순차 패턴 알고리즘을 사용한 트래픽 행위 시그니쳐 생성 방법

윤성호, 박준상, 안현민, 김명섭
고려대학교

{sungho_yoon, junsang_park, queen26, tmskim}@korea.ac.kr

Traffic Behavior Signature Extraction using Sequence Pattern Algorithm

Sung-Ho Yoon, Jun-Sang Park, Hyun-Min An, Myung-Sup Kim
Korea Univ.

요 약

효과적인 네트워크 관리를 위해 응용 트래픽 분석의 중요성이 강조되고 있다. 이를 위해 다양한 분석 방법이 제안되고 있지만, 점점 복잡해지는 응용 트래픽을 분석하기에는 많은 한계점을 가진다. 기존 분석 방법론의 한계점을 보완하기 위해 트래픽 행위 시그니쳐가 제안되었다. 행위 시그니쳐는 응용의 특정 기능을 사용할 때 발생하는 고유한 행동 양식을 의미한다. 본 논문에서는 순차 패턴 알고리즘을 사용하여 행위 시그니쳐을 생성하는 방법을 제안한다. 또한, 제안한 시그니쳐 생성 방법을 실제 응용에 적용한 결과를 제시하여 행위 시그니쳐의 타당성을 보인다.

Ⅰ. 서 론

인터넷 트래픽의 발생 원천을 알아내는 트래픽 분석은 효과적인 네트워크 관리를 위해 반드시 선행되어야 한다. 다양한 분석 방법론이 제안되었지만, 점점 복잡해지는 응용 트래픽으로 인해 실제 네트워크 트래픽에 활용하기에는 많은 한계점을 가진다.

본 연구진은 응용 트래픽의 고유한 행동 양식을 행위 시그니쳐로 생성하는 방법을 제안[1]하였다. 하지만, 기존에 제안된 방법은 모든 후보 시그니쳐를 생성하고 특정 기준 이상의 시그니쳐를 선택하는 구조였기 때문에 입력 데이터가 증가할수록 생성 시간 및 시스템 메모리가 증가는 문제점이 있었다. 이를 개선하기 위하여 본 논문에서는 순차 패턴 알고리즘[2]을 활용하여 행위 시그니쳐를 생성하는 방법을 제안한다. 순차 패턴 알고리즘은 최소 지지도를 만족하지 못하는 부분 패턴을 조기에 제외함으로써 입력 데이터의 크기에 상관없이 시그니쳐를 생성할 수 있다.

본 논문은 2 장에서 5 단계로 진행되는 행위 시그니쳐 생성 방법에 대해 설명하고, 3 장에서는 실험 결과를 기술한다. 마지막으로 4 장에서는 결론과 향후 연구를 언급한다.

Ⅱ. 행위 시그니쳐 생성

본 장에서는 순차 패턴 알고리즘을 사용하여 행위 시그니쳐를 생성하는 방법에 대해 설명한다. 행위 시그니쳐는 응용의 특정 기능을 사용할 때 발생하는 고유한 행동 약식을 의미한다. 즉, 로그인, 채팅, 비디오 시청, 파일 다운로드 등과 같은 기능을 수행 할 때 발생하는 트래픽들의 첫 질의 패킷들을 시그니쳐로 구성한다.

행위 시그니쳐는 수식 1,2 와 같이 정의한다. 행위 시그니쳐 BS는 시간 간격(I)와 단일 호스트에서 발생한 트래픽들의 첫 질의 패킷에서 추출한 엔트리로 구성된다. 엔트리 E 는 $\{ip, port, prot, paybod\}$로 구성된 집합의 $\emptyset$ 집합을 제외한 멱집합(power set)의 집합 원소로 구성된다.

$$BS = \left\{ \begin{matrix} \mathrm{I}, \mathrm{E}_1, \mathrm{E}_2, \ldots, \mathrm{E}_e | \\ e \geq 2, \mathrm{Src}(\mathrm{E}_1) = \mathrm{Src}(\mathrm{E}_2) = \cdots = \mathrm{Src}(\mathrm{E}_e) \end{matrix} \right\} \quad (1)$$

$$E = \{2^C | C = \{\mathrm{ip, port, prot, payload}\}, E \neq \emptyset\} \quad (2)$$

시그니쳐 생성 알고리즘은 엔트리 추출, 트랜잭션 정렬, 순차 패턴 추출, 최대 길이 패턴 추출, 패턴 최적화와 같이 총 5 단계로 수행된다. 본 알고리즘의 입력은 플로우 형태의 트래픽(F)과 최소 지지도(α)이고 출력은 행위 시그니쳐(BS)이다.

