실시간 트래픽 분석을 위한 운영체제 판별 방법에 관한 연구

준회원 이 현 신*, 종신회원 김 명 섭*

Research on OS fingerprinting Method for Real-time Traffic Analysis System

Hyun-shin Lee* *Associate Member*, Myung-sup Kim* *Lifelong Member*

요 약

현대인의 삶에서 인터넷은 다양한 정보를 제공하는 필수 요소가 되었다. 이에 따라 네트워크 트래픽은 폭발적으로 증가하였고, 효율적 네트워크 관리를 위해 네트워크 트래픽 분석의 중요성이 강조되고 있다. 네트워크 트래픽 분석 방법 중 시그니쳐 기반의 분석 방법론이 많이 사용되고 있으며, 이는 다양한 응용의 증가로 시그니쳐의 탐색 공간 증가에 따라 실시간 트래픽 분석에 문제점이 나타나고 있다. 운영체제 정보를 활용하면 해당 운영체제에서 사용되는 응용의 시그니쳐만을 검색하도록 함으로써 앞선 문제점을 해결할 수 있다. 본 논문에서는 실시간 트래픽 분석을 위한 효과적인 운영체제 판별 방법을 제안한다. 본 논문에서 제안하는 운영체제 판별 방법은 실시간 트래픽 분석에 적합한 가볍고 빠른 수동형 방법을 사용하였으며, 기존 수동형 방법이 가지는 분석률과 정확성이 낮은 문제를 해결하였다. 분석률을 향상시키기 위해 입력데이터를 확장시켰으며, 운영체제 판별을 위한 시그니쳐를 HTTP의 User-Agent를 이용하여 생성함으로써 다양한 운영체제에 대한 시그니쳐를 추출하였다. 또한 기존 패킷 기반의 분석을 플로우 단위의 분석으로 변경함으로써 정확성을 향상시켰다.

Key Words : OS Fingerprinting, Traffic Identification, Real-time Traffic Analysis, Pre-Processing

ABSTRACT

The Internet has become an essential part in our modern life by providing useful information. So, the volume of Internent traffic has been increasing rapidly, which emphasizes the importance of network traffic analysis for effective network operation and management. Signature based analysis have been commonly used, but it is shown that the increase of signatures due to the increase of applications causes the performance degradation of real-time traffic analysis on high-speed network links. In this paper, we propose OS fingerprinting method for real-time traffic analysis. The previous problems can be solved by utilizing the OS information. The OS fingerprinting method for real-time traffic analysis, proposed in this paper, conducts under passive mode, and improves the limitation of a previous method such as low completeness and accuracy. In this paper, we enlarged an input data to improve completeness, and used the User-Agent field in HTTP packet to extract various OS signatures. Also, we changed an input data from packet to flow to improve accuracy.

※ 이 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단(2009-0090455)의 지원을 받아 수행된 연구임.

* 고려대학교 컴퓨터정보학과 네트워크관리연구실(hyunshin-lee@korea.ac.kr, tmskim@korea.ac.kr)

논문번호 : KICS2011-03-137, 접수일자 : 2011년 3월 4일, 최종논문접수일자 : 2011년 5월 2일

Ⅰ. 서 론

인터넷은 방송과 통신의 융합, 다양한 유무선 네트워크의 결합, 다양한 서비스 및 응용 프로그램들의 개발, 사용자 요구사항의 다양화 등 현대인의 삶에서 필요한 정보를 제공함으로써 없어서는 안될 필수 요소가 되고 있다. 그러한 추세로 인하여 네트워크 트래픽 사용량은 폭발적으로 증가하고 있다. 네트워크 트래픽 사용량 증가는 네트워크의 병목현상 및 과부하를 유발하여 네트워크의 안정적인 운용과 서비스 제공을 어렵게 한다. 따라서 네트워크의 안정적인 운용과 정확한 현황 파악을 위해서 다양한 네트워크 트래픽 분석은 필수적이다[1].

네트워크 트래픽 분석은 능동형(Active) 방법과 수동형(Passive) 방법으로 이루어 진다.

능동형 방법은 관리자가 직접 조사 패킷을 생성시켜 대상 네트워크에 주입함으로써 네트워크의 성능 및 트래픽 현황을 측정하는 방법이다. Ping, traceroute 등이 대표적인 능동형 방법이다. 그러나 능동형 방법은 패킷을 생성하여 그에 대한 응답 패킷을 분석함으로써 단말의 증가와 같이 네트워크의 부하도 증가한다는 문제와 응답에 대한 지연시간으로 인하여 실시간 처리가 어렵다는 문제를 가지고 있다.

