 Signature Classifier에 학습시킨다.

Packet Chunking 단계는 Signature Classifier를 이용해 HTTP 패킷들을 분류해 Packet Chunk로 묶는 과정이다.

Web App Signature Extract 단계는 묶여진 Packet Chunk에서 웹 어플리케이션의 시그니쳐를 추출하는 과정이다.

Web App Learning단계는 전 단계에서 추출된 웹 어플리케이션 시그니쳐를 Packet Classifier에 학습시키는 단계이다.

마지막인 Packet Classification은 분류할 대상인 패킷을 앞서 학습시킨 Signature Classifier와 Packet Classifier를 이용해 패킷을 분류하는 단계이다.

2. Packet Chunking을 위한 시그니쳐 추출

Packet Chunking은 무작위로 섞여 있는 여러 패킷들을 분석하여 서로 연관성 있는 패킷들로 묶어주는 기술로 패킷분류의 정확도를 향상 시키는데 중요한 요소이다. 패킷에서 시그니쳐를 찾을 때 가장 어려운 점은 여러 개의 시그니쳐 후보군 중에서 어떤 것이 가장 좋은 효율을 보여줄 것인가를 평가하는 것이 어렵기 때문이다. 다시 얘기하면 시그니쳐 후보군 중의 하나를 선택하여 무작위로 선택한 패킷에 넣어 시험해 보았을 때 그 시험결과가 맞는 것인지 혹은 틀린지를 판단할 수 있어야 해당 시그니쳐 후보의 정확도를 계산할 수 있기 때문이다. 이러한 문제점으로 인해 실

제 적용되고 있는 패킷 분류기들은 사용자가 시그니쳐를 입력해 주는 방식을 채택하고 있다.

본 논문에서는 Packet Chunk를 만들기 위해 요구되는 패킷 시그니쳐를 추출하기 위해 IP Protocol의 Source IP, Destination IP, Source Port 그리고 Destination Port를 사용한다.

HTTP 패킷은 크게 HTTP Request와 HTTP Response로 이루어져 있는데 에러가 아닌 정상적인 HTTP 통신이라면 HTTP Request와 HTTP Response가 짝을 이뤄서 존재해야 한다.

아래의 그림처럼 HTTP 프로토콜 자체가 Connectionless 통신방식을 이용하기 때문에 웹브라우져에서 HTTP Request를 Web Server에 전송하면 Web Server에서는 HTTP Request를 분석해 HTTP Response를 보내는 방식으로 이뤄지기 때문에 HTTP Request가 있다면 그에 상응하는 HTTP Response가 있게 된다.

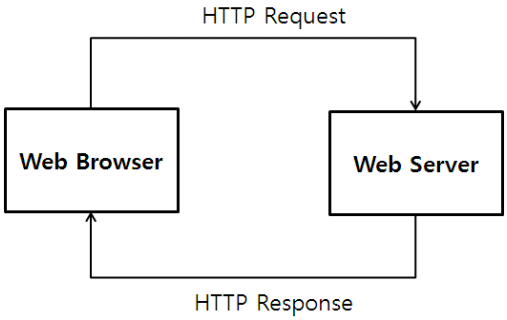

그림 3. HTTP 통신방식

HTTP Response는 HTTP Version, Server, Expires와 같은 항목들을 가지고 있는데 일반적인 내용들이기 때문에 시그니쳐를 추출하기에 적당하지 않다.

```
HTTP/1.1 200 OK₩r₩n Date: Fri, 08 Jul 2011 00:59:41 GMT₩r₩n
Server: Apache/2.2.4 (Unix) PHP/5.2.0₩r₩n
X-Powered-By: PHP/5.2.0₩r₩n Expires: Mon, 26 Jul 1997 05:00:00 GMT₩r₩n
Last-Modified: Fri, 08 Jul 2011 00:59:41 GMT₩r₩n
Cache-Control: no-store, no-cache, must-revalidate₩r₩n
Content-Length: 102₩r₩n
Keep-Alive: timeout=15, max=100₩r₩n
Connection: Keep-Alive₩r₩n
Content-Type: text/html₩r₩n
```