엔트리 추출 모듈은 플로우(F)를 입력 받아 플로우 당 하나의 트랜잭션(T)을 구성한다.

$$F = \left\{ \begin{matrix} srcIP, srcPort, L4Prot, dstIP, dstPort, \\ capture\_time, paybad \mid srcIP = local \end{matrix} \right\} \quad (3)$$

$$T = \left\{ \begin{matrix} host\_id, capture\_time, E | \\ E = \{2^C | C = \{ip, port, prot, paybad\}, E \neq \emptyset\} \end{matrix} \right\} \quad (4)$$

플로우 F 는 수식 5 와 같이 출발지 주소 $(srcIP)$, 출발지 포트번호 $(srcPort)$, 4 계층 프로토콜 $(L4Prot)$, 목적지 주소 $(dstIP)$, 목적지 포트번호 $(dstPort)$, 수집시간 $(capture\_time)$, 페이로드 $(paybad)$ 를 구성요소로 가진다. 트랜잭션 T 는 호스트

본 연구는 2013 년 BK21 플러스 사업 및 2012 년 정부(교육과학기술부) 의 재원으로 한국연구재단(2012R1A1A2007483)의 지원을 받아 수행된 결과임.

아이디(*host_id*), 수집 시간(*capture_time*), 엔트리(E)로 구성된다.

트랜잭션 정렬 모듈에서는 트랜잭션을 host_id 를 주키(major key)로, capture_time을 보조키(minor key)로 사용하여 정렬한다.

순차 패턴 추출 모듈은 입력 받은 트랜잭션으로부터 일정 지지도를 만족하는 순차 패턴 즉, 후보 행위 시그니쳐를 추출한다.

$$S = \{host\_id\ ,\ \langle E_1, E_2,\ \ldots, E_e \rangle\} \tag{5}$$

$$Support\ = \frac{Number\ of\ support\ \ hosts}{Total\ \ number\ of\ hosts} \tag{6}$$

본 모듈에서 사용하는 순차 패턴 S는 수식 5 와 같이 정의한다. 즉, 단일 호스트에서 발생한 모든 트랜잭션의 엔트리들을 조합하여 순차 패턴을 만든다. 본 모듈에서 사용하는 지지도(support)는 수식 6 과 같이 전체 호스트 중 해당 패턴을 포함하는 호스트의 비율을 의미한다.

Algorithm 1. 순차 패턴 추출 알고리즘

Input : $T = \{T_1, T_2,\ \ldots, T_f\}$, $minsupp\ = \alpha$
Output : $BS = \{BS_1, BS_2,\ \ldots, BS_b\}$
sequence $(T,\ \alpha)$

1: ***for*** $i = 0$ ***to*** f ***do***
2: ***if*** $(T_i.host\_id\ \neq\ T_{i-1}.host\_id\ )$ ***then***
3: $j = j + 1$
4:: $S_j = S_j \cup T_i.E$
5: $C_1 = C_1 \cup T_i.E$
6: $k = 2$
5: ***while*** $C_{k-1} \neq \emptyset$ ***do***
7: ***foreach*** *candidate* c *in the candidate set* C_{k-1} ***do***
8: ***for*** $i = 0$ ***to*** s ***do***
9: ***if*** $(S_i$ *include* $c)$ ***then***
10: $count\ = count\ + 1$
11: ***if*** $((count\ /s) > \alpha)$ ***then***
12: $L_{k-1} = L_{k-1} \cup c$
13: C_k = *new candidates generated from* L_{k-1} *using* ***candidate_gen***()
14: $K = k + 1$
15: $BS = \cup_k L_k$
16: ***return*** BS

*__Notation__: T: the set of transactions, BS: the set of behavior signatures, S: the set of sequences, L_k: length k sequences, C_k: length k candidate sequences

