Received: 21 December 2015 | Revised: 7 August 2017 | Accepted: 27 August 2017

DOI: 10.1002/nem.2011

RESEARCH ARTICLE

WILEY

Effective behavior signature extraction method using sequence pattern algorithm for traffic identification

Kyu-Seok Shim | Sung-Ho Yoon | Baraka D. Sija | Jun-Sang Park | Kyunghee Cho | Myung-Sup Kim

Department of Computer and Information Science, Korea University, Sejong, Korea

Correspondence
Kyunghee Cho and Myung-Sup Kim, Department of Computer and Information Science, Korea University, Sejong, Korea.
Email: misskorea69@korea.ac.kr; tmskim@korea.ac.kr

Funding information
Institute for Information and Communications Technology Promotion (IITP), Grant/Award Number: 2017-0-00513

Summary
With the rapid development of the internet and a vigorous emergence of new applications, traffic identification has become a key issue. Although various methods have been proposed, there are still several limitations to achieving fine-grained and application-level identification. Therefore, we previously proposed a behavior signature model for extracting a unique traffic pattern of an application. Although this signature model achieves a good identification performance, it has trouble with the signature extraction, particularly from a huge amount of input traffic, because a *Candidate-Selection method* is used for extracting the signature. To improve this inefficiency in the extraction process, in this paper, we propose a novel behavior signature extraction method using a sequence pattern algorithm. The proposed method can extract a signature regardless of the volume of input traffic because it excludes certain unsatisfactory candidates using a predefined support value during the early stage of the process. We proved experimentally the feasibility of the proposed extraction method for 7 popular applications.

1 | INTRODUCTION

Network management aims at reinforcing well-balanced use of network resources and security. To accomplish these goals, a network operator should set up a proper network policy. To create an efficient network policy, traffic identification should be preemptively achieved because a policy that blocks or adjusts the target traffic is conducted on the basis of the identification results.[1,2] Traffic identification can be defined as the act of ascertaining which application is contributing to a given network traffic data using distinguishable and unique characteristics and then naming the network traffic data according to the corresponding application or service.

In previous work,[3] a new signature model called a behavior signature was proposed. Most internet applications generate multiple traffic flows when a user conducts a specific function such as a log-in, chat, or file transfer. Thus, a behavior signature model was devised on the basis of the idea that the unique patterns of several flows can represent a specific function of a given application. The previous work focused solely on a new signature model and therefore used a very primitive extraction method called the *Candidate-Selection method*, which is used to select candidates exceeding predefined threshold after extracting all possible candidates. This extraction method results in greater computational overhead as the volume of input traffic increases.

For an effective behavior signature extraction method, we propose a novel method using a sequence pattern algorithm.[4,5] The purpose of a sequence pattern algorithm is to find statistically relevant sequence patterns from a huge time-series dataset. This is similar to our behavior signature extraction process for finding unique sequence patterns from several flows representing the specific function of a given application. In this paper, we attempt to improve the sequence pattern algorithm to suit our purposes.

The remainder of this paper is organized as follows. We survey several existing traffic identification methods and a sequence pattern algorithm in Section 2. In Section 3, we describe the concept of a behavior signature in detail. In Section 4, we propose an extraction algorithm for a behavior signature using a sequence pattern algorithm. In Section 5, we propose an identification algorithm using a modified version of the Aho-Corasick algorithm. In Section 6, we discuss the experimental results proving the feasibility of the proposed method. Finally, we conclude and provide some future work in Section 7.

2 | RELATED WORK

In this section, we first categorize identification methods on the basis of the traffic unit used for feature extraction and traffic identification and describe the pros and cons of each method. In addition, we provide the definition, history, and use of a sequence pattern algorithm.

2.1 | Existing traffic identification methods

The most primitive method for identifying traffic is packet-level identification. A packet is the minimum unit of network traffic, and thus, packet-level identification is conducted by checking the existence of a rule defined as an invariant pattern of the target application for each packet traveling across the network. Typical examples of this method include port- and payload-based methods.

A port-based method checks the port number of each packet and identifies the application according to the Internet Assigned Numbers Authority (IANA) list of well-known and registered ports.[6,7] In the early days of the internet, this method performed well; currently, however, its accuracy has become less reliable because of violations to port assignments. A payload-based method conducts a deep packet inspection to find the substring representing the target application from each packet.[8-11] The results are robust in terms of completeness and accuracy, and this type of method is therefore widely used by the industry. However, various limitations, including payload encryption, datasets without a payload for dealing with privacy issues, computational overhead, and a lack of publicly available documentations for a proprietary protocol, restrict its use.

A flow is a set of packets that contain the same 5-tuple packet header information, such as the source and destination IP address, source and destination port number, and transport layer protocol, which contains all of the forward and backward packets in a session established between 2 hosts. Flow-level identification uses statistical flow information, such as the number of packets and bytes in the flow, as well as the flow bitrate, flow size, flow duration, and interarrival time between packets. Flow-based methods can be categorized into 2 types: learning- and distribution-based methods.

A learning-based method uses an existing machine learning technique that can again be divided into supervised[12,13] and unsupervised[14,15] learning. In supervised learning, a statistical model based on a machine learning algorithm is constructed using the ground-truth traffic (training data) labeled on the basis of the nature of the traffic, and the target traffic (test data) is classified. This method guarantees highly accurate results in trained applications. However, the results of this method depend on the quality of the ground-truth traffic, and unknown applications excluded from the training data cannot be classified. Unsupervised learning classifies traffic through clustering techniques using similar statistical information from traffic generated by the same application protocol, does not require a training phase, and can classify unknown applications. However, the results of this method are affected by tunable parameters, and an additional process is required to label the application name of each cluster.

2.2 | Sequence pattern algorithm

A sequence pattern algorithm was first introduced in Agrawal and Srikant.[5] In this algorithm, which is a type of data mining technique, time-series patterns are detected from a database having sequences of values or events. This technique looks very similar to association rule mining[16] because both are similar processes for discovering frequent patterns in large datasets; however, association rule mining aims to extract concurrent patterns from the same transaction, and a sequence pattern algorithm aims to extract patterns with a certain order from different transactions.[17]

The support value is very important for this algorithm. Support value is the frequency rate of a target event which compare with minimum support value to determine the event is meaningful or not. A minimum support predefined by the conductor is used because the number of possible subsequences is very large and conductors have different interests and purposes. Thus, we have to prune out uninteresting patterns using the minimum support during early stage. We determined that the signature is most meaningful if it can commonly occur in all cases and analyze the target application only. Therefore, we extracted the signatures with support value set to 1. If minimum support is set to less than 1, recall can be increased with more signatures generated, while the precision can be decreased, a part that we leave as one of our future work. In this paper, the algorithm is performed after setting the minimum support value as 1. The object of this paper generates highly accurate behavior signatures that can classify a target application traffic.

