IEICE TRANS. INF. & SYST., VOL.E98–D, NO.11 NOVEMBER 2015

LETTER

HTTP Traffic Classification Based on Hierarchical Signature Structure

Sung-Ho YOON[†], **Jun-Sang PARK**[††], **Ji-Hyeok CHOI**[†††], **Youngjoon WON**[††††], ***Nonmembers***, ***and*** **Myung-Sup KIM**[†a)], ***Member***

SUMMARY Considering diversified HTTP types, the performance bottleneck of signature-based classification must be resolved. We define a signature model classifying the traffic in multiple dimensions and suggest a hierarchical signature structure to remove signature redundancy and minimize search space. Our experiments on campus traffic demonstrated 1.8 times faster processing speed than the Aho-Corasick matching algorithm in Snort.

key words: *HTTP traffic classification, payload signature, signature hierarchy, hash*

1. Introduction

HTTP is a communication protocol for web browsing, P2P, multimedia, and many others. Most enterprise networks offer blocking at the application protocol level through a firewall; however, HTTP is allowed without intervention. The service providers offer their services that can communicate over HTTP. Under this circumstance, the classification taxonomy needs multi-dimensional criteria (e.g., application, function), not the L7 protocol criteria [1]. For example, it is not obvious how to classify the HTTP traffic from the YouTube service. The L7 protocol in use is HTTP; however, the service name is YouTube. To classify the HTTP traffic in more detail and in a different manners, we need to define classification criteria that address the HTTP features and the signature model. For better accuracy and classification completeness, a payload signature-based classification method is appropriate. However, the performance bottleneck of the signature-based classification must first be resolved regarding HTTP types and heavy traffic services.

García-Dorado *et al.* [2] reported that HTTP traffic accounts for over 30% of ISP and campus network. M. Baldi *et al.* [3] suggested a method grouping traffic based on server IP with seven major services offered via HTTP. However, these studies have an accuracy issue because they could not provide subdivided services. For accuracy reason, the packet payload must be examined. Snort [4], an open source intrusion detection system, analyses payload signature. 70% of the whole processing time dedicates to the signature pattern matching. Its performance relies on time complexity of matching algorithm and search space. Kawano *et al.* [5] suggest that hash-based matching from the malignant URL detection field is an appropriate method to minimize signature search space. To accomplish this, the matching process is performed only on signatures that coincide with a given hash-key value.

In this paper, we define a payload-signature model that can classify HTTP traffic into multi-dimensional classes. It stratifies the HTTP fields to determine the suggested criteria. For performance enhancement, we minimize signature redundancy and search space by establishing a signature tree and hash-based matching. We evaluate our proposal using campus traffic and compare to the Aho-Corasick (AC) string-matching algorithm [6] in Snort. Our proposal demonstrates 1.8 times faster in processing speed.

The remainder of this paper is organized as follows. Section 2 explains our HTTP signature model and classification method. Section 3 the validation on the campus traffic. Finally, Sect. 4 concludes this paper.

2. HTTP Traffic Classification

In this section, we describe a method for multidimensional and rapid classification of HTTP traffic.

2.1 HTTP Signature Model

We apply flow-based traffic classification. A flow can be seen from the viewpoint of a service, application, and function. For example, assume that a client watches a YouTube video using Internet Explorer. If the traffic is divided into the above three criteria: service, application, and function, then they correspond to YouTube, Internet Explorer, and video streaming, respectively. Table 1 defines the multi-dimensional classification criteria. It can expand users' understanding of traffic.

We define an HTTP signature model based on Table 1 criteria. We first extract the information that can distinguish each criterion, namely User-Agent, Host, and URI fields

Table 1 Classification criteria.

Criteria	Definition
Service	All forms of IT services are to offer content to user
Application	Application program used by clients for service
Function	Purpose of the traffic by user behavior

Manuscript received September 24, 2014.
Manuscript revised July 22, 2015.
Manuscript publicized August 19, 2015.
[†]The authors are with Korea Univ., Korea.
[††]The author is with LG Electronics Inc., Korea.
[†††]The author is with ETRI, Korea.
[††††]The author is with Hanyang Univ., Korea.
a) E-mail: tmskim@korea.ac.kr (Corresponding author)
DOI: 10.1587/transinf.2014EDL8191

(a) Request packet

GET /checkupdate.php?cl=&v=103246420 HTTP/1.1
Host : update.utorrent.com:7070
User-Agent : BTWebClient/3130(27220)

(b) Corresponding HTTP signature model

<signature code = "1">
<msg = "utorrent – BTWebClient – update">
<payload = User-Agent : BTWebClient, Domain : utorrent, Host : update, URI: checkupdate>

Fig. 1 HTTP request packet and its corresponding signature model.

Table 2 HTTP signature field.