알고리즘 1 은 순차 패턴 추출 알고리즘을 나타낸다. 순차 패턴 추출 알고리즘은 입력 받은 트랜잭션으로부터 호스트당 하나의 순차 패턴을 추출(Line:2~4)하고 길이 1 인 후보 순차 패턴 C_1을 생성(Line:5)한다. 생성된 후보 순차 패턴을 검사하여 지지도를 측정하고 최소 지지도 α 을 만족하는 패턴에 한해 L_{k-1} 에 포함(Line:7~12)시킨다. 생성된 L_{k-1} 을 이용하여 길이 K 인 후보 순차 패턴 C_k을 생성(Line:13)한다. 이와 같은 과정은 더 이상 후보 순차 패턴 C_{k-1}을 생성하지 못할 때까지 반복(Line:5~14)한다. 최종 추출된 모든 길이의 순차 패턴은 후보 행위 시그니쳐로 생성(Line:15)된다.

최대 길이 패턴 추출 모듈은 입력 받은 행위 시그니쳐로부터 포함성이 있는 부분 시그니쳐를 제거한다.

패턴 최적화 모듈은 입력 받은 엔트리 추출 모듈에서 단순화된 행위 시그니쳐의 I, dstIP, payload 를 실제 트래픽과 비교하여 구체화 시킨다.

Ⅲ. 실험 및 결과

본 장에서는 제안한 알고리즘을 사용하여 행위 시그니쳐를 생성한 결과를 제시한다. 이를 위해 동영상 공유 사이트인 유투브[3]의 특정 기능을 사용할 때 발생하는 트래픽을 수집하였다. 실험 결과의 타당성을 높이기 위해 5 대 이상의 서로 다른 호스트에서 수집하였고, 최소 지지도(α)를 1 로 설정하였다. 즉, 모든 호스트에서 특정 기능을 사용할 때, 반드시 발생하는 행동 양식을 시그니쳐로 생성하였다.

Table 1. 행위 시그니쳐 예

Function	Signature Example
Login*	3804, {173.194.126.0/24, 80, 6, "GET /signin? ... "}, {173.194.0.0/16, 80, 6, "GET /N4061/ ... "}, {173.194.126/16, 80, 6, "GET /v"}
Search Video	2515, {173.194.0.0/16, 80, 6, "GET /results? ... "}, {173.194.126.0/24, 80, 6, "GET /pagead/ ... "}, {173.194.126/16, 80, 6, "GET /csi?v=..."}
Watch Video	1534, {173.194.126.0/24, 80, 6, "GET /watch? v="}, {74.125.10.0/24, 80, 6, "GET /crossdomain ... "}, {74.125.10.0/24, 80, 6, "GET /generate_204..."}, {173.194.38.0/24, 80, 6, "GET /"}, {173.194.0.0/16, 80, 6, "GET /pagead/"}, {74.125.0.0/16, 80, 6, "GET /csi? v= ... "}

표 1 은 본 논문에서 제안하는 알고리즘을 사용하여 생성한 시그니쳐의 예를 보여준다. 예를 들어, 유투브 로그인 행위 시그니쳐*는 3,804ms 이내에 표 1 에 기술된 3 개의 엔트리를 특징으로 가지는 질의 패킷을 포함하는 플로우가 발생하면 해당 플로우를 유투브의 로그인 기능으로 분석한다.

Ⅳ. 결론 및 향후 연구

본 논문에서는 행위 시그니쳐를 효과적으로 생성하기 위해 순차패턴 추출 알고리즘을 사용한 생성 방법을 제안하였다.

향후 연구로는 생성된 시그니쳐를 사용하여 실제 네트워크 트래픽을 분석하는 시스템을 개발할 예정이다.

참 고 문 헌

[1]Y. Sung-Ho, P. Jun-Sang, K. Myung-Sup, L. ChaeTae, and C. JunHyung, "Behavior signature for big data traffic identification," in Big Data and Smart Computing (BIGCOMP), 2014 International Conference on, 2014, pp. 261-266.

[2]R. Agrawal and R. Srikant, "Mining sequential patterns," in Data Engineering, 1995. Proceedings of the Eleventh International Conference on, 1995, pp. 3-14.

[3]YouTube. Available: http://www.youtube.com/