Several methods have been proposed during recent years. In this section, we describe some typical examples of such methods. AprioriAll is based on the Apriori property, which means that any subsequence of a frequently occurring sequence is frequent.[5] This method generates candidate sequences and checks the support of each candidate to determine frequently occurring sequences. PrefixSpan examines the prefix subsequences and projects their corresponding postfix subsequences into projected databases.[18] The philosophy of a sequence pattern algorithm is detecting time-series patterns from a huge dataset. This is parallel to purpose of behavior signature in terms of finding unique sequence patterns that can represent the specific function of a given application. We therefore improved a sequence pattern algorithm to fit the signature extraction.

3 | BEHAVIOR SIGNATURE

Advanced multitasking technology, ie, the sharing of common processing resources, has become possible on various applications through the rapid growth of the internet and other computer technologies. Using multiple applications, internet users may intentionally or unintentionally generate various types of concurrent application traffic simultaneously.

Because several applications generate traffic simultaneously, traffic identification by means of a flow or packet unit causes a significant amount of overhead in the identification system and cannot guarantee a high accuracy of the identification result. Most applications generate multiple flows when conducting a particular function, and these flows have clear patterns, such as their sequence and interval, from the perspective of the application behaviors. Also, most modern applications generate the multiple flows.[19] We can therefore identify these concurrent flows as the application level at once using their significant patterns. The proposed behavior signature–based identification method extracts a signature combining traffic features and a unique pattern from the target application and identifies their concurrent traffic (flow) simultaneously.

Figure 1 shows the various traffic units for traffic identification, ie, the packet, flow, and interflow units. The packet unit uses the packet header information (IP address, port number, and layer-4 protocol) and payload in a single packet. Using the identification results, the packet unit is a suitable means for real-time control owing to its ability to control unwanted traffic immediately after a packet inspection. However, it is difficult to extract a signature because the range of extraction is relatively small. The flow unit, on the other hand, uses not only the packet attributes but also additional information, such as the interarrival time, packet size distribution, and total byte size. The flow unit is suitable for extracting signatures because of its relatively large range of extraction. Moreover, it can be applied to encrypted traffic

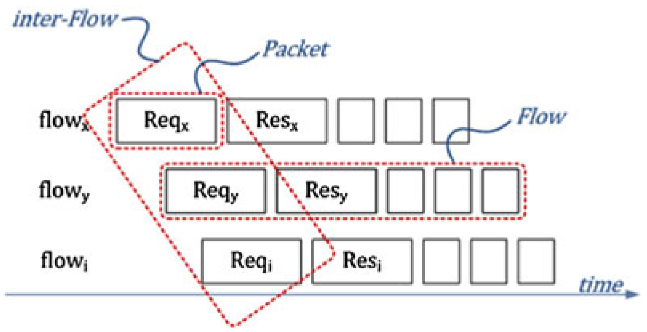


FIGURE 1 Various traffic units for traffic identification

using only statistical features. However, limitations such as a low accuracy and real-time control are incurred. The flow unit can be used to identify traffic after a flow has ended.

The behavior signature is applied using an interflow unit, as shown in Figure 1, and seeks to minimize the limitations of both the packet and flow units and maximize their advantages. An interflow is a set of first request packets of several flows. Therefore, it uses not only packet unit attributes but also unique patterns such as the sequence and interval information. It is easy to create a signature because the range of extraction is large owing to the use of plural flows. In other words, a signature is extracted using various traffic features. In addition, an interflow has the ability to control traffic in real time because the traffic is identified by the signature in the first request packet located at the beginning of the flow. A behavior signature consists of several entries with the aforementioned traffic attributes. The following equations can be used to define a behavior signature.

$$BS = \left\{ \begin{array}{c} A, I, E_1, E_2, \dots, E_n | \\ n \geq 1, Src(E_1) = Src(E_2) = \dots = Src(E_n) \end{array} \right\} \tag{1}$$

$$E = \left\{ 2^F | F = \{ip, port, prot, payload, offset\}, E \neq \varnothing \right\} \tag{2}$$

A behavior signature (*BS*) consists of the application name (*A*), interval (*I*), and entries (E_n), where $n \geq 1$. The entry (*E*) is a power set of the destination IP address (*ip*), destination port (*port*), layer-4 protocol (*prot*), and *N*-byte payload with an offset (*payload*), where a null set is excluded. In other words, we selectively use the features of entry in the traffic identification if the selected feature is meaningful. Figure 2 shows an example of a behavior signature. For example, we extract representative signatures from a Youtube user with clear information, such as log-in and video functions. We can prove that several features in the first request packet create a significant pattern representing certain function of a service.

4 | EXTRACTION ALGORITHM

In this section, we describe an extraction algorithm used for a behavior signature using a sequence pattern algorithm. As previously stated, we improved our method using the *AprioriAll* algorithm.[5] We apply AproriAll algorithm for quick search and accurate extraction of behavior signatures. The AprioriAll algorithm is a technique to extract frequent subsequences in a large number of sequences. Table 1 shows list of transaction set generated by 3 hosts. The ultimate purpose of this algorithm is extracting a sequence pattern satisfying certain support value from several hosts accessing a single application.