능동형 방법이 가지는 문제를 해결할 수 있는 수동형 방법은 특정 구간에서 패킷을 수집하여 직접 분석하는 방법이다. 대표적인 방법으로 페이로드 분석 방법과 포트 분석 방법, 통계적 분석 방법 등 다양한 방법들이 존재한다. 수동형 방법은 네트워크의 부하를 증가시키지 않는 장점을 가지고 있지만, 대표적인 페이로드 분석 방법론은 페이로드가 존재하지 않거나, 페이로드를 암호화한 트래픽에 대해서는 분석을 하지 못한다. 또한 응용의 증가에 따른 시그니쳐 수의 증가는 탐색 공간에 대한 분석 시스템의 부하를 증가시켜 실시간 트래픽 분석에 있어서 어려움을 가지고 있다.

이러한 단점을 극복하고자 멀티 레벨 분석 방법론[2]-[4]에 관한 연구가 진행되었다. 그러나 멀티 레벨 분석 방법론 또한 다양한 분석 방법론들의 시그니쳐 탐색 공간의 증가에 대한 문제는 해결하지 못하였으며, 포터블 기기에서 발생한 트래픽들로 인하여 모든 트래픽에 대한 분석을 하지 못하는 한계를 나타내었다.

본 논문에서는 실시간 트래픽 분석을 위한 운영체제 판별 방법론을 제안한다.

운영체제 판별 정보를 실시간 트래픽 분석 방법론에 제공함으로써 운영체제를 기준으로 시그니쳐를 계층적으로 재배치한다. 재배치된 시그니쳐는 탐색공간을 축소시켜 시그니쳐 증가로 인해 발생한 탐색 공간의 부하를 감소시킬 수 있다. 예를 들어 대상 단말이 스마트폰의 운영체제를 사용하는 경우 해당 단말의 트래픽을 스마트폰 응용 프로그램들의 시그니쳐들 중에서 검색함으로써 탐색 공간을 감소시킬 수 있다.

운영체제 판별 방법은 네트워크 트래픽 분석 방법론과 마찬가지로 능동형 방법과 수동형 방법으로 이루어 진다.

전처리 과정에서 수행될 운영체제 판별 방법을 선정하기 위해서 본 논문에서는 선행 연구의 운영체제 판별 방법론들의 수행 시간을 측정하여 선정하였다. 또한 운영체제 판별 결과에 대한 분석률과 정확성을 향상시키기 위해서 필요한 새로운 방법을 제시하였다.

본 논문에서 제안하는 운영체제 판별 방법은 실시간 트래픽 분석에 적합한 가볍고 빠른 수동형 방법을 사용하였으며, 기존 수동형 방법이 가지는 분석률과 정확성이 낮은 문제를 해결하였다. 분석률을 향상시키기 위해 입력데이터를 확장시켰으며, 운영체제 판별을 위한 시그니쳐를 HTTP의 User-Agent를 이용하여 생성함으로써 다양한 운영체제에 대한 시그니쳐를 추출하였다. 또한 기존 패킷 기반의 분석을 플로우 단위의 분석으로 변경함으로써 정확성을 향상시켰다. 그리고 제안하는 방법의 타당성을 검증하기 위해서 학내 망을 이용하여 그 성능을 검증하였다.

본 논문의 구성은 다음과 같다. 2장에서 선행 연구의 운영체제 판별 방법론을 기술하고 문제점을 기술한다. 3장에서는 본 논문에서 제안하는 방법론을 기술한다. 4장에서는 제안한 방법론의 성능을 평가하고 마지막으로 5장에서는 결론을 맺는다.

Ⅱ. 관련연구

본 장에서는 운영체제 판별의 정의를 기술하고 트래픽 분석을 위해서 운영체제 정보가 필요한 이유와 운영체제 판별 방법이 트래픽 분석에 이용되기 위한 고려사항에 대해서 기술한다. 또한 선행 연구의 운영체제 판별 방법들에 대해 기술하고 실시간 트래픽 분석을 위한 운영체제 판별 방법을 선정한다.

2.1 OS Fingerprinting

운영체제 판별(OS Fingerprinting)이란 원격지에 위치한 단말의 운영체제를 패킷의 헤더 정보와 페이로드 정보를 이용하여 판단하는 것[5]이다.

2.2 운영체제 판별 방법의 고려사항

실시간 트래픽 분석을 위한 운영체제 판별 방법은 전처리 과정에 해당한다. 따라서 다음과 같이 2가지 고려사항을 만족하여야 한다.

2.2.1 빠른 수행 시간과 적은 부하

빠른 수행 시간을 통하여 필요한 정보만을 트래픽 분석 과정에 전달하여야 한다. 운영체제 판별 정보를 분석하기 위해 많은 시간이 소요된다면 실시간 트래픽은 처리를 못하게 된다. 이러한 이유에서 실시간 트래픽 분석을 위한 운영체제 판별 방법은 많은 부하를 발생시키면 안 된다.