Field	Extraction Method
User-Agent	From the end of the User-Agent keyword to before '/', \n' (version information is excluded from signature)
Domain	Defines keywords that are placed in front of Level 1 Domains such as .com as domain (except IP and port number)
Host	Defines keywords that are placed immediately ahead of Domain as Host
URI	Defines from after the Method field to before the HTTP version as URI (extracted directly by administrator)

from the HTTP request packet in Fig. 1 (a). The URI field consists of the attributes and their values, representing function, for the requested data. From the keyword 'checkupdate' in the URI, we can determine that this is an update function. The host field can be divided into domain (utorrent.com) and host names (update). The domain is used as a signature to identify the service name; the host indicates the function by its domain. The user-agent field specifies the application program. The program appears to be BTWebClient, an uTorrent service. Figure 1 (b) represents our signature format consisting of four fields. The signature code is an identifier of the signature and msg specifies service, application, and function. The payload specifies the signatures corresponding to user-agent, domain, host, and URI. If it contains the three classification criteria, it is registered as a signature. Base on the long-term analysis, we assure that normal HTTP traffic contains the four fields indicating proposed criteria. Thus, the proposed HTTP signature model covers HTTP traffic except one of abnormal. Table 2 shows the fields for signature extraction.

2.2 Signature Tree

We construct a signature tree to remove any redundancy in the HTTP signatures. Keywords, such as Mozilla and YouTube, commonly appeared at multiple locations. This is denoted as signature redundancy. Because there are many types of HTTP-based services that require numerous signature instances, signature redundancy slows down processing speed. The tree can minimize the signature search space of the classification system because it searches only the child. Figure 2 presents a tree structure of seven signatures.

The tree consists of four levels, called 1-level (user-agent), 2-level (domain), 3-level (host), and 4-level (URI). Each level is defined based on the inclusion relationship among the signatures. With respect to Table 2, we state that level is formed starting from the user-agent, which has a small number of unique signatures for domain, host, and

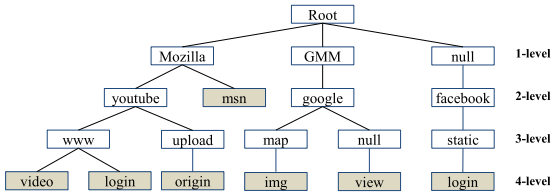

Fig. 2 HTTP signature tree.

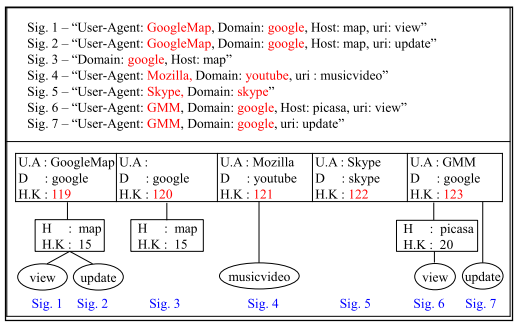

Fig. 3 Signature tree based on hash key.

URI in order. Upon completion of the HTTP flow, signature matching is performed on the flow progressively from 1-level to 4-level in the tree. It finishes when a leaf node is encountered.

The signature tree can divide traffic into service, application, and function. This tree can solve the signature-overlapping problem because the overlapped strings per level create a single node. It can build a single-tree regardless of the number of signatures. The signature tree can classify traffic with a single search because the entire signatures consist of a tree. We compose each level of the tree as a hash table, consequently minimizing the signature search space. The hash functions are the Executable and Linkable Format (ELF) function and the Bob Jenkins (BJ) function [7], [8]. We create each hash using both functions, and use one with better performance, determined by comparing key collisions and processing speed.

2.3 Hash-Based Matching

Figure 3 shows an example of hash-based tree. It consists of three levels. 1-level is obtained by combining user-agent and domain. 2-level and 3-level consist of the host and URI, respectively.

A difference from the signature tree is that the user-agent and domain levels are aggregated into a single hash node in order to increase the matching speed. The user-agent and domain-level, indicating application and service, can be a matching key because each field value is extracted under the same policy in the process of signature extraction and signature matching. Thus, it is more effective to match into a single after applying previous levels together. The host and URI field-identifying functions are signatures that require the intervention of administrator. It is possible for the signature field values and matching traffic to be differ-

```
1:  SHT : Service Level Hash Table
2:  HHT : Host Level Hash Table
3:  UL_SHT : Sub URI list of SHT node
4:  UL_HHT : Sub URI list of HHT node
5:  F: Flows, TF: Termination Flag
5:
6:  //if setsigCode()is called, it inspects a next flow
7:  for F_0 - F_n do
8:      if sig = ServiceKeyMatch(F_n, SHT)
9:      if ServiceTextMatch(F_n, sig)
10:                 if sig.TF   setSigCode(F_n)
11:     else   return unknown
12:
13:     if sig = HostKeyMatch(F_n, HHT)
14:             if HostTextMatch(F_n, sig)
15:                 if sig.TF              setSigCode(F_n)
17:                 if URITextMatch(F_n, UL_HHT)  setSigCode(F_n)
18:       else
19:                 if URITextMatch(F_n, UL_SHT)  setSigCode(F_n)
20:     else
21:             if URITextMatch(F_n, UL_SHT)    setSigCode(F_n)
22:   else   return unknown
23: done
```

