Multi-Pattern Boyer-Moore - Horspool Algorithm based Hashing Function for Intrusion Detection System

Awsan A. Hasan
School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia
Email: aaha10_com070@student.usm.my

Nur' Aini Abdul Rashid, Muhannad A. Abu-Hashem, and Atheer A. Abdulrazzaq
School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia
Email: nuraini@cs.usm.my, {abu_hashem4, athproof}@yahoo.com

***Abstract*—Every day the networks are expanding in very large scale and running in very high speed. The Intrusion Detection System (IDS) becomes as an essential part in any new network structure. The IDS is relying on the string matching algorithm to detect any signature attack, but the high speed of the modern networks are preventing the IDS string matching algorithms to work properly. There is a need to develop a robust IDS string matching algorithms to overcome these weaknesses. In this paper, we continue to developing Boyer-Moore Horspool algorithm by adding a new multi-pattern hashing feature to its original structure, which is called MPH-BMH. The developed algorithm is able to reduce the matching time because it can compare a group of pattern at one time. The results show that MPH-BMH is around 50% faster than BMH and QS algorithms in which it can scan and match large number of network packets in the given time and consequently it can be very useful to work in high speed networks.**

***Index Terms*—IDS, string matching algorithms, multi pattern matching, hash function.**

I. Introduction

In any modern network the security is considered as an important factor to protect the users from any type of attack. The intrusion detection system is creating and becomes an essential component of any modern network. The IDS has the ability to monitor the network traffic and detect the signature attacks. There are two IDS approaches; signature based and anomaly based. The signature based detection approach is produce exact results and may generate low false alarm. This is because it compares and matches the content of packets with a set of rule policies. But its disadvantage is inability to discover the new attacks. Consequently, the IDS based signature needs frequently to update the rule policies [1]. On the other hand, the anomaly based detection produce a great performance and it is able to discover the new attacks, but it generates very high rate of false alarm, besides it consumes more system resources [2], [3]. In this paper we concentrate on the IDS based signature because it is commonly used in the networks.

The IDS which is running in high speed network is facing many problems such as the huge number of the traffic that passing through the networks and the increased number of rule policies which make the IDS is unable to processor and analyze all the network traffic in real time. Consequently, some of the packets are dropped without any checking [4], [5]. This is because, the IDS is attempting to look for small rate of abnormal network behaviors among a very huge number of normal network behaviors. Where the abnormal behaviors are considered very small compared to the normal behaviors [1], [6].

Another problem that prevents the IDS from working properly is the overhead, in which the string matching algorithms which are running inside the IDS detection engine is considered as the weakest point of IDS components. The string matching algorithms are comparing the content of incoming packets with the rule policies to detect the signature attack. This process is consuming almost of IDS processing time and consequently slows down the performance of IDS by 70% [7]. There is a real need to improve the performance of string matching algorithms which is running inside IDS. In this paper, we continue to developing our previous work Hash - Boyer-Moore – Horspool [8] by adding a multi-pattern feature to its previous structure to reduce the processing time and enhance the performance of IDS.

II. Related Work

String matching algorithms are considered as the critical part of the IDS components. These algorithms are performing what is called Deep Packet Inspection (DPI) where the content of each individual packets are deeply scanned looking for any signature attacks [9]. String matching algorithms are commonly applied in computer science applications and the IDS is one of these applications [10]. Aho-Corassick algorithm (AC) is a multiple string matching algorithm based DFA [11]. But this algorithm is consuming very large memory space [7].

Manuscript received December 15, 2012; revised March 10, 2013.

Boyer-Moore algorithm (BM) is an exact string matching algorithm based on a single pattern matching [12]. The disadvantage of this algorithm is inability to process pattern of small size properly [7]. Boyer-Moore Horspool Algorithm (BMH) is a modified version of original Boyer-Moore algorithm [13]. It performs fast matching compare to other algorithms regardless of the pattern size that process it, but its drawback is when different patterns size are applied simultaneously then its performance becomes slow [14]. Wu-Manber algorithm (WM) [15], is also based on the original Boyer-Moore algorithm, but it has the ability to process multiple pattern matching in single pass. However, the main drawback of WM algorithm is the inability to process a large number of rule polices that contain different pattern size [7]. E2xB [16] is multiple string-matching algorithm which is applied in IDS to give a higher performance. It is work based on mismatch of pattern instead of traditional matching idea. The drawback here is inability of this algorithm to work properly with small pattern size [7] besides, its high dependency on the cache memory to work which has small size compared to the main memory [17]. Piranha [18] is an algorithm which is creates to work for IDS. The algorithm is searching inside the content of packets for any rarest four bytes substring of pattern. The drawback of Piranha is inability to process pattern length that is smaller than four bytes. Besides, the algorithm is surfing from the collision problem.