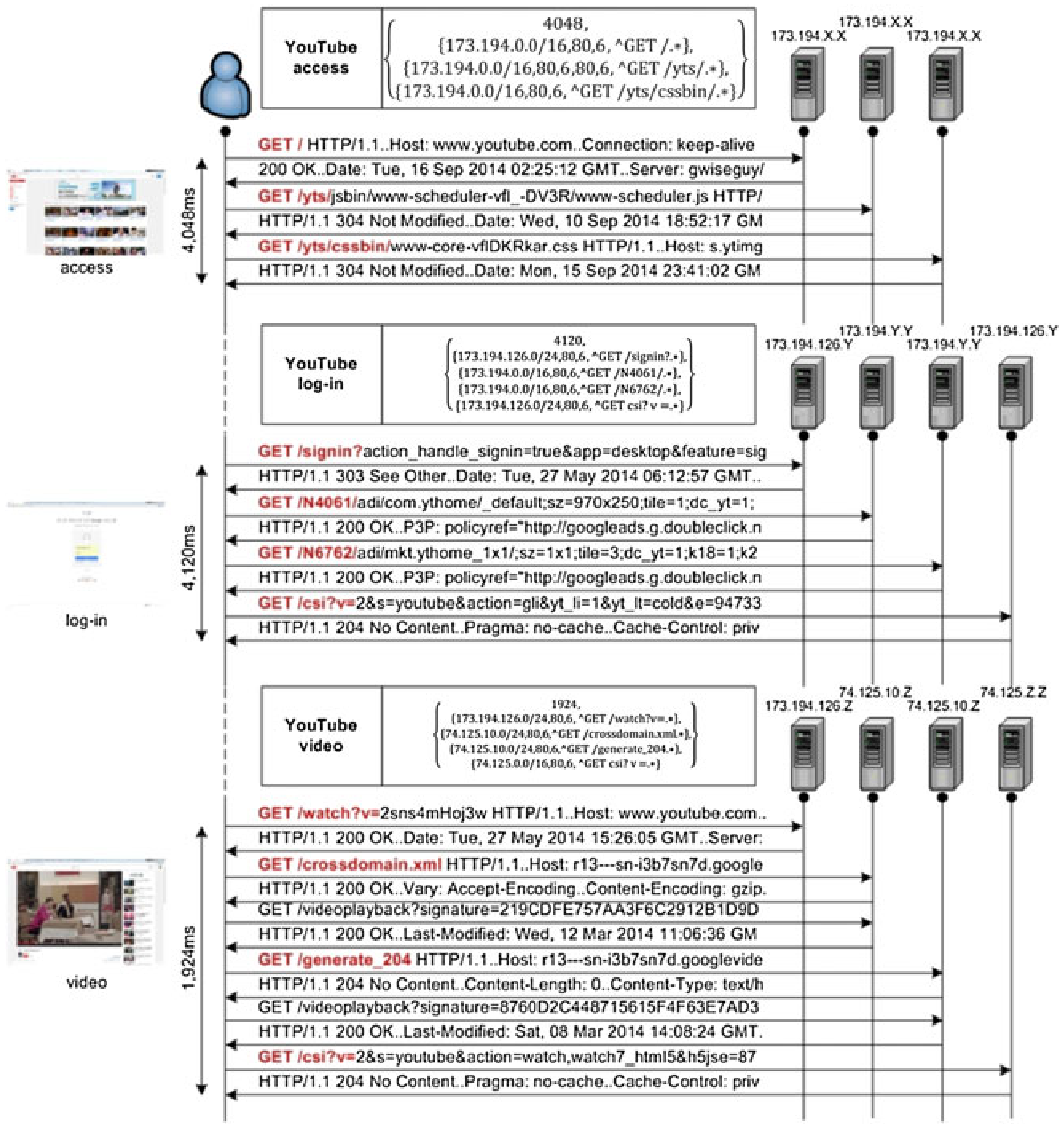
Table 2 shows an example of a behavior signature from Table 1 with minimum support of 60% (minimum of 2 hosts). The example consists of a certain interval and 3 entries. Thus, this signature identifies traffic matching with these 3 entries within a given interval of 13 000 milliseconds. To extract behavior signature, we devise 3 modules. The following subsections describe these submodules.

4.1 | Sequence extraction module

The sequence extraction module extracts sequence set from the input transaction set consisting of 2 more transactions. Equations 3 and 4 show definition of the transaction. Each transaction means set of flows generated from a single host. The flow consists of source IP address, source port, layer-4 protocol, destination IP address, destination port, payload data, start time, forward data packet set, and backward data packet set as shown in Equation 5. As output of this module, Equations 6 and 7 show definition of the sequence set and the sequence, respectively. Each sequence consists of a sequence id and set of items. The sequence id means a transaction id. Thus, one sequence is extracted from one transaction. The definition of item is similar to definition of an entry as described in Section 3. The item is converted to entry in the sequence module.

$$T = \{T_1, T_2, \dots, T_t | t \geq 2\} \tag{3}$$

$$T_i = \{id, \{F_1, F_2, \dots, F_x\}\} \tag{4}$$



```
4048,
{173.194.0.0/16,80,6, ^GET /.*},
{173.194.0.0/16,80,6,80,6, ^GET /yts/.*},
{173.194.0.0/16,80,6, ^GET /yts/cssbin/.*}

GET / HTTP/1.1..Host: www.youtube.com..Connection: keep-alive
200 OK..Date: Tue, 16 Sep 2014 02:25:12 GMT..Server: gwiseguy/
GET /yts/jsbin/www-scheduler-vfl_-DV3R/www-scheduler.js HTTP/
HTTP/1.1 304 Not Modified..Date: Wed, 10 Sep 2014 18:52:17 GM
GET /yts/cssbin/www-core-vflDKRkar.css HTTP/1.1..Host: s.ytimg
HTTP/1.1 304 Not Modified..Date: Mon, 15 Sep 2014 23:41:02 GM
```

```
4120,
{173.194.126.0/24,80,6, ^GET /signin?.*},
{173.194.0.0/16,80,6,^GET /N4061/.*},
{173.194.0.0/16,80,6,^GET /N6762/.*},
{173.194.126.0/24,80,6, ^GET csi? v =.*}

GET /signin?action_handle_signin=true&app=desktop&feature=sig
HTTP/1.1 303 See Other..Date: Tue, 27 May 2014 06:12:57 GMT..
GET /N4061/adi/com.ythome/_default;sz=970x250;tile=1;dc_yt=1;
HTTP/1.1 200 OK..P3P: policyref="http://googleads.g.doubleclick.n
GET /N6762/adi/mkt.ythome_1x1/;sz=1x1;tile=3;dc_yt=1;k18=1;k2
HTTP/1.1 200 OK..P3P: policyref="http://googleads.g.doubleclick.n
GET /csi?v=2&s=youtube&action=gli&yt_li=1&yt_lt=cold&e=94733
HTTP/1.1 204 No Content..Pragma: no-cache..Cache-Control: priv
```

```
1924,
{173.194.126.0/24,80,6, ^GET /watch?v=.*},
{74.125.10.0/24,80,6,^GET /crossdomain.xml.*},
{74.125.10.0/24,80,6,^GET /generate_204.*},
{74.125.0.0/16,80,6, ^GET csi? v =.*}

GET /watch?v=2sns4mHoj3w HTTP/1.1..Host: www.youtube.com..
HTTP/1.1 200 OK..Date: Tue, 27 May 2014 15:26:05 GMT..Server:
GET /crossdomain.xml HTTP/1.1..Host: r13---sn-i3b7sn7d.google
HTTP/1.1 200 OK..Vary: Accept-Encoding..Content-Encoding: gzip.
GET /videoplayback?signature=219CDFE757AA3F6C2912B1D9D
HTTP/1.1 200 OK..Last-Modified: Wed, 12 Mar 2014 11:06:36 GM
GET /generate_204 HTTP/1.1..Host: r13---sn-i3b7sn7d.googlevide
HTTP/1.1 204 No Content..Content-Length: 0..Content-Type: text/h
GET /videoplayback?signature=8760D2C448715615F4F63E7AD3
HTTP/1.1 200 OK..Last-Modified: Sat, 08 Mar 2014 14:08:24 GMT.
GET /csi?v=2&s=youtube&action=watch,watch7_html5&h5jse=87
HTTP/1.1 204 No Content..Pragma: no-cache..Cache-Control: priv
```