2.2.2 정확한 결과를 제공하여야 한다.

운영체제 판별 정보를 기반으로 응용에 대한 시그니쳐를 재배치하였을 경우, 운영체제 정보가 잘못 제공되면 해당 트래픽을 분석하지 못하게 된다. 따라서 실시간 트래픽 분석을 위한 운영체제 판별 방법은 완전하게 신뢰할 수 있는 운영체제 판별 결과를 트래픽 분석 과정에 전달하여야 한다.

• 운영체제 판별 방법

운영체제 판별 방법은 크게 능동형 방법과 수동형 방법으로 이루어 진다.

능동형 방법은 조사 패킷에 대한 응답 패킷의 분석을 통해서 이루어지는 방법으로 ICMP, SYN, FIN등과 같은 다양한 패킷을 해당 단말에 전송시켜 그에 대한 응답을 확인하여 운영체제를 추측한다. nmap[6]은 능동형 방법을 이용하는 대표적인 방법이다. 다음 표 1은 nmap에서 운영체제 판별을 위해서 사용되는 항목들을 나열한 것이다.

nmap은 표 1에서 나열한 항목들의 조합을 이용하여 운영체제를 정확하게 판별할 뿐만 아니라, 매칭되지 못하는 조합에 대해서는 추측된 운영체제를 확률로 제시한다. nmap은 표 1에서 제시한 항목 이외에도 운영체제를 판별하기 위하여 단말에서 활성화된 서비스들의 조합도 사용된다. 그러나 조사 패킷에 대한 응답 패킷을 기다려야 하는 지연이 발생하여 실시간 트래픽 분석을 위한 방법으로 적합하지 않다.

수동형 방법은 단말이 발생시킨 트래픽을 원격에서 수집하여 해당 단말의 운영체제를 추측하는 방법이다. 대표적으로 [7]은 수집된 SYN패킷의 TCP/IP 헤더 정보를 이용하여 해당 단말의 운영체제를 추측하는 방법론을 제시하였다. 헤더 정보는 운영체제 별로 다양한 조합의 형태를 취하고 있으며, 운영체제 버전에 따라서도 조합은 변경된다.

표 1. nmap의 운영체제 판별을 위한 feature

No.	Feature
1	The Fin probe
2	The BOGUS flag probe
3	TCP ISN Sampling
4	IPID Sampling
5	TCP Timestamp
6	Don't Fragment bit
7	TCP Initial Window
8	ACK Value
9	ICMP Error Message Quenching
10	ICMP Message Quoting
11	ICMP Error message echoing integrity
12	Type of Service
13	Fragmentation Handling
14	TCP Options
15	Exploit Chronology
16	SYN Flood Resistance

다음 표 2는 [7]에서 이용하는 TCP/IP 헤더 정보를 나열한 것이다.

대표적으로p0f[8]와 SinFP[9]와 같은 도구들이 수동형 방법론을 적용하여 운영체제를 판별한다. p0f와 SinFP는 정확하게 매칭되는(Exact Matching: EM) 운영체제가 없을 경우 nmap과 유사하게 해당 운영체제에 대한 추측 결과를(Approximate Matching: AM) 사용자에게 제공한다.

수동형 방법은 능동형 방법이 비해 수집된 트래픽의 헤더만을 분석하므로 실시간 처리에 적합하지만,

표 2. 운영체제 판별을 위한 TCP/IP 헤더 정보

No	Feature	Description	Header
1	DF	Don't Fragment bit	IP Header
2	iTTL	Initial TTL	
3	NC	Nop Count	TCP Header
4	TS	Timestamp	
5	SACK	SACK Permitted	
6	WSS	Window Scale Size	
7	WSF	Window Scale Factor	
8	WS	Window Size	
9	MSS	Maximum Segment Size	
10	EOL	EOL	

정확성과 분석률 측면에서 능동형 방법에 비해 성능이 낮다라는 단점을 가지고 있다.

• 전처리 과정의 운영체제 판별 방법

전처리 과정에서 사용되는 운영체제 판별 방법은 실시간 트래픽 분석을 위해서 빠른 수행 시간을 가지고 있어야 한다.

운영체제 판별 방법의 수행 시간을 측정하기 위해서 능동형 방법론으로 nmap을 이용하였으며, 수동형 방법론은 p0f를 이용하였다.

표 3은 1개의 단말을 대상으로 운영체제를 판별하는데 소요된 수행 시간을 나타낸다.

능동형 방법은 실제 수행 시간에서 수동형 방법보다 많은 시간이 소요된 것을 확인할 수 있다. 이는 능동형 방법이 해당 단말에 조사 패킷을 전송하여 이에 대한 응답 패킷을 수신하는데 걸리는 시간이 많이 소요되기 때문이다. user 시간과 sys 시간에 대한 real 시간의 차이를 통해서도 확인이 가능하다.

