A Fast Pattern-Matching Algorithm for Network Intrusion Detection System*

Jung-Sik Sung[1], Seok-Min Kang[2], Taeck-Geun Kwon[2]

[1] ETRI, 161 Gajeong-dong, Yuseong-gu, Daejeon, 305-700, Korea
jssung@etri.re.kr
[2] Chungnam National University, 220 Gung-dong, Yuseong-gu, Daejeon, 305-764, Korea
{esemkang, tgkwon}@cnu.ac.kr

Abstract. We present a multi-gigabit rate multiple pattern-matching algorithm with TCAM that enables protecting against malicious attacks in a high-speed network. The proposed algorithm significantly reduces the number of TCAM lookups per payload with *m*-byte *jumping window* scheme. Due to the reduced number of TCAM lookups, we can easily achieve multi-gigabit rate for scanning the packet payload in order to inspect the content. Furthermore, multi-packet inspection is achieved easily by the extended state transition diagram with the *shifting distance*. With experimental results, we have clearly justified the proposed algorithm works well for a multi-gigabit network intrusion detection system.

1. Introduction

Network intrusion detection systems (NIDSs) monitor every packet in the network to detect malicious attacks. In a high-speed network, an NIDS may be overloaded as the packet arrival rate becomes high. Hence, the hardware-based approach of implementing the NIDS will be appropriate in order to support the high-speed network. Some researches [1], [2], [3] focus on the hardware implementation to achieve the line-speed intrusion detection. Recently, technologies for high performance network processors have driven a new breed of solutions that perform at high data rates while remaining flexible through software [4].

There are many approaches for solving multiple pattern-matching problems. The multiple pattern-matching algorithms [5], [6] use software approaches. However, software-based pattern-matching is not able to inspect all packets in the high-speed network. Gigabit rate pattern-matching algorithms such as [7], [8] are TCAM-based algorithms that can be used with TCAM. In this paper, the scheme in [7], [8] is referred to a '*sliding window*' in which every one-byte shifted fixed-length partial payload should be examined to match the TCAM and the partial payload should be extracted with a sliding window manner. Although the sliding window based pattern matching is intuitive and simple, the scheme has three problems. First, it has lower

* This research was supported in part by ITRC program of the Ministry of Information and Communication, Korea. *Correspondence to T. -G. Kwon*

scan speed. It can provide an answer for searching a packet of length n, in a deterministic time of $O(n)$ TCAM lookups, because one TCAM lookup is needed for every byte position in the packet. Since the TCAM lookup time is known and fixed, we need to minimize the number of TCAM lookups per packet to support the multi-gigabit rate NIDS. Second, it is very complicated and needs more memory when the pattern is longer than TCAM width. Suppose the width of the TCAM is w bytes and let $T = t_0,t_1,\ldots,t_{n-1}$ be the text. Partial pattern is matched at $t_i,\ldots,t_{i+w-1}$, it should check whether occurrence of previous partial matching at $t_{i-w},\ldots,t_{i-1}$, and keeps matching information for next partial matching. Third, it does not support multi-packet inspection where the pattern split into continuous two payloads. This situation is common in NIDS, where intrusion signatures can be segmented into packets which contain the user data such as E-mails, attached files, etc.

In this paper, we revise deep packet inspection algorithm introduced in our recent paper [9] and extend the algorithm to provide content inspection over multiple packets. In our algorithm, TCAM lookups for searching a packet of length n, is $O(n/m)$, if the size of the jumping window is m. We devise the state transition diagram for keeping previous partial matching when the pattern is longer than TCAM width. So it is very simple and does not waste memory. In order to support multi-packet inspection, we use extended state transition diagram by alignment of the last jumping window of previous payload. In addition, we have implemented the proposed algorithm using Intel IXP28XX network processors (NPs) with TCAM. We have some preliminary experimental results which verify the proposed scheme improves significantly the performance of deep packet inspection.

The rest of the paper is organized as follows. In Section 2, we describe problems of multiple pattern-matching using TCAM. We explain the jumping window algorithms to map the multiple patterns into TCAM and efficiently scan packets at high speeds in Section 3. In Section 4, we extend the algorithm in order to inspect multiple packets. In Section 5, we give experimental results with our 10Gbps network processor based NIDS. Finally, we conclude the paper.