그림 4. HTTP Response 예시

HTTP Request는 웹서버에 데이터를 요청할 때 보내는 것으로 HTTP Method, Encoding, Accept, Host와 같은 데이터를 가지고 있지만 HTTP Response와 마찬가지로 특정 웹 어플리케이션의 특징을 보여줄 수 있는 시그니쳐의 추출에는 적합하지 않다.

```
GET /cgi-bin/http_trace.pl HTTP/1.1₩r₩n
ACCEPT_ENCODING: gzip,deflate,sdch₩r₩n
CONNECTION: keep-alive₩r₩n
ACCEPT: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8₩r₩n
ACCEPT_CHARSET: windows-949,utf-8;q=0.7,*;q=0.3₩r₩n
USER_AGENT: Mozilla/5.0 (X11; Linux i686) AppleWebKit/535.1 (KHTML, like
Gecko) Chrome/13.0.782.24₩r₩n
ACCEPT_LANGUAGE: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4₩rn
HOST: www.joinc.co.kr₩r₩n
```

그림 5. HTTP Request 예시

HTTP 통신방식을 보면 Request와 Response에 대한 것으로 데이터를 주고 받으며 HTTP Protocol은 TCP/IP에 기반하기 때문에 TCP/IP 헤더에서 제공하는 정보를 시그니쳐로 활용하는 것이 가능하다. 왜냐하면 HTTP Request와 HTTP Response는 서로 같은 포트를 공유해야 하기 때문이다.

오프셋	옥텟	0								1								2								3							
옥텟	비트	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved				N S	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N	Window Size														
16	128	Checksum																								Urgent pointer (URG 플래그가 설정된 경우)								
20	160	Options (*data offset* > 5인 경우, 필요시 끝부분에 "0" 바이트로 패딩 됨.)																															
...	...	...																															

그림 6. TCP Header

HTTP를 사용하는 방식은 웹브라우져와 같은 소프트웨어를 이용하거나 API 클라이언트와 같은 것을 이용하는데 웹브라우져의 경우에는 웹 로딩 속도를 빠르게 하기 위해 동시에 여러 개의 HTTP Connection을 가지고 멀티 쓰레딩으로 여러 서버에서 동시에 필요한 데이터를 다운로드 하는 방식도 많이 취하고 있다. 또한 HTTP 통신 방식 자체가 Connectionless이기 때문에 새로운 HTTP Request는 새로운 HTTP Connection을 만들어 처리하는 방식이므로 사용되는 포트가 지속적으로 변화한다. 따라서 이를 처리하기 위한 방법이 필요하다.

3. Naive-Bayesian Classification

Naive-Bayesian Classifier는 Packet Chunking을 위한 시그니쳐의 학습, 패킷을 웹 어플리케이션 별로 분류하기 위한 시그니쳐의 학습에 활용된다. 선행연구들을 보면 Machine Learning에 의한 패킷 분류가 기존의 시그니쳐 기반의 패킷분류보다 우수한 성능을 보여주었다. 특히 새로운 환경, 새로운 프로토콜도 사용자가 시그니쳐를 등

록할 필요 없이 학습을 통해 패킷을 분류할 수 있기 때문에 본 논문에서 지향하는 DevOps 환경에 적합한 방식이라고 할 수 있다. 하지만 Machine Learning 기반의 패킷 분류는 1) 학습용 데이터셋 개발, 2) 속도문제, 3) 지속적 학습이라는 3가지 문제를 가지고 있다.

Naive-Bayesian Classifier의 가장 큰 장점은 데이터 셋을 학습하고 분류하는 속도가 매우 빠르다는 것이다. 지속적 학습에 있어서도 매우 효과적인데 새롭게 학습을 할 때 기존의 학습데이터는 필요치 않고 신규 학습 데이터셋을 이용해 학습시킬 수 있다. 이러한 Naive-Bayesian Classifier의 장점은 앞서 언급한 기존 Machine Learning 기반의 학습이 가지고 있던 문제점들을 해결할 수 있다.