Unfortunately, most of string matching algorithms that applied in IDS are not designed to fully adapt to work with IDS [19]. However, the proposed method in this paper is added a new feature to Boyer-Moore Horspool algorithm to reduce the required processing time, in which it reformulates the algorithm to work as a multi pattern matching instead of single pattern matching.

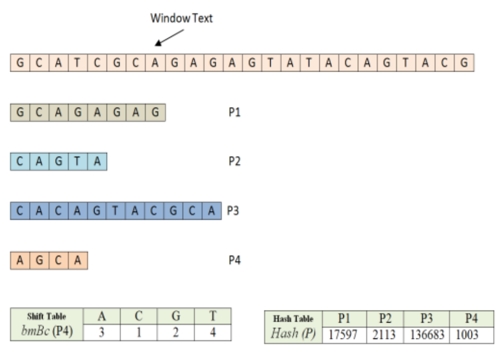
III. PROPOSED METHOD

In this section, we explain the idea of multi pattern Boyer-Moore Horspool by utilizing of the hash function that used in Karp-Rabin algorithm (KR) [20]. The new method is called Multi Pattern Hash Boyer-Moore Horspool algorithm (MPH-BMH). By referring to our previous work HBMH [8], we added the hash function only to original BMH algorithm to reduce the number of character comparisons that perform by it in each attempt as well as reduce the processing time. The new method boosts the performance of HBMH algorithm by reformulate it to work as multi pattern with the hash function. Like any string matching algorithm MPH-BMH has two phases; preprocessing phase and searching phase.

A. Preprocessing Phase

In preprocessing phase, the algorithm prepares the hash function table (which is calculated in the same manner of KR algorithm) for the entire patterns group being compared with window text. Besides, it prepares the bmBc table for only the shortest pattern of the patterns group. The choice of the shortest pattern is made to make the movement of patterns more accurate and to give more chance to perform more matching probability. If two or more short patterns have the same size then the algorithm will choose only one of them to build the bmBc shifting table.

For example suppose that we have the text (GCATCGCAGAGAGTATACAGTACG) and we have the flowing group of patterns that shown in Fig. 1. In the first step, the algorithm will calculate the hash table for this group of patterns. Second, the algorithm will choose the shortest pattern to create bmBc table which will control the movement for all patterns together after each comparison. After that, the algorithm will start the next phase which is the searching phase.



Shift Table	A	C	G	T
bmBc (P4)	3	1	2	4

Hash Table	P1	P2	P3	P4
Hash (P)	17597	2113	136683	1003

Figure 1. Preprocessing phase of MPH-BMH algorithm.

B. Searching Phase

In the searching phase, the MPH-BMH algorithm performs the comparison between the pattern group and the window text. The matching process starts by calculate the hash value for the window text which is variable in every shift of the patterns group. After that, the algorithm compares the hash value for window text with its corresponding patterns group. If the numerical value of the window text is identical with all patterns group or any one of the pattern in the group then the algorithm will compare all the characters of the identical patterns and window text one by one. Otherwise, the algorithm performs the shifting based on the value of the bmBc table. The operation will continue until all the patterns group compare with the window text as shown in Fig. 2.

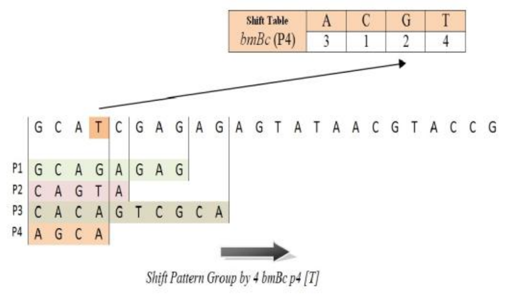


Figure 2. Searching phase of MPH-BMH algorithm

With the same manner the algorithm will perform the comparison and the shifting process based on the value determined in the bmBc table for the shortest pattern, until all patterns are shifting to the end of the text as shown in Fig. 3.

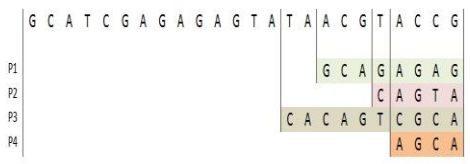


Figure 3. End of searching phase of MPH-BMH algorithm.

IV. Results

MPH-BMH algorithm is test using DEFCON 11 Trace data [21] which are trace files contain a very large quantity of suspicious attacks. Besides, we utilized of DARPA 1999 intrusion detection evaluation [22] which is another trace data contain files with attack and without attack. Finally, we used a trace files which are collected by wireless and wired sniffers (S1-S10) [23] user activities. In addition to, we utilized of Snort data set [24] as rule policies.