신뢰할 수 있는 정보의 제공이란 측면에서 능동형 방법과 수동형 방법이 가지는 문제점은 다음과 같은 사항이 존재한다.

능동형 방법은 NAT 공유기에 연결된 단말의 운영체제를 판별하지 못하는 단점을 가지고 있다. NAT 공유기를 사용하였을 경우 능동형은 NAT 공유기의 운영체제를 판별하게 된다. 이는 조사 패킷에 대한 응답 패킷을 NAT 공유기에서 전송하기 때문이다.

수동형 방법은 그림 1과 같은 경우에 문제점을 가진다. 윈도우(Windows)를 사용하는 컴퓨터를 가장한 리눅스 기반의 컴퓨터가 발생시킨 스캐닝 공격이 있을 경우 윈도우 컴퓨터를 해커의 운영체제인 리눅스로 판별하게 된다. 이는 스캐닝에 사용되는 SYN 패킷만을 대상으로 분석하기 때문에 생기는 문제이다.

수동형 방법은 분석률과 정확성 문제점도 갖고 있다. 이는 능동형 방법에 비해서 취할 수 있는 정보가 한정적이기 때문이다. 다음 표 4는 수동형 방법을 이용하는 p0f와 SinFP에 대해서 분석률(Completeness)과 정확성(Accuracy)을 측정한 내용이다. 표 4는 분석률과 정확성을 완전 매칭(Exact Matching)과 근사 매칭(Approximate Matching)으로 나누어 나타냈다.

표 3. 운영체제 판별을 위한 수행 시간

Time	능동형(nmap)	수동형(p0f)
User	0.581sec	0.038sec
Sys	0.031sec	0.031sec
Real	22.997sec	0.071sec

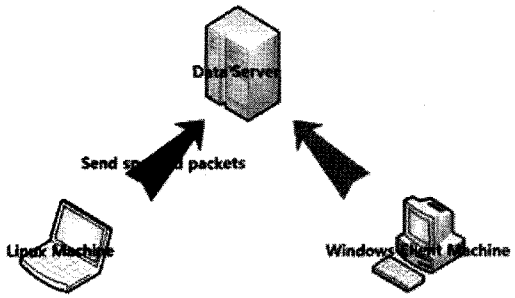


그림 1. 해커와 사용자의 운영체제가 틀린 경우

표 4. p0f와 SinFP의 분석률과 정확성

	Completeness		Accuracy	
	E.M.	A.M.	E.M.	A.M.
p0f	71.32	100%	15.87%	2.87%
SinFP	69.84	100%	31.25%	3.10%

정확성 평가를 위해서 수작업으로 해당 단말의 운영체제를 조사하였으며, NAT공유기를 사용하는 단말에 대해서는 판별을 제외하였다.

표 4에서 확인되는 바와 같이 수동형 방법의 가장 큰 문제점은 분석률과 정확성이 낮다라는 것이다. p0f와 SinFP에서는 분석률 향상을 위해서 근사 매칭을 이용한 운영체제 판별 방법도 제공하지만, 이는 정확성이 떨어지는 문제를 가지고 있다. 또한 현재 많이 사용되는 운영체제에 대한 시그니쳐는 제공되지 않아 분석률과 정확성에 대해서 낮은 결과를 갖는다.

그러나 실시간 트래픽 분석을 위한 운영체제 판별 방법은 수행 시간을 고려하여 수동형 방법이 적합하다. 따라서 본 논문에서는 실시간 트래픽 분석을 위한 운영체제 판별방법으로 수동형 방법을 사용하고, 분석률과 정확성을 향상시키는 방법에 대해서 제안한다.

Ⅲ. 실시간 트래픽 분석을 위한 운영체제 판별 방법

본 장에서는 수동형 방법이 가지는 낮은 분석률과 정확도를 높이기 위해서 본 논문에서 제안하는 방법론에 대해서 기술한다.

3.1 입력데이터의 확장

선행 연구[7]의 수동형 방법에서는 SYN 패킷을 이용하여 운영체제를 판별한다. 이는 단말이 초기에 발생시킨 패킷의 헤더 정보를 통해서 운영체제를 판별

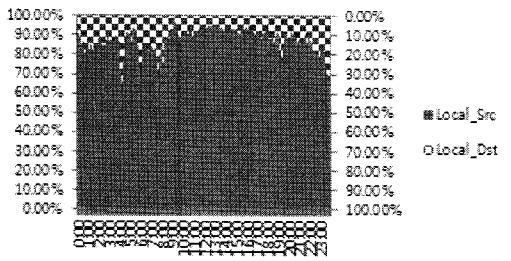


그림 2. SYN/ACK적용에 따른 입력데이터 증가율

하기 때문이다. 그러나 내부 망에서 사용되는 단말이 서버인 경우 분석 대상은 감소하게 된다.