FIGURE 2 Example of header signature of Youtube

TABLE 1 Transaction set

ID	StartTime	srcIP	srcPort	L4Prot	dstIP	dstPort	payload
1	12:30′ 11″	xxx.xxx.xxx.1	51912	6	173.194.126.130	80	pear
2	12:30′ 13″	xxx.xxx.xxx.2	35691	6	173.194.126.131	80	peach
	12:30′ 18″	xxx.xxx.xxx.2	35692	6	74.125.23.120	80	kiwi
	12:30′ 21″	xxx.xxx.xxx.2	35694	17	173.194.38.76	80	kiwi
	12:30′ 26″	xxx.xxx.xxx.2	35693	6	173.194.72.95	80	lemon
3	12:30′ 12″	xxx.xxx.xxx.3	21511	6	173.194.126.133	80	pea
	12:30′ 17″	xxx.xxx.xxx.3	21513	6	74.125.23.120	80	kiwiju
	12:30′ 22″	xxx.xxx.xxx.3	21519	6	173.194.72.95	80	leek
	12:30′ 25″	xxx.xxx.xxx.3	21520	6	173.194.126.137	80	peach

TABLE 2 Example of behavior signature

Behavior Signature (Support > 60%)
$\left\langle \begin{array}{c} 13000, \{173.194.126.0/24,\ 80,\ 6, \hat{}pea.^{*}, 0\}, \\ \{74.125.23.120/32,\ 80,\ 6, \hat{}kiwi.^{*}, 0\}, \\ \{173.194.72.95/32,\ 80,\ 6, \hat{}le.^{*}, 0\} \end{array} \right\rangle$

$$F_i = \left\{ \begin{array}{c} srcIP, srcPort, L4Prot, dstIP, dstPort, paylaod, \\ startTime, \\ forward = \left\{P_1, P_2, ..., P_x | P_{1.5tupe} ... = P_{x.5tupe}\right\}, \\ backward = \left\{P_1, P_2, ..., P_y | P_{1.5tupe} ... = P_{y.5tupe}\right\} \\ |startTime = forward.p_1.startTime, \\ srcIP = forward.srcIP = backward.dstIP, \\ srcPort = forward.srcPort = backward.dstPort, \\ L4Prot = forward.L4Prot = backward.L4Prot, \\ dstIP = forward.dstIP = backward.srctIP, \\ dstPort = forward.dstPort = backward.srcPort \end{array} \right\} \tag{5}$$

$$S = \{S_1, S_2, ..., S_t\} \tag{6}$$

$$S_i = \{id, \langle I_1, I_2, ..., I_x \rangle\} \tag{7}$$

$$I_i = \left\{ \begin{array}{c} startTime, 2^C \\ |C = \{ip, port, prot, payload\}, 2^C \neq \emptyset \end{array} \right\} \tag{8}$$

Algorithm 1 Sequence extraction algorithm

```
Input: $T = \{T_1, T_2, ..., T_t\}$
Output: $S = \{S_1, S_2, ..., S_t\}$
Function extract(F)
1: for $i=1$ to $t$ do
2:     $S_i.\ id = T_i.\ id$
3:     foreach flow $f_j$ in $T_i$ do
4:         $S_i.\ I_j.\ startTime = f_j.\ startTime$
5:         $S_i.\ I_j.\ ip =$ first 24 bits of $f_j.\ dstIP$
6:         $S_i.\ I_j.\ port = f_j.\ dstPort$
7:         $S_i.\ I_j.\ prot = f_j.\ L4Prot$
8:         $N = getMaxLength(f_j.\ payload)$
9:         $S_i.\ I_j.\ payload =$ first $N$ bytes of $f_j.\ payload$
10: return S
```

Algorithm 1 shows the sequence extraction algorithm. In the case of dstIP, we consider the first 24 bits (C class) to extract the same values in same server farm (line:5). In the case of payload, we consider the first *N* bytes (line:8~9). Here, *N* is determined by using string tree as shown in Figure 3. The string tree is created using all flow payloads. A single node means a 1-byte character of payload, and each node has a support value. The support means the ratio of hosts having a character string from root node to target node. For example, node "c" at a depth of 4 has 0.67 (2/3) as a support because 2 hosts have the character string "peac" (ID 2 and 3). Excluding a variable length string such as a user ID and time value, we can extract *F.payload* using the maximum substring within nondecreasing support. Thus, 3 payloads of Table 1 such as "pear," "pea," and "peach" are extracted as "pea." Table 3 shows the actual sequence output for Figure 3. The sequence includes not only payload but also time, IP, port, and protocol information. The system extracts the pattern using the following information.

4.2 | Sequence pattern extraction module

The sequence pattern extraction module extracts sequence pattern set with a certain support from the input sequences. This pattern is considered a candidate behavior signature. To calculate support, we use the Equation 9. The support used

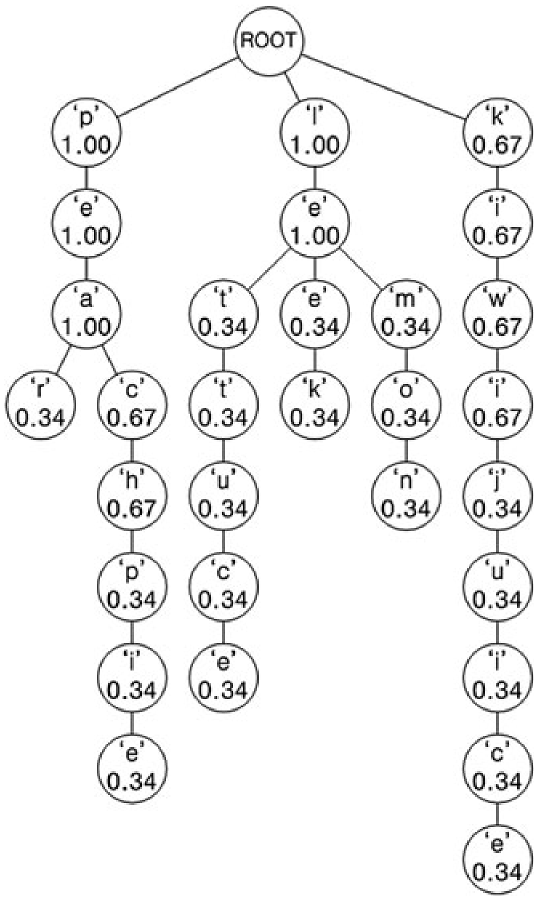


FIGURE 3 Example of string tree

TABLE 3 Result of sequence extraction module

S_i	$S_i.\ I_{id}$	$S_i.\ I_j.$ *startTime*	$S_i.\ I_j.$ *ip*	$S_i.\ I_j.$ *port*	$S_i.\ I_j.$ *prot*	$S_i.\ I_j.$ *payload*
S_1	1	12:30′ 11″	173.194.126.0	80	6	pea
S_2	1	12:30′ 13″	173.194.126.0	80	6	pea
	2	12:30′ 18″	74.125.23.0	80	6	kiwi
	3	12:30′ 21″	173.194.38.0	80	17	kiwi
	4	12:30′ 26″	173.194.72.0	80	6	le
S_3	1	12:30′ 12″	173.194.126.0	80	6	pea
	2	12:30′ 17″	74.125.23.0	80	6	kiwi
	3	12:30′ 22″	173.194.72.0	80	6	le
	4	12:30′ 25″	173.194.126.0	80	6	pea

in this module is the ratio of supporting sequences to total sequences.