Fig. 4 Hash-based matching algorithm

ent. For this reason, our model does not aggregate compose the host and URI fields into a single level. Unlike other levels, the URI level is composed of strings that are not a key value. The URI field cannot be a matching key with the URI field that exists in the HTTP traffic because it does not use the entire field as signature. Rather, it defines necessary parts according to administrator's decision.

Figure 4 presents the pseudocode of matching hash tree and flow. A single input flow starts the key matching from the user-agent and domain (U-D) level. If the key matching succeeds, it continues with text matching. If text matching agrees, then it confirms the ending flag. Matching can be closed from the U-D level if the ending flag is set. However, we examine at the host and URI levels, and its analysis is performed with a signature of the U-D level if the examination fails. A host-level examination assigns priority on key matching, in the same manner as the U-D level. If the matching succeeds, it continues to text matching. If the text matching succeeds, it verifies the ending flag and continues on URI level matching. If the ending flag is set at the host-level and the URI matching fails, then the analysis terminates at the host level. If the URI level matching succeeds, the analysis is done at the URI level. If there is no host level, the matching starts at the URI level.

3. Evaluation

We collected one-day sample traffic of 3,000 hosts at the campus network. The HTTP traffic took 25.8%, 42.2%, and 48.6% in flow, packet, and byte. Approximately 50% of the total byte was HTTP traffic and this tends to increase. Figure 5 illustrates a distribution of HTTP signatures for classification. Using HTTP traffic, we generated 1,065 HTTP signatures specified three criteria such as service, application, and function. Because there is traffic having insufficient information to extract the three criteria, some signatures have only one or two criteria. The venn diagram indicates number of signature having the same criteria set. For example, three criteria are specified in 406 signatures, and only 562 signatures have service and application criteria.

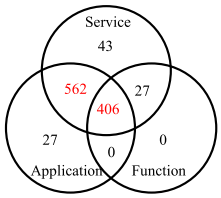

Fig. 5 HTTP signature distribution.

Table 3 Classification result on campus traffic.

Service	App.	Function	Flow	Packet	Byte
Naver	Mozilla	Browsing	1,482 K (37.2%)	52,470 K (26.3%)	43,172 M (25.8%)
Naver	Mozilla	News	649 K (16.3%)	43,954 K (22.1%)	31,760 M (18.9%)
Naver	Mozilla	Ad	470 K (11.8%)	19,134 K (9.6%)	16,540 M (9.9%)
Daum	Mozilla	Browsing	158 K (3.9%)	11,163 K (5.6%)	11,319 M (6.8%)
Daum	Mozilla	News	69 K (1.7%)	13,435 K (6.7%)	10,420 M (6.2%)
Daum	Mozilla	Ad	62 K (1.5%)	6,458 K (3.2%)	5,523 M (3.3%)
Google	Mozilla	Browsing	40 K (1.1%)	3,415 K (1.7%)	3,248 M (1.9%)
Korea U	Mozilla	Browsing	35 K (1.8%)	2,224 K (1.1%)	2,222 M (1.3%)
Korea U	Mozilla	Mail	32 K (0.8%)	2,024 K (1.0%)	2,104 M (1.2%)
Nate	NateOn	-	27 K (0.7%)	1,913 K (0.9%)	1,821 M (1.1%)

The analysis results showed that 1,065 signatures classified 66.2%, 74.7%, and 78.6% in flow, packet, and byte, respectively. Its classification rate increased when more service signatures were found. Table 3 presents the classified results by the signatures with more than two criteria from service, application, and function. The majority of those could be divided into service and application criteria.

In the campus traffic, the most used service was 'Naver', a #1 portal service in Korea. From the service point of view, we observe a heavy traffic from popular portal sites such as Naver and Google. The identified functions are 'browsing', 'news', 'ad', and 'mail' as in order of popularity. In-depth classification of HTTP traffic is possible using our multidimensional criteria and signature tree.

To evaluate the impact of BJ and ELF, we compute the average conflict as the ratio of total signature over total key. The average conflict reflects the degree of change and signature is the only hash key value. If it is equal to 1, it indicates that the signature responds as the only key value. If it is greater than 1, it implies a key collision. Figure 6 compares the average conflict values of BJ and ELF according to hash size. BJ is relatively less influenced by the decrease of hash size. If the hash size becomes large, both hash functions are converging to 1. Thus, BJ shows a better performance and it is our choice for hash function.