그림 2는 전체 트래픽에 대한 클라이언트와 서버의 비율을 나타낸다. Local_Dst는 SYN패킷을 기준으로 목적지 주소가 내부 망의 단말인 경우이며, Local_Src는 반대의 경우이다. 다시 말해서 Local_Dst는 SYN/AK패킷을 초기 패킷으로 발생시키는 단말이며, Local_Src는 SYN패킷을 초기 패킷으로 발생시키는 단말이다.

그림 2에서 보이는 바와 같이 SYN/ACK패킷을 추가하였을 경우 평균 20%의 분석 대상이 증가되었다. 따라서 본 논문에서는 SYN과 SYN/ACK에 대해서 운영체제를 판별함으로써 분석률을 향상시켰다.

3.2 Feature의 추가

선행 연구[7]는 총 10개의 feature를 사용한다. 그러나 이는 SYN패킷만을 대상으로 하여 본 논문에서 제안하는 방법에는 적합하지 않다.

본 논문에서는 SYN패킷과 SYN/ACK패킷을 입력 데이터로 사용하기 위해서 TCP flag를 feature로 추가하였다. 이는 동일한 운영체제에서 발생한 SYN패킷과 SYN/ACK패킷의 헤더 정보가 다르기 때문이다. 따라서 본 논문에서 사용한 TCP/IP 헤더의 필드 값은 총 11개이다.

3.3 플로우 기반의 분석

선행 연구의 수동형 방법은 패킷 기반의 분석을 통해서 운영체제를 판별한다. 다시 말해서 분석 대상이 되는 단말에서 SYN패킷만 발생이 되어도 해당 단말의 운영체제를 판별한다. 이는 앞서 기술한 스푸핑된 단말에 대해서 운영체제를 잘못 판별하는 문제점을 가지고 있다.

따라서 본 논문에서는 정확성을 향상시키기 위해서 플로우 기반의 운영체제 판별을 한다.

플로우 기반의 방법은 3-way handshake유무를 확인 후 운영체제를 판별한다. 이를 통해서 SYN패킷만 발생한 호스트에 대해서는 분석을 하지 않게 되며, 스푸핑된 단말의 문제점을 해결한다.

3.4 시그니쳐 추출

현재 사용되는 운영체제들은 동일한 운영체제에 대해서도 다양한 TCP/IP헤더 정보의 조합을 나타낸다. 이와 같이 동일 운영체제에 대해서 헤더 정보의 조합이 다양한 이유는 크게 2가지이다.

첫째, 사용자가 설치한 응용 프로그램에 의해 변경되는 경우이다. 일부 네트워크를 사용하는 응용 프로그램은 운영체제의 기본 환경 설정을 변경하여 해당 응용 프로그램이 사용되는 환경을 최적으로 변경하기 위해 운영체제 판별의 기준이 되는 헤더 정보가 변경이 된다

둘째, 사용자에 의해 임의로 변경된 경우이다. 일반적인 환경에서는 헤더 정보를 변경하는 경우가 드물지만, 이러한 경우로 운영체제를 판별하지 못하는 경우가 발생한다. [10]에서는 Windows 운영체제에 대해서 레지스트리 값을 조작하여 헤더 정보를 변경하는 방법을 기술하고 있다. 이는 네트워크 사용을 효율적으로 하기 위해서 이다.

이와 같은 이유로 운영체제에 대한 다양한 시그니쳐를 추출하는 문제점을 가지고 있다. 선행 연구에서는 근사 매칭을 이용하여 앞선 문제점을 해결하려 했으나, 근사 매칭의 결과는 정확성이 낮은 문제를 갖고 있다.

본 논문에서는 시그니쳐 추출 문제를 해결하기 위해서 HTTP 트래픽을 이용한다. HTTP의 User-Agent 항목에는 해당 단말의 운영체제 정보를 포함한다. 이를 이용하여 해당 단말의 운영체제에 대한 TCP/IP 헤더 정보를 시그니쳐로 자동 추출함으로써 다양한 시그니쳐를 통한 정확한 분류를 한다.

또한 HTTP트래픽을 이용한 시그니쳐를 자동으로 추출하였을 경우, 현재 사용되고 있는 다양한 운영체제들의 시그니쳐도 추출이 가능하다. 이는 p0f와 SinFP의 문제점인 최신의 운영체제에 대한 시그니쳐 부재 문제를 해결하였다.

3.5 제안하는 수동형 운영체제 판별 방법

본 절에서는 앞서 기술한 방법을 이용하여 본 논문에서 제안하고자 하는 방법을 기술한다. 그림 3은 본 논문에서 제안하는 방법론의 순서도이다.