$$\text{Support} = \frac{\text{Number of support transactions (sequences)}}{\text{Total number of transactions (sequences)}} \tag{9}$$

Algorithm 2 shows the sequence pattern extraction algorithm. This algorithm extracts sequence patterns form the sequence set. First, this algorithm extracts length 1 candidate pattern C_1 (line:1~3). Exploring all candidate patterns C_{k-1}, we measure the support and move it to L_{k-1} if the pattern has support values exceeding the minimum support (α) (line:6~11). Next, we create length K candidate pattern C_k using L_{k-1} in the candidate pattern extraction algorithm (line:12). This iteration is repeated until no more candidates are created (line:5~13). Finally, all extracted patterns in L_k become candidate behavior signatures (line:14).

Algorithm 2 Sequence pattern extraction algorithm

```
Input: $S = \{S_1, S_2, ..., S_t\}, minsupp = \alpha$
Output: $BS_{temp} = \{BS_1, BS_2, ..., BS_b\}$
Function sequence($S,\alpha$)
1: for $i=1$ to $t$ do
2:     foreach item $I$ in $S_i$ do
3:         $C_1 = C_1 \cup I$
4: $k = 1$
5: while $C_k \neq \varnothing$ do
6:     $\mathbf{L_k} \neq \varnothing$
7:     foreach candidate $c$ in the candidate set $C_k$ do
8:         count $= 0$
9:         for $i = 0$ to $t$ do
10:            if($S_i$ include $c$) then $count = count + 1$
11:        if($(count/t) \geq \alpha$) then $L_k = L_k \cup c$
12:    $C_{k+1}$ = new candidates generated from $L_k$ using candidate_gen()
13:    $k = k + 1$
14: $BS_{temp} = \cup_k L_k$
15: return $BS_{temp}$
```

Algorithm 3 shows the candidate pattern extraction algorithm. This algorithm inputs length $k - 1$ sequence patterns L_{k-1} and outputs length K candidate patterns C_k. The length K candidate pattern is created using length $K - 1$ patterns p and q, if the subpattern of p excluding the first entry and the subpattern of q excluding the last entry are the same (line:7).

Table 4 shows the results of the sequence pattern extraction module. As an intuitive explanation of this, we transform the items of the sequence into symbols. After extracting length 1 pattern C_1, we measure the support of each pattern using Equation 9. We move C_1 to L_1 only when exceeding minimum support α (which is set at 60%). The support value is very important in extracting signatures in this system. The minimum support value is used as a baseline threshold to extract signatures from candidates. The minimum support value with 60% means that candidates with over 60% support will be extracted as signatures. Candidates with 100% support value indicate that they appear in all the sequences. The lower the minimum support is, the more candidates are extracted as signatures, which result in increased coverage but decreased accuracy. On the other hand, the higher the minimum support is, the less candidates are extracted as the signatures. In that case, the accuracy can increase, but the coverage will decrease. We then make length 2 patterns C_2 using L_1. This iteration is repeated until no more candidates are created. Finally, we consider all candidate patterns exceeding minimum support α as candidate behavior signatures.

Algorithm 3 Candidate pattern extraction algorithm

```
Input : $L_{k-1}$
Output : $C_k$
candidate_gen($L_{k-1}$)
1: foreach sequence $p$ in $L_{k-1}$ do
2:     foreach sequence $q$ in $L_{k-1}$ do
3:         if($p.\ E_2 = q.\ E_1$) then
4:             if($p.\ E_3 = q.\ E_2$) then...
5:                 if($p.\ E_{k-1} = q.\ E_{k-2}$) then
6:                     $C_k = C_k \cup \langle p.\ E_1, p.\ E_2, ..., p.\ E_{k-1}, q.\ E_{k-1} \rangle$
7: return $C_k$
```

TABLE 4 Result of sequence pattern extraction module

S_i	S_i Transformed as Symbol
$\{1, \{12:30'11'', \{173.194.126.0, 80, 60, pea\}\}\}$	A
$\left\{2, \begin{matrix}\{12{:}30'13''\{173.194.126.0,\ 80,\ 6,\ pea\}\},\\ \{12{:}30'18''\{74.125.23.0,\ 80,\ 6,\ kiwi\}\},\\ \{12{:}30'21''\{173.194.38.0,\ 80,\ 17,\ kiwi\}\},\\ \{12{:}30'26''\{173.194.72.0,\ 80,\ 6,\ le\}\}\end{matrix}\right\}$	$A\,B\,C\,D$
$\left\{3, \begin{matrix}\{12{:}30'12''\{173.194.126.0,\ 80,\ 6,\ pea\}\},\\ \{12{:}30'17''\{74.125.23.0,\ 80,\ 6,\ kiwi\}\},\\ \{12{:}30'22''\{173.194.72.0,\ 80,\ 6,\ le\}\},\\ \{12{:}30'25''\{173.194.126.0,\ 80,\ 6,\ pea\}\}\end{matrix}\right\}$	$A\,B\,D\,A$
C_1	
1-Sequences	**Support, count/s**
$\langle A\rangle$	1.00
$\langle B\rangle$	0.67
~~$\langle C\rangle$~~	~~0.34~~
$\langle D\rangle$	0.67
MinSupp = 60%	
C_2	
2-Sequences	**Support, count/s**
~~$\langle A\,A\rangle$~~	~~0.00~~
$\langle A\,B\rangle$	0.67
$\langle A\,D\rangle$	0.67
~~$\langle B\,A\rangle$~~	~~0.34~~
~~$\langle B\,B\rangle$~~	~~0.00~~
$\langle B\,D\rangle$	0.67
~~$\langle D\,A\rangle$~~	~~0.34~~
~~$\langle D\,B\rangle$~~	~~0.00~~
~~$\langle D\,D\rangle$~~	~~0.00~~
MinSupp = 60%	
C_3	
3-Sequences	**Support, count/s**
$\langle A\,B\,D\rangle$	0.67
MinSupp = 60%	

4.3 | Optimization pattern module

The optimization pattern module eliminates the redundancy of the input behavior signatures and maximizes the interval (*I*), destination IP address (dstIP), and payload (*payload*) of the behavior signature through a comparison with the original input traffic. This checks their inclusion relation among extracted behavior signatures and deletes unnecessary signatures. It also optimizes the extracted behavior signature and extracts effective behavior signature that reduces false-positive rate with high coverage.