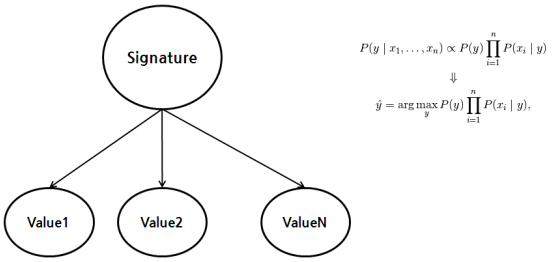

그림 7. Naive-Bayesian Classifier

Naive-Bayesian Classifier을 이용해 패킷을 분류한다는 것은 학습시점에서 주어진 시그니쳐와 연관된 값들에 대한 확률을 계산 후 저장하고, 이 확률 값을 바탕으로 패킷 분류할 때 가장 확률이 높은 시그니쳐를 선택하는 것을 의미한다.

본 논문에서는 사용자의 개입이 없는 자동화된 패킷분류를 위해 Naive-Bayesian Classifier 학습을 위해 필요한 데이터셋을 자동으로 만들 수 있도록 알고리즘을 설계한다.

Naive-Bayesian Classifier를 학습시키는 위해 필요한 데이터의 형태는 "Keyword + Label"이다.

```
('charset 2131 102.123.112.11', 'signature1'),
('gzip apache ibm 2112', ' signature1 '),
('accept encoding utf-8 1105', ' signature3 '),
('132 server nginx', ' signature2'),
('length cache-control 1411', ' signature4'),
```

그림 8. Training Set 예시

Keyword는 시그니쳐의 특성을 나타내는 값들을 나열한 것이고 Label은 시그니쳐의 이름을 의미한다.

Naive-Bayesian Classifier 학습을 위한 데이터셋을 위와 같은 형식으로 자동으로 만들기 위해서는 Packet Chunk를 적극적으로 활용한다.

먼저 Packet Chunk로부터 해당 시그니쳐에 연관된 모든 키워드 후보군을 TCP/IP 헤더, HTTP Request, HTTP Response 헤더로부터 추출한다. 참고로 본 논문에서는 패킷의 페이로드(Payload)에 있는 정보는 활용하지 않는데, 이는 암호화된 데이터가 있을 수 있어 모든 패킷에 적용하기 힘들고, 데이터를 분석하는데 시간이 너무 오래 걸리기 때문이다. 추출된 키워드 후보군들을 통계처리해 각각의 출현빈도를 계산하고 키워드 출현빈도를 정규화하고 정규분포로 만든다.

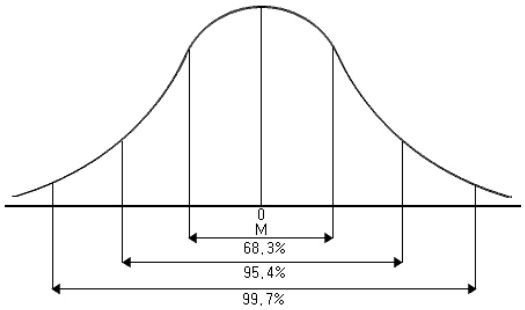

그림 9. 정규분포

키워드 후보군에서 높은 확률을 보여주는 키워드를 선정해서 시그니쳐에 적용한다. 키워드 선정 수에 대한 개수는 제한이 없지만 다른 시그니쳐와의 키워드 중복 여부 체크와 더불어 분류시간에 대한 고려가 필요하다. 지속적인 학습을 통해 Naive-Bayesian Classifier의 정확도를 향상 시키기 때문에 너무 많은 수의 키워드를 시그니쳐에 할당하는 것은 실시간 패킷분류 수행속도에 많은 영향을 미칠 수 있다.

Ⅳ. 평가 및 검증

제안한 APCPCUL(Automated Packet Classification by Packet Chunking and Unsupervised Learning)의 성능을 평가하기 위해 연구실에서 다수의 사용자가 웹브라우져를 통해 웹 어플리케이션을 사용하고 동시에 API Server, Linux Server, NAS Server등이 실제로 동작하는 환경에서 패킷을 수집하였다.