본 논문에서 제안하는 방법론은 크게 3가지의 과정으로 나눠진다.

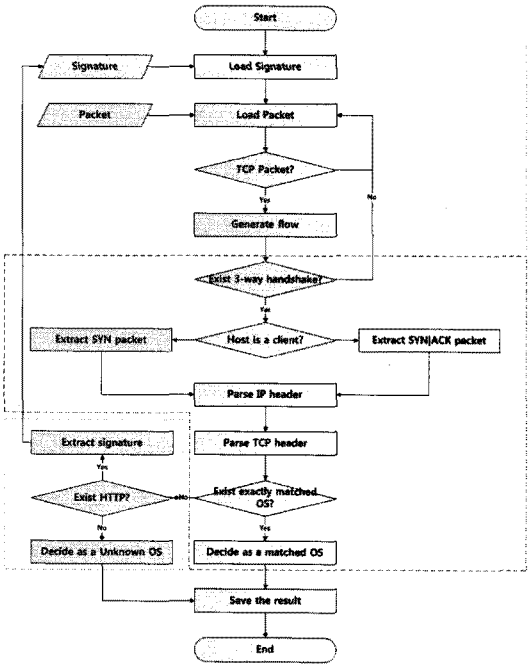


그림 3. 실시간 트래픽 분석을 위한 운영체제 판별 방법론의 흐름도

첫 번째 과정은 시그니쳐와 패킷을 로드한 후 패킷에 대한 플로우를 만들어 주는 과정이며, 이 과정에서 플로우는 SYN과 SYN/ACK를 모두 포함하기 위해서 TCP를 대상으로 하며 양방향이다.

두 번째 과정에서는 3-way handshake 유무를 확인하고 해당 단말에 대해서 클라이언트와 서버를 판별하여 각각에 대한 SYN패킷과 SYN/ACK패킷을 추출한다. 추출된 패킷에서 헤더 정보를 읽어 시그니쳐와의 매칭을 통해서 운영체제를 분류한다.

마지막 과정에서는 운영체제를 분류하지 못한 단말에 대해서 HTTP 트래픽 유무를 확인하여 HTTP 트래픽이 존재할 경우 해당 운영체제에 대한 시그니쳐를 추출한다.

Ⅳ. 실험 및 성능 평가

본 장에서는 성능 평가를 위한 실험 환경과 정답지 생성 방법을 기술하고 본 논문에서 제시한 운영체제 판별 방법에 관한 성능을 평가한다.

4.1 실험 환경

트래픽은 학내 망을 대상으로 수집되었으며, KU-MON[2,11,12]을 이용하였다. 1일 동안의 데이터를 대상으로 하였다. 대상이 된 전체 단말의 수는 2,714개이며, 1분 평균 250MB의 트래픽이 발생하였다.

입력 데이터는 KU-MON을 통해서 수집된 플로우에서 앞서 기술한 스프핑된 단말의 잘못 분류되는 문제를 해결하기 위해서 3-way handshake가 존재하는 플로우를 사용한다.

4.2 정답지

운영체제 판별 방법의 정확성을 측정하기 위해서 사용된 정답지는 HTTP 트래픽의 User-Agent항목을 이용하여 생성하였다.

HTTP의 User-Agent항목은 웹 브라우저의 종류와 운영체제 정보를 가지고 있으며, 입력데이터로 사용된 플로우를 대상으로 생성되었다. 전체 2,714개의 단말 중 2,346개의 단말에 대한 정답지를 생성하였다.

또한 NAT 공유기 사용자에 의한 2개 이상의 운영체제를 가지는 단말에 대해서는 정답지 대상에서 제외하였다.

정답지를 생성하는 단위 시간은 운영체제 판별 방법론의 분석 시간을 기준으로 하기 위해서 1분을 기준으로 하였다.

4.3 성능 평가

분석률은 전체 트래픽을 발생한 단말을 대상으로 해당 운영체제를 분류한 단말의 비율을 나타낸다.

$$Comp. = \frac{N_C}{N_T} \times 100 \quad (1)$$

위 식1은 분석률을 구하는 것으로 N_T는 전체 3-way handshake를 발생시킨 단말의 수이며, N_C는 해당 운영체제를 분류한 단말의 수이다.

정확성은 HTTP트래픽 분석을 통하여 정답을 알고 있는 단말의 수 중에서 정확하게 운영체제를 판별한 단말의 비율을 나타낸다.

$$Acc. = \frac{N_{CT}}{N_{GT}} \times 100 \quad (2)$$

위 식2는 정확성을 구하는 것으로 N_{GT}는 정답을 알고 있는 전체 단말의 수이며, N_{CT}는 정확하게 운영체제를 판별한 단말의 수이다.