The experiments are performed on Intel Core i5 2.3 GHz machine with 4 GB of ram and 3MB of L2 cache. We compared MPH-BMH against BMH and quick search (QS) [25] algorithms with different size of these network trace files and we used around 4000 of Snort rule patterns. We run all the algorithms for 300 seconds and we recorded how many packets can be matched with Snort rule policies by each algorithm in the given time as shown in Table I.

TABLE I. Performance of MPH-BMH in 300 seconds

Algorithms	DEFCON 1.20 GB	DARPA 503 MB	CRAWDAD 137 MB
BMH	184	179	174
QS	186	180	191
MPH-BMH	365	335	373

The experiments show that MPH-BMH can matched large number of packets with Snort rule policies, and consequently, it has less running time whatever the size of incoming packets that the IDS process them as shown in Fig. 4.

MPH-BMH algorithm has a faster matching mechanism as it compares multi pattern of rule policies against the incoming packets at the same time. This can make its performance faster in high speed network where the IDS indeed needs very fast string matching algorithms.

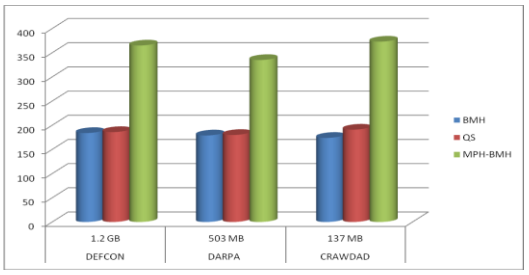


Figure 4. Performance of MPH-BMH against other algorithms.

V. Conclusion

As the security is considered as important issue in the modern networks, the need for IDS becomes imperative to secure the networks. In this research, we reformulated the Boyer-Moore Horspool algorithm by adding new feature to make it working as multi pattern string matching algorithm based on the hashing function. This can boost the performance of IDS in which MPH-BMH overcomes the weakness of the other string matching algorithms. The experiments show that the performance of MPH-BMH is around two times faster than BMH and QS algorithms.

Acknowledgment

This Research reported here is pursued under the Exploratory Research Grant Scheme (ERGS) by Ministry of Higher Education (Malaysia) and Universiti Sains Malaysia for ”Parallel Exact String Matching Algorithm For Intrusion Detection System” [203/PKOMP/6730074].

References

[1] A. A. Ghorbani, W. Lu, and M. Tavallaee, “Network intrusion detection and prevention: concepts and technique,” in *Advances in Information Security*, US Springer Publisher, 2010, vol. 47, pp. 161-183.

[2] M. Ali Aydın, A. Halim Zaim, and K. G. Ceylan, “A hybrid intrusion detection system design for computer network security,” *Computers & Electrical Engineering*, vol. 35, no. 3, pp. 517-526, 2009.

[3] M. Tenase. (2003). The great ids debate: Signature analysis versus protocol analysis. *Infocus*. [Online]. Available: http://www.securityfocus.com/infocus/1663

[4] J. Liu and F-M. Dong, “A dynamic adaptive load balance algorithm in parallel intrusion detection system,” in *IEEE International Symposium on Computer Science and Computational Technology*, Dec 2008, pp. 180-183.

[5] W. Yang, B. X. Fang, B. Liu, and H. L. Zhang, “Intrusion detection system for high-speed network,” *Computer Communications*, vol. 27, no. 13, pp. 1288-1294, 2004.

[6] Y. Chen, Y. Li, X. Q. Cheng, and L. Guo, “Survey and taxonomy of feature selection algorithms in intrusion detection system,” in *Information Security and Cryptology*. Berlin / Heidelberg Springer Publisher, 2006, pp. 153-167.

[7] S. Antonatos, K. G. Anagnostakis, M. Polychronakis, and E. P. Markatos, “Performance analysis of content matching intrusion detection systems,” in *Proc 4th IEEE/IPSJ Symposium on Applications and the Internet*, 2004, pp. 208-215.

[8] A. A. Hasan and N. A. Rashid, “Hash-boyer-moore-horspool string matching algorithm for intrusion detection system,” in *Proc. International Conference on Computer Networks and Communication Systems*, April 2012, vol. 35, pp. 20-25.

[9] P. Lin, Y. Lin, T. Lee and Y. Lai, “Using string matching for deep packet inspection,” *IEEE Computer Society*, vol. 41. no. 4, pp. 23-28, April 2008.

[10] B. Kun, G. Nai-jie, T. Kun, L. X. Hu, and L. Gang, “A practical distributed string matching algorithm architecture and implementation,” in *Proc. World Academy of Science, Engineering and Technology*, Dec 2005, pp. 196-200.

[11] A. V. Aho and M. J. Corasick, “Efficient string matching: an aid to bibliographic search,” *Communications of the ACM*, vol. 18, no. 6, pp. 333-340, 1975.