2. Jumping Window Pattern Match Algorithm

A pattern is a string to be searched for a payload and it usually appears at an arbitrary position in the payload. For example, virus and worm patterns are located in an attached file and they may appear at any position in the packet payload. We should search several sub-patterns relevant to each jumping window for matching a pattern if the pattern would be occupied into continuous several jumping windows of a payload. In other words, we can create TCAM entries, all possible position-aware patterns (PAPs) from one pattern. Therefore, we succeed pattern-matching with jumping-window scheme although the pattern appears at an arbitrary position in the payload.
Let $T = t_0,t_1,\ldots,t_{n-1}$ be the payload, and its length be n bytes. Let $P = p_0,p_1,\ldots, p_{m-1}$ be the pattern to be searched, and its length be m bytes. P is located in the substring of T, where

$$t_s,\ldots,t_{s+m-1} = p_0,\ldots,p_{m-1},\ 0 \le s \le n-m. \qquad \textbf{(1)}$$

The payload is divided into multiple jumping window substrings with m-byte window, where

$$t_{sm},\ldots,t_{(s+1)m-1}, 0 \le s \le \left\lceil \frac{n-m}{m} \right\rceil. \quad \textbf{(2)}$$

If (m-i) sequential bytes of P is found from ith position within sth jumping window substring of T, where

$$t_{sm+i},\ldots,t_{(s+1)m-1} = p_0,\ldots,p_{m-1-i},\ i=0,1,2,\ldots,m-1. \quad \textbf{(3)}$$

Then, the rest of P, the remaining i bytes is found from the first position within (s+1)th jumping window substring of T. On this occasion, P is found in T. Therefore, we can make PAPs from P with 0~(m-1) shifting and can split PAPs into fixed-size sub-patterns, position-aware sub-patterns(PASes) as shown in Fig. 1 ('-' denotes "don't care" state of TCAM). We can match the pattern P in the payload T with jumping window scheme when these PASes generated from P are stored in the TCAM.

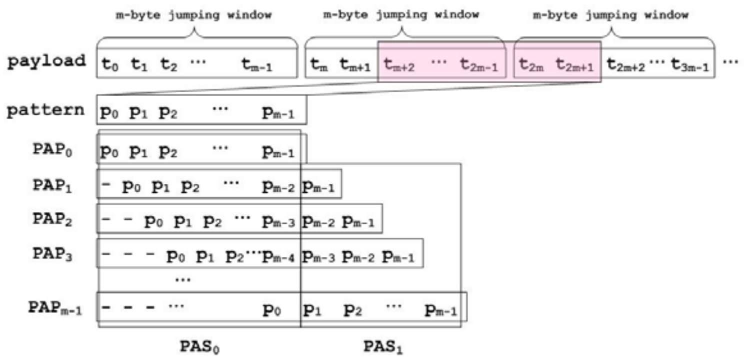

Fig. 1. Position-aware patterns and position-aware fixed sub-patterns

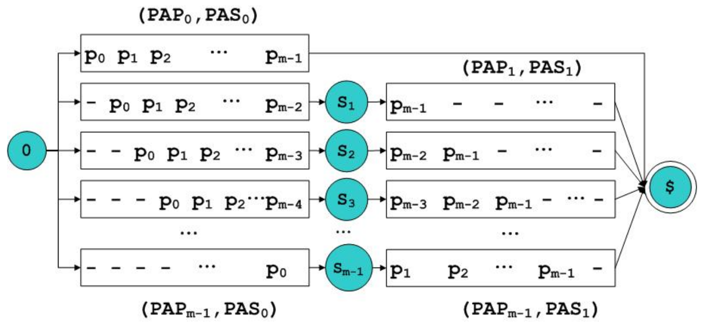

Fig. 2. State transition diagram

Given the pattern of "GATT" the position of the pattern in the payload is one of "GATT," "–GATT," "––GATT," …, "(m -1)–GATT." When m is 4, the pattern may be found at the different position of the payload such as "GATT," "–GATT," "––GATT", or "–––GATT." We put the above derived patterns into the TCAM table. Then, a TCAM lookup operation is carried out for every segment of m bytes called a jumping window for a packet payload. Usually, the width of the TCAM, which will be used for matching the pattern in a parallel way, is fixed. Therefore, if the TCAM width is smaller than the pattern, we have to split a long pattern into shorter sub-patterns with the same length of the TCAM width. If one PAP splits into several PASes, a pattern-matching operation will be completed when all PASes are matched to the TCAM entries in series. Hence, for the matching operation of multiple PASes, a PAS matching function requires the result of the previous PAS matching operation. To increase the speed of searching, we employ the state transition diagram to find the result of the previous PAS matching operation as shown in Fig. 2.