Figure 7 shows processing times for the BJ tree and the AC algorithm. Processing time can be determined as

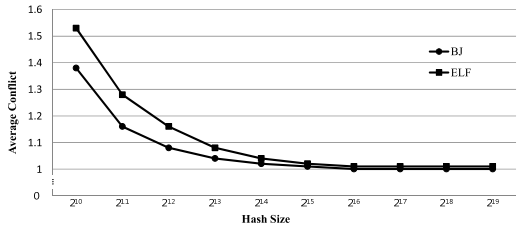

Fig. 6 Average conflict of hash size.

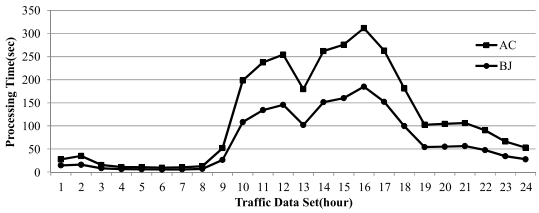

Fig. 7 Processing time: AC vs. BJ.

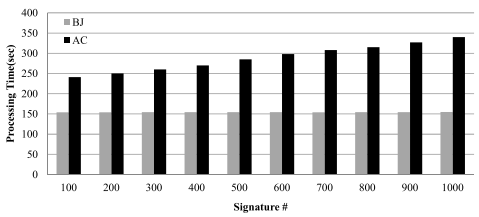

Fig. 8 Processing time according to the number of signatures.

the time required to extract the necessary fields from the traffic, create a hash key, and match the fields with those of the actual traffic. Both processing times show minimal difference when the traffic volume is relatively small. Over the entire dataset in Fig. 7, the difference was 4 sec, 126 sec, and 52 sec for min, max, and average. In fact, our BJ tree shows approximately 1.8 times faster than AC.

The processing speed of general-purpose traffic classification systems can be slow when the number of signatures for matching increases. On the contrary, Fig. 8 shows that BJ does not have significant performance degradation in speed while AC in Snort slows down. Because BJ is based on hash function, it works in constant processing time regardless of the number of signature. In contrast, AC reconstitutes the signature as a finite set. The size of finite set increases according to the number of signature. Therefore, the processing time proportionally increases with the number of signature.

4. Conclusions

We suggested multidimensional traffic-classification criteria considering various HTTP classification requirements. We defined three classification criteria and signature tree to resolve signature redundancy and minimize its search space. We used hash functions to stabilize the process speed regardless of the number of signatures. Our experiment on the campus traffic showed 1.8 times faster in process speed than the conventional AC algorithm in Snort. For future work, we plan to deal with the encryption traffic for HTTP services.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2014R1A1A2057301, 2015R1D1A3A01018057), and Brain Korea 21 Plus (BK21+).

References

[1] J.-H. Kim, S.-H. Yoon, and M.-S. Kim, “Study on Traffic Classification Taxonomy for Multilateral and Hierarchical Traffic Classification,” Proc. APNOMS, Seoul, Korea, Sept. 2012. DOI: 10.1109/APNOMS.2012.6356105

[2] J.L. García-Dorado, J.A. Hernández, J. Aracil, J.E. López de Vergara, F.J. Montserrat, E. Robles, and T.P. de Miguel, “On the Duration and Spatial Characteristics of Internet Traffic Measurement Experiments,” Proc. IEEE Commun. Mag., vol.46, no.11, pp.148–155, Nov. 2008. DOI: 10.1109/MCOM.2008.4689258

[3] M. Baldi, Politec. di Torino, Turin, A. Baldini, N. Cascarano, and F. Risso, “Service-based traffic classification: Principles and Validation,” Proc. Sarnoff Symposium, NJ, USA, March 2009. DOI: 10.1109/SARNOF.2009.4850330

[4] K. McAreavey, W. Liu, P. Miller, and K. Mu, “Measuring inconsistency in a network intrusion detection rule set based on snort,” Proc. International Journal of Semantic Computing, vol.5, no.03, pp.281–322, Sept. 2011. DOI: 10.1142/S1793351X11001274

[5] S. Kawano, T. Okugawa, T. Yamamoto, T. Motono, and Y. Takagi, “High-Speed DPI Method Using Multi-Stage Packet Flow Analyses,” Proc. APSITT, Santiago, Nov. 2012.

[6] M. Finsterbusch, C. Richter, E. Rocha, J. Muller, and K. Hanssgen, “A Survey of Payload-Based Traffic Classification Approaches,” Proc. Communications Surveys & Tutorials, IEEE, vol.16, no.2, pp.1135–1156, Oct. 2013. DOI: 10.1109/SURV.2013.100613.00161

[7] Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2 (May 1995).

[8] B. Jenkins, “A New Hash Function for Hash Table Lookup,” 1997.