평가에는 Packet Classifier만을 대상으로 하였으며 평가를 위해 18,000개의 패킷을 사용하였다.

제안 방법의 성능을 평가하기 위해 적용하려는 요소들은 다음과 같다.

표 1. 평가요소

학습 데이터셋의 크기	Classifier을 학습시키기 위해 필요한 데이터셋의 크기정도
패킷분류정확도	분류된 패킷의 웹 어플리케이션 정확도
분류시간	패킷분류 갯수에 따른 시간
분류재현율	동일한 분류결과 재현율

1. 패킷분류 정확도

패킷의 수에 따른 분류 정확도를 측정해보면 평균 89.8%의 정확도를 보여주고 있으며 패킷양에 따라 분류 정확도에 5-6%의 차이가 있음을 알 수 있다. 패킷의 수가 1000개인 경우에는 정확도가 81%로 확연히 낮은데 그 이유는 Packet Classifier에 충분한 학습 데이터를 제공하지 못해서이다.

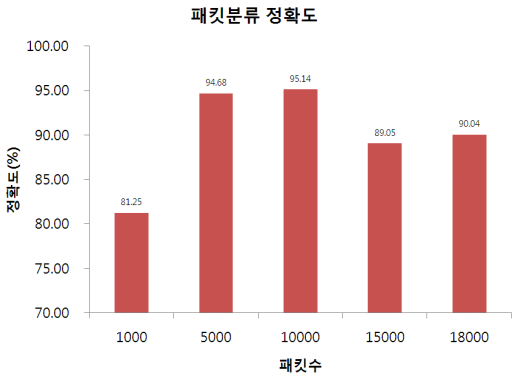

그림 10. 패킷분류정확도

2. Packet Classifier 학습시간

이 항목은 학습데이터셋이 준비되었을 때 Packet Classifier이 제공되는 데이터셋의 크기에 따라 어느 정도의 시간이 소요되는지를 측정한 값이다. 학습시간이 길면 실시간으로 Packet Classifier를 학습시킬 수 없어 패킷분류의 정확도를 지속적으로 향상시키는데 어려움이 있을 수 있다. 18000개의 패킷을 학습시키는데 3.06초가 소요되는 것으로 측정되었다.

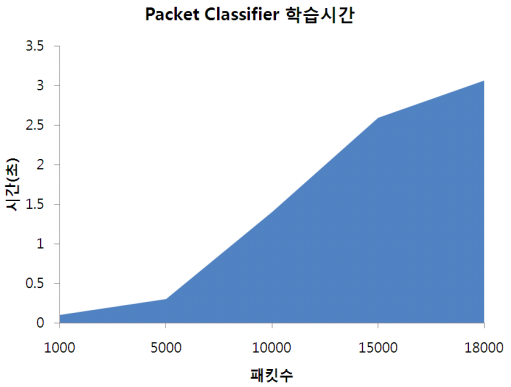

그림 11. Packet Classifier 학습시간

3. 패킷당 분류시간

이 항목은 학습된 Packet Classifier에서 하나의 패킷을 분류하는데 걸리는 시간을 측정한 것으로 평균 0.09초가 소요되며 학습된 패킷의 수에 따라 패킷분류시간이 선형적으로 늘어남을 확인할 수 있다.

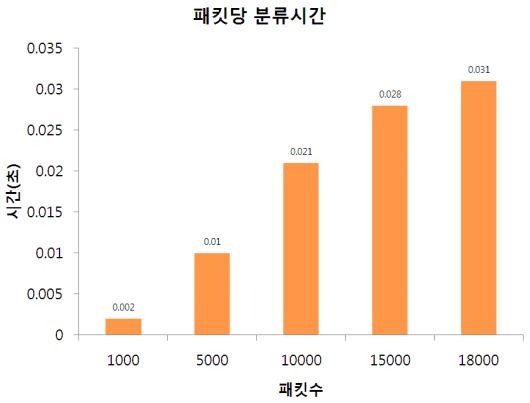

그림 12. 패킷당 분류시간

4. 패킷분류정확도 재현율

재현율은 동일한 조건으로 패킷을 분류하였을 때 패킷분류 정확도가 그대로 유지되는지를 확인하기 위한 항목이다.