Algorithm 4 shows the optimization pattern algorithm. This algorithm takes traffic in the form of a flow and the behavior signatures and creates optimized signatures. First, exploring all behavior signatures, it eliminates subsignatures that are included in other signatures (line:1~4). Exploring all signatures, it then assigns intervals based on the capture time of the input traffic (line:7). If the amount of identified traffic of the candidate signature and the amount of identified traffic of the optimized signature are the same, we convert the signature to optimized signature (line:8~12). Optimizing the interval here refers to measuring the intervals among flow sets, where each set is identified by the same *BS* and selects the maximum value as the interval of behavior signature. In case of the dstIP, we maximize to 32 bits of length of dstIP if the amount of identified traffic using it and those of 24 bits of dstIP is the same. In case of the payload, we increase *N* (payload length) unless the amount decreases.

Algorithm 4 Optimization pattern algorithm

```
Input : $F = \{F_1, F_2, ..., F_f\}, BS_{temp} = \{BS_1, BS_2, ..., BS_b\}$
Output : $BS = \{BS_1, BS_2, ..., BS_c\}, (c \le b)$
optimize($F$, $BS_{temp}$)
1: for $i = 0$ to $b$ do
2:     for $j = 0$ to $b$ do
3:         if($BS_i$ include $BS_j$) then
4:             $BS = BS - BS_j$
5: $c = size(BS)$
6: for $i = 0$ to $c$ do
7:     $BS_i.\ \mathrm{I} =$ find $BS_i.\ I_{max}$ using $F$
8:     $BS_{i \cdot \mathrm{optimize}} = BS_i$
9: $BS_{i \cdot optimize}.\ dstIP =$ exchange_32($BS_i.\ dstIP$)
10:    $BS_{i \cdot optimize}.\ N(payload\ length) =$ increase($BS_i.\ N(payload\ length)$)
11:    if($F(BS_i) == F(BS_{i \cdot optimize})$) then
12:        $BS_i = BS_{i \cdot optimize}$
13: return $BS$
```

5 | IDENTIFICATION ALGORITHM

In this section, we describe the identification method using a behavior signature. Figure 4 shows the network environment when adopting a behavior signature to identify the traffic. We capture the traffic using a network tap or switch located in the middle of an internet link to enable all internet traffic to be captured between the internet and the target network. The captured traffic in the form of a packet input into the flow generator (FG) in real time. The FG creates a flow using input packets having same 5-tuple (source IP address, source port, layer-4 protocol, destination IP address, and destination port). The identifier based on an identification algorithm takes 2 types of traffic from the FG. A first request packet (FRP) is transmitted because the behavior signature only uses it to identify traffic. The FRP indicates

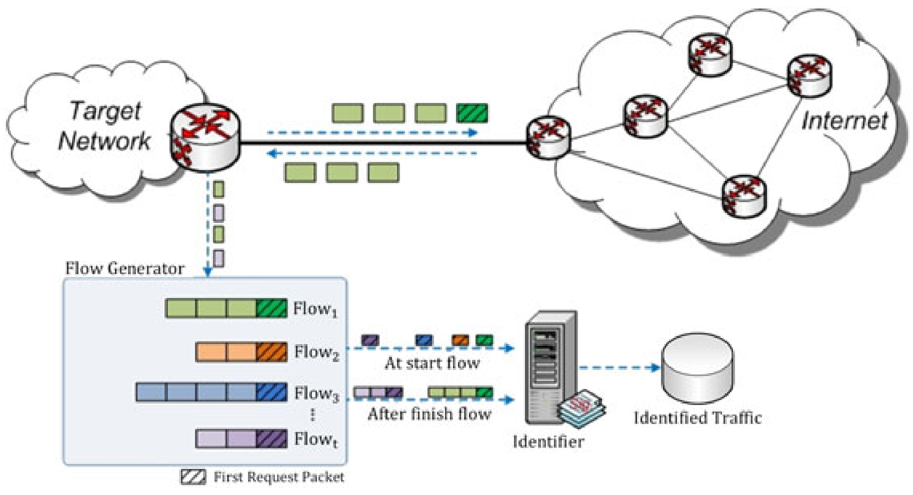


FIGURE 4 Traffic identification network environment using behavior signature

the first request packet with a payload in the flow. When the FG process completes, the FG transmits the flows to the identifier to measure the completeness (volume) of the identified traffic.

The behavior signature identifies the FRP in a real-time network environment, as shown in Figure 4. Thus, each time the FRP of a flow is entered into an identifier, the identifier carries out the identification algorithm. For this reason, we improved on the *Aho-Corasick algorithm*[20] to adopt traffic identification. The purpose of this algorithm is detecting predefined words in a string. Similarly, the proposed algorithm detects an FRP pattern using a behavior signature. The difference between the 2 algorithms is that Aho-Corasick detects only continuative characters, whereas our algorithm can detect noncontinuous patterns on several hosts.

Algorithm 5 Traffic identification algorithm

```
Input: $F = \{F_1, F_2, ..., F_f\}, BS = \{BS_1, BS_2, ..., BS_b\}$
Output: $F' = \{F_1, F_2, ..., F_f\}$
identifyTraffic(F, BS)
1: $T_{transition}, T_{output}, T_{timeout} \leftarrow constructTable(BS)$
2: while true do
3:     $FRP \leftarrow$ wait$(getFirstRequestPacket(\ ))$
4:     $T_{status} \leftarrow insertStatus(FPR, T_{status}, T_{transition})$
5:       $T_{status} \leftarrow updateStatus(FRP, T_{status}, T_{transition})$
6:       $F' \leftarrow outputStatus(FRP, T_{status}, T_{output})$
7: return $F'$
```

The traffic identification algorithm consists of 1 preparation phase and 4 identification phases as shown in Algorithm 5. When the algorithm starts, constructTable module creates a Transition Table, Output Table, and Timeout Table using behavior signature, as the preparation phase (line:1). The created tables are used for maintaining the Status Table during the identification phase. The identification phase consists of 3 detailed modules, where each every FRP is entered, and these 3 modules are conducted repetitively (line:2~6). When an FRP is entered (line:3), referring to Transition Table, insertStatus module inserts the Status Record to Status Table (line:4), and updateStatus module updates Status Record of Status Table (line:5). Finally, outputStatus module determines a list of identified traffic if a certain Status Record of Status Table reaches the output_state by referring to Output Table (line:6). Thus, the input FRP history of Status Record used to transfer their state is identified as traffic by the signature.