3 Multi-packet inspection

We consider a pattern that split on two payloads T_i, T_{i+1} and it usually appears at several jumping windows of T_i, T_{i+1}. That is, it split into tail end of the former and beginning of the latter. In case of Fig. 3 (a), the pattern $P = p_0,p_1,\ldots,p_{m-1}$ split into "p_0" and "$p_1p_2\ldots p_{m-1}$" in T_i and T_{i+1}, respectively. At the window before last of T_i as illustrated in Fig. 3 (a), pattern-matching does not occur and the state is initial state, 0. We fit the last window with shifting distance because the remaining bytes are too small to fit into the jumping window. There is no previous pattern-matching, the state of the last window is initial state, 0. However in case of Fig. 3 (b), the partial pattern "p_0" is matched and transits to state S_{m-1}. Due to lookup TCAM with *m*-byte window, the last window of T_i needs *m*-byte alignment. The start point of last window is shifted to *m*-(*n%m*) bytes left. We call it *shifting distance*. In order to fit the last window, the pre-condition state should be changed according to the shifting distance. For example, state S_{m-1} must move into initial state 0 if 3 bytes, i.e., "‒‒p_0," are shifted for fitting the last search window.

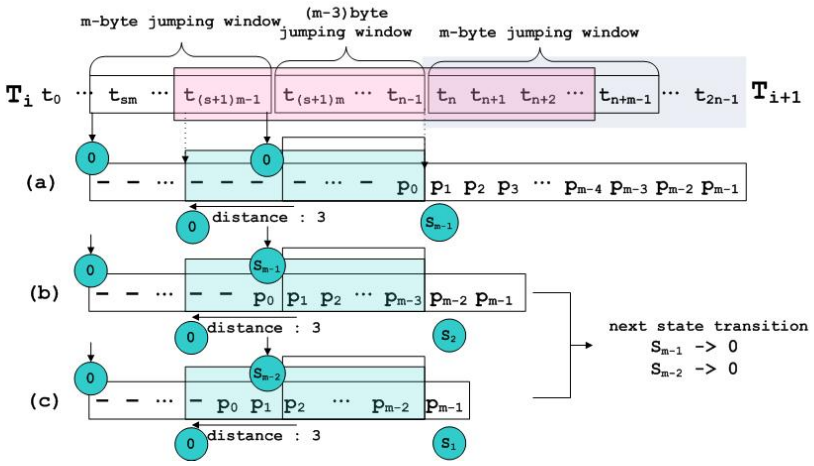
m-byte jumping window
(m-3) byte jumping window
m-byte jumping window
T_i t_0 ··· t_{sm} ··· $t_{(s+1)m-1}$ $t_{(s+1)m}$ ··· t_{n-1} t_n t_{n+1} t_{n+2} ··· t_{n+m-1} ··· t_{2n-1} T_{i+1}
(a)
0
0
‒ ‒ ··· ‒ ‒ ‒ ‒ ··· ‒ p_0 p_1 p_2 p_3 ··· p_{m-4} p_{m-3} p_{m-2} p_{m-1}
distance : 3
S_{m-1}
(b)
S_{m-1}
‒ ‒ ··· ‒ ‒ p_0 p_1 p_2 ··· p_{m-3} p_{m-2} p_{m-1}
distance : 3
S_2
(c)
S_{m-2}
‒ ‒ ··· ‒ p_0 p_1 p_2 ··· p_{m-2} p_{m-1}
distance : 3
S_1
next state transition
S_{m-1} -> 0
S_{m-2} -> 0

Fig. 3. Example of multi-packet inspection processing: shifted state transition for alignment of a search window

For the split pattern matching, states can move to other states in order to align the last window if the previous partial match is done successfully. The state transition diagram should have this state transition information for the alignment. With the extended state transition, split pattern into the next packet payload can be easily matched. Furthermore, it requires storing only the last state information instead of the large packet reassembly buffer.

4 Performance Evaluation

For the evaluation of the multi-gigabit rate pattern-matching in NIDS, we have implemented IDS microblock which is a microcode program of Intel IXP28XX NP [10] to detect intrusion patterns in the packet payload. The IXP28XX NP development platform consists of dual network processor units (NPUs), 9-Mbit IDT's TCAM [11], and 10 ports of gigabit Ethernet (GbE). In this paper, we have generated packets matched with the Snort rule header using the traffic generator Smartbits 6000B. In the following experiments, throughputs are measured for various packet lengths with a single microengine (ME). Although the Intel IXP28XX NPU has 16 MEs, only 12 MEs are used to receive and transmit packets through external interfaces and process them in the current Intel's 10-port GbE IPv4/IPv6 forwarding application. We could add at most 4 MEs without modification of the current application for implementing our algorithm.