동일한 조건으로 1차와 2차에 걸쳐 시험을 해보았는데 100% 일치하는 결과가 나와서 제안 방법이 재현율이 우수함을 알 수 있었다.

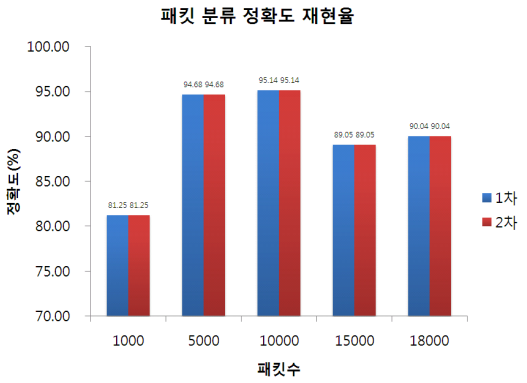

그림 13. 패킷 분류 정확도 재현율

Ⅴ. 결과 및 개선사항

본 논문에서는 APCPCUL(Automated Packet Classification by Packet Chunking and Unsupervised Learning) 방법에 대한 개요와 특징 그리고 구현상의 중요한 사항들을 설명하였는데 몇 가지 개선사항이 있다. 첫 번째는 HTTP 프로토콜에만 적용 가능한 방법이라는 것이다. 제안 방법에 있어 중요한 요소기술인 Packet Chunking은 HTTP에 최적화되어 있어 HTTP가 아닌 다른 프로토콜에 적용하기 위해서는 보다 일반화된 방법이 필요하다. 두 번째는 Naive-Bayesian Classifier의 데이터의 리셋 시점과 방법에 대한 문제이다. APCPCUL은 실시간으로 패킷을 학습해서 분류하는 방식을 취하고 있기 때문에 네트워크 트래픽이 많다면 1-2시간정도만 지나도 학습데이터의 양과 Classifier에서 유지하는 데이터의 크기가 적지 않게 된다. 이는 불필요한 데이터 적재에 따른 성능저하와 정확도에 좋지 않은 영향을 미칠 수 있기 때문에 어느 시점에, 어떤 방법으로 데이터 리셋이 좋은 지를 추가 연구 할 필요가 있다.

APCPCUL은 사용자의 개입 없이 자동으로 빠른 시간 내에 HTTP 프로토콜을 분류할 수 있는 방법이므로 클라우드 DevOps환경에서 사용자가 최소한의 노력으로 준 실시간으로 네트워크 트래픽 상황을 파악할 수 있고, 이를 통해 보안상에 문제가 될 수 있는 패킷을 빠른 시간에 파악하고 대처할 수 있는 방안을 제시했다는데 그 의미가 있다고 생각한다.

참고문헌

[1] http://ko.wikipedia.org/wiki/클라우드컴퓨팅
[2] http://en.wikipedia.org/wiki/DevOps
[3] H. Kim, K.C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K.Y. Lee, Internet traffic classification demystified: Myths, caveats, and the best practices, Proceedings of the 2008 ACM CoNEXT conference, 2008.
[4] D. Adami, C. Callegari, S. Giordano, M. Pagano, and T. Pepe, A Real-Time Algorithm for Skype Traffic Detection and Classification, Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking and Second Conference on Smart Spaces, 2009, p. 179.
[5] D. Barman and M. Faloutsos, Comparison of Internet Traffic Classification Tools.
[6] WEKA: Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka/

안명호(Ahn Myong Ho)

고려대, KAIST SW공학석사
주) MHR 대표이사(현)
*관심분야 : 클라우드 컴퓨팅, 보안, SDN

류미현(Ryoo Mi Hyeon)

고려대,동국대 정보보호석사
주) MHR 수석연구원(현)
*관심분야 : 클라우드 컴퓨팅, 보안, SDN