6 | EXPERIMENT AND RESULTS

This section details the experimental results and feasibility of the proposed method on the basis of 7 popular applications. We selected the 7 applications shown in Table 5. To collect pure traffic (ground-truth) of the target application, we deploy a Traffic Measurement Agent on the selected hosts generating the target application traffic.[21] This agent continuously monitors the network interface information and transfers the log-data to the designated server. The log-data

TABLE 5 List of selected applications

		For Extraction		For Identification	
Application Type	**Name**	**Flow**	**Bytes**	**Flow**	**Bytes**
P2P (file sharing)	uTorrent	14 430	19 398 M	5819	366 048 K
Web storage	DropBox	1110	186 989 K	38	62 357 K
Social network service	FaceBook	10 860	1914 M	130	45 221 K
Messenger	Line	1980	104 393 K	36	4329 K
	Skype	26 850	143 599 K	43	5845 K
Video streaming	TVPot	21 690	9190 M	480	258 242 K
Video sharing	Youtube	17 010	18 178 M	216	137 508 K

includes not only socket data but also process information responsible for the opening and closing of the socket. Thus, we can obtain the absolute ground-truth traffic of the target application.

6.1 | Signature extraction

To maximize the accuracy of a signature, we set the minimum support (α) as 100%. This means that common patterns observed in all hosts are extracted as a signature. On the other hand, we actually plan to study a method that can keep precision while decreasing the support value. We extract length-1 behavior signatures containing only one entry. Although these signatures cannot be applied to the identification of a specific application function, they are sufficient for identifying the target application because they are commonly observed in the traffic of all hosts using the target application. Table 6 shows some examples of a behavior signature using the proposed method.

6.2 | Performance of extraction algorithm

We measured the accuracy (precision and recall) of the proposed signature method using a mixture of traffic from the 7 applications considered. The following equations are used to measure the precision and recall.

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{10}$$

$$\text{Recall} = \frac{TP}{(TP + FN)} \tag{11}$$

The precision is the ratio of accurately identified traffic to the total amount of identified traffic, and the recall is the ratio of accurately identified traffic to the amount of application traffic.

Table 7 shows the accuracy (precision and recall) of a behavior signature for the 7 applications. All signatures identify the traffic precisely, ie, with a precision of 1.00, for all applications, which is achieved because the signatures are extracted from several different hosts. This test result proves that the behavior signature is very accurate. However, the test results in terms of recall vary according to the type of application. For example, Skype has a low recall rate because this application is operated using peer-to-peer (P2P) networking technology and most of its traffic is encrypted. The traffic features of Skype used by the proposed method are the header information and payload data for entry extraction. Therefore, in our method, some applications using both peer-to-peer networking and encryption may remain in a blind area. However, if various features such as the statistical characteristics are used in extracting an entry, the recall performance will increase. We have left this issue for future work.

Figure 5 compares the test results between the *Candidate-Selection* (CS) and *Sequence Pattern* (SP) in terms of memory use and execution time. The CS method proposed in a previous work determines a signature after listing all possible

TABLE 6 Example of behavior signature

Application	Examples
uTorrent	<0,{any,any,17,""d1:ad2:id20:\"",0}> <21406,{any,any,6,"GET /utorrent-onbording" ,0}, {any,any,6,"POST /e?",0}, {any,any,6,"GET /images/" ,0}>
DropBox	<3825,{108.160.0.0/16,80,6, "GET /subscribe?host_int=",0},{108.160.0.0/16,443,6, "0x41 0x01 0x00 0x00 0x3D 0x03 0x01 0x54",4}>
FaceBook	<7896,{23.65.188.0/24,443,6,"0xE0 0x01 0x00 0x00 0xDC 0x03 0x03",4}, {31.13.0.0/16,443,6,"0xDA 0x01 0x00 0x00 0xD6 0x03 0x03",4}>
Line	<6555,{23.76.153.88/32,80,6, "GET /client/win",0}, {119.235.235.84/32,443,6, "0xE8 0x01 0x00 0x00 0xE4 0x03 0x03",4}, {125.209.222.0/24,443,6, "0xE8 0x01 0x00 0x00 0xE4 0x03 0x03",4}>
Skype	<4542,{any,any,6, "0x82 0x01 0x00 0x00 0x7E 0x03 0x01 0x54 0x48",4}, { any,any,6, "0x7A 0x01 0x00 0x00 0x76 0x03 0x01 0x54 0x48",4}, { any,any,6, " 0x77 0x01 0x00 0x00 0x73 0x03 0x01 0x54 0x48",4}>
TVPot	<5445,{any,any,6, "GET /api/open/live",0},{any,any,6, "SET PRIVATE TRUE",0},{any,any,6, "GET /v",0}>
Youtube	<2472,{173.194.127.0/24,80,6, "GET /generate_204",0},{173.194.127.0/24,80,6, "GET /pagead/",0}>

TABLE 7 Accuracy of behavior signature

	Precision		Recall	
Name	**Flow**	**Bytes**	**Flow**	**Bytes**
uTorrent	100%	100%	90.29%	75.77%
DropBox	100%	100%	100%	100%
FaceBook	100%	100%	78.46%	93.71%
Line	100%	100%	100%	100%
Skype	100%	100%	53.49%	6.32%
TVPot	100%	100%	97.08%	99.95%
Youtube	100%	100%	68.52%	93.16%

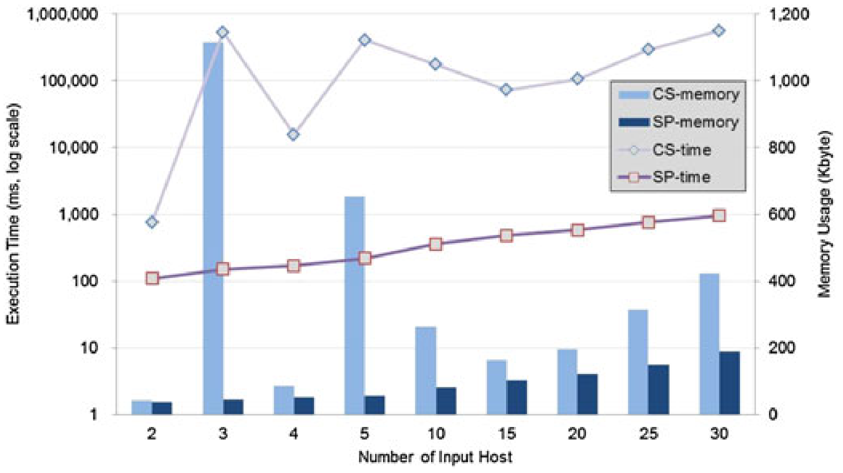


FIGURE 5 Comparison of memory use and execution time between *Candidate-Selection* (CS) and *Sequence Pattern* (SP)

candidate signatures over a certain threshold. In contrast, the SP method proposed in this paper uses a sequence pattern algorithm for extracting a signature. For a precise performance measurement of these 2 methods, we tested them under various conditions while increasing the number of hosts generating the test traffic. Traffic volume increases as the number of hosts increases.

In terms of memory use, the proposed method is about 6 to 3563 times faster and 2 to 11 times lighter than the previous method. The CS method shows a large amount of memory use and a longer execution time because it should list all possible combinations from all extracted entries. If the number of entries from the test traffic is n, the number of combinations is 2^n. However, the proposed SP method only uses length L candidate signatures over a certain support value to generate length $L + 1$ candidate. Again, any candidates under the support value are eliminated in the early stage, increasing the efficiency of the proposed method over the previous method.