다음 표 5는 SYN패킷만을 이용한 분석률을 나타낸다. 표 5의 N_{UC}는 분류되지 못한 단말의 수를 나타낸다. Total은 테스트 트래픽에 나타난 모든 단말을 대상으로 한 것이고 GT는 HTTP 트래픽 분석을 통하

표 5. SYN 패킷만을 이용한 분석률

	HOST	N_C	N_{UC}	Comp.
N_T	2,714	2,080	634	76.64%
N_{GT}	1,619	1,406	213	86.84%

여 정답을 알고 있는 단말을 대상으로 측정한 것이다.

다음 표 6은 SYN/ACK 패킷을 포함하여 분석한 결과이다. SYN/ACK 패킷에 대한 입력데이터의 확장은 약 16%의 분석률을 향상시켰다. 또한 미분류 단말에 대한 수도 감소하였다. 그러나 정답지가 있는 95개의 미분류 단말을 분류하기 위해서 추가적인 방법이 요구된다.

선행 연구에서는 근사 매칭 방법의 사용에 있어 의사 결정 트리(Decision tree, j48)와 근접 이웃 방법(Nearest-Neighborhood)을 제시하였다. 다음 표 7은 각각에 대한 정확성을 나타낸 것이다. 분석률을 표기하지 않은 이유는 분석 하지 못한 단말에 대해서는 근사 매칭을 적용하는 것이므로 무의미한 성능 지표이기 때문이다.

의사 결정 트리가 약 87%로 근사 매칭에서 높은 결과를 가지고 있으나 이는 판별 결과를 기반 정보로 사용해야 하는 트래픽 분석 과정에서는 낮은 정확성이다. 따라서 시그니쳐 추가를 통한 완전 매칭만을 사용한다.

그림 4는 시그니쳐 추가를 위해서 정답을 알고 있고 동일한 헤더 정보를 가지는 단말끼리 그룹을 지은 결과를 나타낸다. 그룹8번과 17, 24, 31번의 경우 단말의 수가 각각 12, 22, 10, 5개로 총 95개의 미분류 단말에 대해서 약 52%를 차지한다. 또한 모든 그룹의 정답은 동일한 운영체제에 대한 정답을 가지고 있으므로 시그니쳐 추가를 통해서 분석률을 향상시킬 수 있다.

다음 표 8은 시그니쳐 추가를 통해 향상된 분석률

표 6. SYN과 SYN/ACK를 이용한 분석률

	HOST	NC	NUC	Comp.
NT	2,714	2,511	203	92.52%
NGT	2,346	2,251	95	95.95%

표 7. 근사 매칭의 정확성 비교

	Desision Tree(j48)	Nearest-Neighborhood
Acc.	87.37%	55.40%

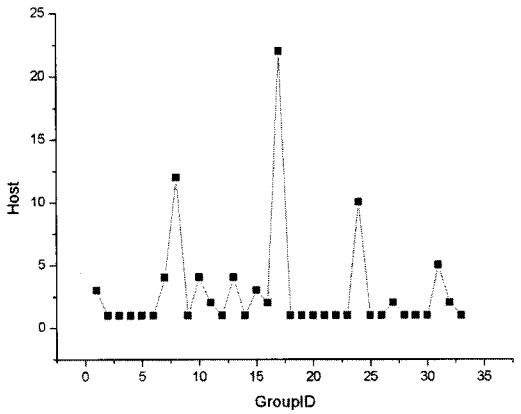


그림 4. 미분류 단말에 대한 그룹핑 결과

을 나타낸다. 시그니쳐의 추가는 전체 분석률의 약 4% 향상을 가져왔으며, 정답이 존재하는 모든 단말에 대해서는 100% 정확하게 운영체제를 판별할 수 있었다.

표 9는 본 논문에서 제안하는 방법의 분석률과 정확성을 p0f와의 비교한 것이다. 분석률 측면에서는 약 23% 이상의 성능이 향상되었으며, 정확성은 100%를 만족한다. 표 9의 전체 단말 중 분석하지 못한 108개의 단말(N_{CF})이 존재하는 이유는 해당 단말에서 HTTP 트래픽을 발생시키지 않아 분석을 하지 못하였다. 그러나 이 단말에서 발생한 TCP/IP 헤더 정보와 동일한 HTTP 트래픽이 발생한다면 시그니쳐의 추가를 통해서 모든 단말에 대한 분석이 가능하다.