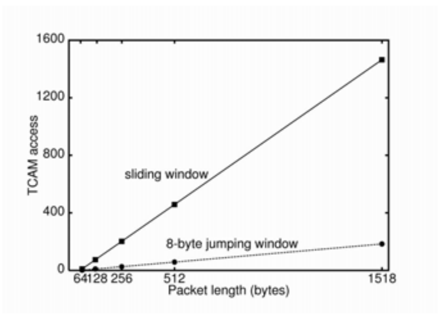

Fig. 4. Compares of TCAM Access

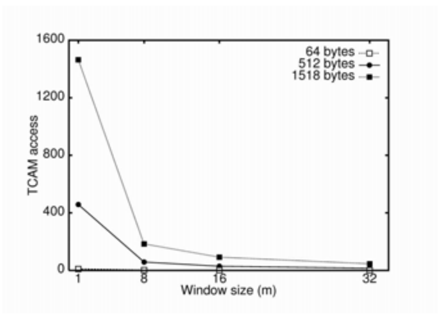

Fig. 5. Effects of the window size, *m*

Fig. 4 shows the result of TCAM access for pattern matching with varying the packet size for sliding window and 8-byte jumping window. For this experiment, only one ME is used for deep packet inspection among 16MEs. The number of TCAM access of sliding window increases rapidly as the packet length increases, while that of our algorithm increases slowly. With the maximum packet size, i.e., 1518 bytes in the Internet, the throughput of our proposed algorithm is about 1Gbps, while sliding window shows only the performance of about 200Mbps[12]. In this experiment, we proved that the number of TCAM access in the 8-byte jumping window is approximately 1/8 of the number of TCAM access in the sliding window scheme. Fig. 5 shows the effect of the window size, *m*. As the window size increases, the number of TCAM access is reduced 1/*m*.

5 Conclusion

In this paper, we have presented a multi-gigabit pattern-matching algorithm for network intrusion detection system in the high-speed network. The TCAM-based deep packet inspection algorithm developed in this paper uses a jumping window scheme, which is supported by position-aware sub-patterns and the state transition diagram to reduce the number of TCAM lookups. We have implemented the proposed algorithm on the Intel IXDP28xx platform. The performance of packet processing with our proposed algorithm is more than 3Gbps at the worst-case situation with the maximum packet size. We expect an increase of the performance through microcode optimization and window size augment. We've proven the feasibility of the proposed algorithm with our experimental implementation that runs on the IXDP28xx platform. In order to detect malicious attack split in two continuous packet, we extended the state transition diagram with shifting distance. Therefore, first pattern-matching in next packet is achieved easily with the state transition of previous packet. In this paper, we describe that the pattern length is the same value as window size, m. However, our proposed algorithm is applicable even if it is greater or lesser than window size m.

References

1. S. Dharmapurikar, P. Krishnamurthy, T. S. Sproull and J. W. Lockwood: Deep Packet Inspection using Parallel Bloom Filters in IEEE Micro, Vol. 24, No. 1, Jan. 2004, 52-61.
2. J. Lockwood, J. Moscola, M. Kulig, D. Reddick, and T. Brooks: Internet Worm and Virus Protection in Dynamically Reconfigurable Hardware in Military and Aerospace Programmable Logic Device (MAPLD), Sep. 2003.
3. I. Sourdis and D. Pnevmatikatos: Fast, Large-Scale String Match for a 10Gbps FPGA-based Network Intrusion Detection System in Conference on Field Programmable Logic and Applications, Sep. 2003.
4. P. Jungck and S. S.Y. Shim: Issues in high-speed internet security in IEEE Computer Magazine, Vol. 37, No. 7, July 2004, 22-28.
5. M. Fisk and G. Varghese: Fast content-based packet handling for intrusion detection in Tech. Report CS2001-0670, UCSD, May 2001.
6. S. Wu and U. Manber: A fast algorithm for multi-pattern searching in Tech. Report, TR94-17, University of Arizona, May 1994.
7. J. Bo and L. Bin: High-speed discrete content Sensitive pattern match algorithm for deep packet filtering in Int'l Conf on Computer Networks and Mobile Computing, 2003.
8. F. Yu, R. H. Katz and T. V. Lakshman: Gigabit rate packet pattern-matching using TCAM in IEEE Int'l Conf on Network Protocols, Oct. 2004, 174-183.
9. J. Sung, S. Kang, Y. Lee, T. Kwon, and B. Kim: A Multi-gigabit Rate Deep Packet Inspection Algorithm using TCAM in IEEE Globecom, Nov. 2005.
10. Intel: Intel 2800 Network Processor in Hardware Reference Manual, Jan. 2004.
11. IDT: Integrated IP Co-Processor (IIPC) with QDR Interface in IDT75K52134/ IDT75K62134 User Manual, Sep. 2002.
12. S. Kang, I. Song, Y. Lee, and T. Kwon: Design and Implementation of a Multi-gigabit Intrusion and Virus/Worm Detection System in IEEE ICC, June 2006 (to appear).