7 | CONCLUSION AND FUTURE WORK

In this paper, we proposed an effective behavior signature extraction method using sequence pattern algorithms to overcome the limitations of previous works. We used 7 popular applications to prove the feasibility of the proposed signature method. Although some applications show a slightly low recall, the precision is 100% for all applications, which means that all extracted signatures correctly identified the corresponding traffic. A comparison test against a previous method, *Candidate-Selection*, shows that the proposed method is faster and lighter.

As future research, we plan to improve the extraction algorithm by applying it to various networks and applications. Moreover, we plan to develop an identification system on the basis of the proposed signature for operation in a real network. Although the use of an interflow unit has many advantages, it also has certain disadvantages. As an example, an interflow unit operates under the assumption that plural flows occur when a single function is conducted. If a single function makes a single flow, a behavior signature is not applicable. We plan to address the solution of this limitation in our future work.

ACKNOWLEDGEMENTS

This work was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (no. 2017-0-00513, developing threat analysis and response technology based on Security Analytics for Heterogeneous security solution).

ORCID

Sung-Ho Yoon http://orcid.org/0000-0002-0266-4973
Myung-Sup Kim http://orcid.org/0000-0002-3809-2057

REFERENCES

1. Wang Y, Xiang Y, Zhou WL, Yu SZ. Generating regular expression signatures for network traffic classification in trusted network management. *Journal of Network and Computer Applications*. May 2012;35:992-1000.
2. Park B, Won Y, Chung J, Kim MS, Hong JWK. Fine-grained traffic classification based on functional separation. *International Journal of Network Management*. Sep 2013;23:350-381.
3. Sung-Ho Y., Jun-Sang P., Myung-Sup K., ChaeTae L., and JunHyung C. Behavior signature for big data traffic identification. In Big Data and Smart Computing (BIGCOMP), 2014 International Conference on 2014; 261-266.
4. Agrawal R. and Srikant R. Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB, 1994; 487-499.
5. Agrawal R. and Srikant R.. Mining sequential patterns. In Data Engineering, 1995. Proceedings of the Eleventh International Conference on 1995; 3-14.
6. CAIDA CoralReef Software Suite. Available: http://www.caida.org/tools/measurement/coralreef/
7. IANA port number list. Available: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml
8. Choi T., Kim C., Yoon S., Park J., Lee B., Kim H.. et al.. Content-aware internet application traffic measurement and analysis. In Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP. 2004; 511-524.
9. Roesch M., Snort: lightweight intrusion detection for networks. In LISA, 1999; 229-238.
10. Moore AW, Papagiannaki K. Toward the accurate identification of network applications. *In Passive and Active Measurement Workshop 2005*. PAM 2005: Boston, MA, USA; 2005:41-54.
11. Kim H, Jang JW. Reduction of power consumption for pipelined DPI systems on FPGA. *J Inf Sci Eng*. 2014;30:1395-1406.
12. Soysal M, Schmidt EG. Machine learning algorithms for accurate flow-based network traffic classification: evaluation and comparison. *Performance Evaluation*. Jun 2010;67:451-467.
13. Yuan RX, Li Z, Guan XH, Xu L. An SVM-based machine learning method for accurate internet traffic classification. *Information Systems Frontiers*. 2010;12:149-156.
14. Dong S, Zhou DD, Ding W, Gong J. Flow cluster algorithm based on improved K-means method. *Iete Journal of Research*. 2013;59:326-333.
15. Hu CC, Luo NAX, Yan XH, Shi WZ. Traffic flow data mining and evaluation based on fuzzy clustering techniques. *International Journal of Fuzzy Systems*. 2011;13:344-349.
16. Agrawal R., Imieli T., #324, ski, and Swami A.. Mining association rules between sets of items in large databases. Presented at the Proceedings of the 1993 ACM SIGMOD international conference on Management of data, Washington, D.C., USA. 1993:207-216.
17. Zhao Q. and Bhowmick S. S., Sequential pattern mining: a survey. ITechnical Report CAIS Nayang Technological University Singapore. 2003; 1-26.
18. Pei J., Han J., Mortazavi-Asl B., Pinto H., Chen Q., Dayal U. et al.. PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In 2013 IEEE 29th International Conference on Data Engineering (ICDE). 2001; 0215-0215.
19. Park JS, Yoon SH, Lee SK, Won YJ, Kim MS. Performance improvement of traffic classification based on application traffic locality. *J Inf Sci Eng*. 2016;32(5):1241-1259.
20. Aho AV, Corasick MJ. Efficient string matching: an aid to bibliographic search. *Commun ACM*. 1975;18:333-340.
21. Park B.-C., Won Y. J., Kim M.-S., and Hong J. W.. Towards automated application signature generation for traffic identification. In Network Operations and Management Symposium, 2008. NOMS 2008. IEEE. 2008; 160-167.

Kyu-Seok Shim is a PhD degree student in the Department of Computer and Information Science, Korea University, Korea. He received his BS and MS degree in the Department of Computer Science and the Department of Computer and Information Science, Korea University, Korea, in 2014 and 2016, respectively. His research interests include internet traffic classification and network management.

Sung-Ho Yoon received his BS, MS, and PhD degree in Computer Science from Korea University, Korea, in 2009, 2011, and 2015, respectively. He is currently a researcher of Vehicle Components Company, LG Electronics, Korea. His research interests include internet traffic classification and network management.

Baraka D. Sija is an MS degree student in the Department of Computer and Information Science, Korea University, Korea. He received his BS degree in Information and Communication System from Semyung University, Korea, in 2016. His research interests include internet traffic monitoring and analysis, service and network management, and internet security.

Jun-Sang Park received the BS and MS degree in Computer Science from Korea University, Korea, in 2008 and 2010, respectively. He is currently a PhD candidate student of Korea University, Korea. His research interests include internet traffic classification and network management.

Kyunghee Cho received her BA in English Literature from Syung-gyul University in 1994 and her MA in English Education from Yonsei University in 1997. In 2013, she started her PhD work in Computer Science at Korea University. Her current work in Software Engineering Lab is focused on the home network system related to internet of things.

Myung-Sup Kim received his BS, MS, and PhD degree in Computer Science and Engineering from POSTECH, Korea, in 1998, 2000, and 2004, respectively. From September 2004 to August 2006, he was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University, Korea, in 2006, where he is working currently as an associate professor in the Department of Computer and Information Science. His research interests include internet traffic monitoring and analysis, service and network management, and internet security.

How to cite this article: Shim K-S, Yoon S-H, Sija BD, Park J-S, Cho K, Kim M-S. Effective behavior signature extraction method using sequence pattern algorithm for traffic identification. *Int J Network Mgmt*. 2017;e2011. https://doi.org/10.1002/nem.2011