표 8. 시그니쳐 추가를 시행한 분석률

	Host	NC	NUC	Comp.
NT	2,714	2,606	108	96.02%
NGT	2,346	2,346	0	100%

표 9. p0f와의 성능 비교 및 평가

	Proposal Method	p0f
NT	2,714	2,714
NGT	2,346	2,346
NC	2,606	1,990
NCF	108	724
NCT	2,346	316
Comp.	96.02%	73.32%
Acc.	100%	15.87%

V. 결론 및 향후 연구 방향

본 논문에서는 실시간 트래픽 분석 방법론을 위한 운영체제 판별 방법론을 제안하였다. 수동형 방법의 문제점인 분석률과 정확성을 향상시키기 위해서 패킷 기반의 분석을 플로우 기반의 분석으로 변경하였으며, SYN/ACK패킷의 추가를 통해서 단말이 서버인 경우도 분석이 가능하도록 하였다. 또한 HTTP트래픽을 이용하여 시그니쳐 추출의 문제점을 해결함으로써 분석률과 정확성을 향상시켰다. 이는 제안한 방법에 대해서 실험을 통한 검증을 하였다.

향후 운영체제 별로 시그니쳐를 재배치시켜 실시간 트래픽 분석 방법론의 성능 향상을 검증할 것이다.

참 고 문 헌

[1] 유재학, 박준상, 이한성, 임영회, 김명섭, 박대희, "다차원 데이터 큐브 모델에서의 네트워크 트래픽 분석", 2010년도 정보과학회 추계학술대회, 단국대, pp.100-105, Nov., 5, 2010.

[2] 오영석, 박준상, 윤성호, 박진완, 이상우, 김명섭, "멀티 레벨 기반의 응용 트래픽 분석 방법", 통신학회논문지 Vol.35 No.8, pp.1170-1178, Aug., 2010.

[3] Chenjie Gu, Shunyi Zhuan, Yanfei Sun, Junrong Yan, "Multi-levels Traffic Classification Technique", ICFCC 2010 Vol1, Wuhan, pp.448-452, May., 21-24, 2010.

[4] Li Jun, Zhang Shunyi, Lu Yanqing, Yan Junrong, "Hybrid Internet Traffic Classification Technique", Journal of Electronics Vol 26 No.1, China, pp.101-112, Jan., 2009.

[5] "OS Fingerprinting", http://en.wikipedia.org/wiki/OS_Fingerprinting

[6] "nmap", http://nmap.org

[7] Greg Taleck, "Ambiguity Resolution via Passive OS Fingerprinting", RAID 2003, pp.192-206, 2003

[8] "p0f", http://lcamtuf.coredump.cx/p0f.shtml

[9] "SynFP", http://www.gomor.org/bin/view/Sinfp/WebHome

[10] http://support.microsoft.com/kb/314053

[11] 박상훈, 박진완, 김명섭, "Flow 기반 실시간 트래픽 수집 및 분석 시스템", 정보처리학회 추계학술대회, 목포대학교, 전주, pp.1061, Nov., 9-10, 2007.

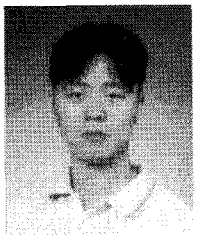
[12] 윤성호, 노현구, 김명섭, "TMA(Traffic Measurement Agent)를 이용한 인터넷 응용 트래픽 분류", 통신학회 하계종합학술발표회, 라마다플라자호텔, pp.618, Jul., 2-4, 2008.

[13] J.Caballero, S. Venkataraman, P.Poosankam, M.G. Kang, D.Song, and A. Blum, "FiG : Automatic Fingerprint Generation", NDSS, February, 2007.

[14] Jun-Sang Park, Sung-Ho Yoon, and Myung-Sup Kim, "Software Architecture for a Lightweight Payload Signature-based Traffic Classification System," Proc. of the Traffic Monitoring and Analysis (TMA) Workshop 2011, LNCS6613, Vienna, Austria, pp.136-149, Apr., 27, 2011.

[15] Young J. Won, Byung-Chul Park, Hong-Taek Ju, Myung-Sup Kim, and James W. Hong, "A Hybrid Approach for Accurate Application Traffic Identification," IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (E2EMON) 2006, Vancouver, Canada, pp.1-8, Apr., 3, 2006.

이 현 신 (Hyun-shin Lee) 준회원

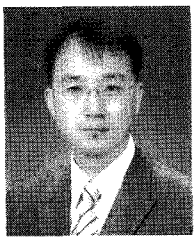


2009년 8월 고려대학교 정보수학과 학사

2009년~현재 고려대학교 컴퓨터정보학과 석사과정

<관심분유> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

김 명 섭 (Myung-sup Kim) 종신회원



1998년 포항공과대학교 전자계산학과 학사

1998년~2000년 포항공과대학교 컴퓨터공학과 석사

2000년~2004년 포항공과대학교 컴퓨터공학과 박사

2004년~2006년 Post-Doc., Dept. of ECE, Univ. of Toronto, Cananda

2006년~현재 고려대학교 컴퓨터정보학과 부교수

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크