[12] R. S. Boyer, and J. S. Moore, “A fast string searching algorithm,” *Communications of the Association for Computing Machinery*, vol. 20, no. 10, pp. 762-772, 1970.

[13] R. N. Horspool, “Practical fast searching in strings,” *Software-Practice and Experience*, vol. 10, no. 6, pp. 501-506, 1980.

[14] Charras and T. Lecroq, *Handbook of Exact String Matching Algorithms*, King’s College Publications. France, 2004.

[15] S. Wu, and U. Manber, “A fast algorithm for multi-pattern searching,” Tech. Rep. TR94-17. University of Arizona, 1994.

[16] K. G. Anagnostakis, S. Antonatos, E. P. Markatos, and M. Polchronakis, “E2xb: A domain-specific string matching algorithm for intrusion detection,” in *Proc 18th IFIP International Information Security Conference*, 2003, pp. 217-228.

[17] R. T. Liu, N. F. Huang, C. N. Kao, C. H. Chen, and C. C. Chou, “A fast pattern-match engine for network processor-based network intrusion detection system,” in *Proc International Conference on Information Technology: Coding and Computing*, April 2004, pp. 97–101.

[18] S. Antonatos, M. Polychronakis, P. Akritidis, K. G. Anagnostakis, and E. P. Markatos, “Piranha: Fast and Memory-Efficient Pattern Matching for Intrusion Detection,” in *Proc 20th IFIP International Information Security Conference*, 2005.

[19] J. V. Lunteren, “High-performance pattern-matching for intrusion detection,” in *Proc 25th IEEE International Conference on Computer Communications (INFOCOM)*, April 2006, pp. 1-13.

[20] R. M. Karp and M. O. Rabin, “Efficient randomized pattern-matching algorithms,” *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249-260, 1987.

[1] Defcon. (2009). [Online].Available: http://cctf.shmoo.com/data/

[21] MIT Lincoln Lab. DARPA. (1999). [Online]. Available: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/

[22] Crawdad. Trace of Network Activity. (2006). [Online]. Available: http://uk.crawdad.org/meta.php?name=microsoft/osdi2006#N101A8

[23] Sourcefire. Snort. [Online]. Available: http://www.snort.org.

[24] D. M.Sunday, “A very fast substring search algorithm,” *Communications of the Association for Computing Machinery*, vol. 33, pp. 132–142, 1990.

Awsan A. H. Abdullah was born in Aden, Yemen. He got his bachelor from University of Aden, Yemen in 2003. His master degree was got from Universiti Sains Malaysia in 2010 in Penang, Malaysia. He is currently Ph.D candidate in department of Parallel and Distributed Processing, School of Computer Sciences Universiti Sains Malaysia.

He work as graduate assistant in School of Computer Sciences Universiti Sains Malaysia from January 2011 to September 2012 and from October 2012 he works under the Exploratory Research Grant Scheme (ERGS) by Ministry of Higher Education (Malaysia) and Universiti Sains Malaysia.

Mr. Abdullah is interested in network security, parallel processing and string matching algorithms developments

Associate Prof. Dr. Nur’Aini Abdul Rashid was born in Malaysia. She got her bachelor from Mississippi State, U.S.A. she got her MSc and PhD from Universiti Sains Malaysia.

She was Deputy Dean of Academic & Students Development, School of Computer Sciences, Universiti Sains Malaysia. She is professional in Parallel and Distributed Processing, Genomic Information Processing, and String and Pattern Matching. Her interest now is applying the parallel techniques to genomic information retrieval and analysis. The method that she is investigating is on shared memory platforms. The concentration is on parallel sequence comparison and parallel graph-based clustering algorithms.

Dr. Abdul Rashid was member in Association of Computing Machinery (ACM) from 2002 to 2010, International Society of Computational Biology from 2008 to 2009 and MASBIC, Malaysian Bioinformatics from 2007 to 2010.

Muhannad A. Abu-Hashem was born in Jordan. He got his Bachelor degree in Computer Information System (CIS) from Philadelphia University, Jordan in 2003, and he got his M.Sc. in computer science from Universiti Sains Malaysia in 2008. He is currently a Ph.D candidate in department of Parallel and Distributed Processing, School of Computer Sciences Universiti Sains Malaysia.

He works under the University Research Grant Scheme (RU) by Universiti Sains Malaysia. He specialized in parallel and Bioinformatics.

Atheer Akram AbdulRazzaq was born in Baghdad, Iraq. He got his bachelor from Al Mustansiriya University, Iraq in 2006. He got his master degree from Universiti Sains Malaysia in 2009. He is currently Ph.D candidate in department of Parallel and Distributed Processing, School of Computer Sciences, Universiti Sains Malaysia. He specialized in exact string matching algorithms, parallel, and network security.