2012 International Conference on Computer Networks and Communication Systems (CNCS 2012)
IPCSIT vol.35(2012) © (2012) IACSIT Press, Singapore

Hash - Boyer-Moore - Horspool String Matching Algorithm for Intrusion Detection System

Awsan Abdulrahman Hasan [1] and Nur Aini Abdul Rashid [2]

[1] School of Computer Sciences, University Sciences Malaysia, Penang Malaysia

[2] School of Computer Sciences, University Sciences Malaysia, Penang Malaysia

Abstract. String matching algorithms have become important in many applications. The Intrusion Detection System (IDS) is one of them. An IDS is created and becomes an important part of modern network structure to detect the malicious activities. The detection engine of IDS depends on string matching algorithms to perform the searching for these malicious activities, but this searching phase consumed around 70% of IDS processing time. Nowadays, as the networks increases rapidly, the searching time will increase due to a huge number of incoming packets together with the increasing of signatures rule policy. There is a real need to reduce the IDS processing time by developing a method that can work faster. In this paper, we suggest enhancing the Boyer-Moore Horspool algorithm by adding hashing function called HBMH to reduce the characters comparison time. The simple experiment result proves that the enhanced algorithm performs faster speed in lowest time.

Keywords: String Matching Algorithms, IDS, Processing Time, Hash Function, Pattern, Rules Policy.

1. Introduction

Nowadays the network has expanded to very large scales and the amount of the network traffic and the data exchange between the networks has increased tremendously. On the other hand, the world today depends on Internet to do the whole work. Therefore, very large numbers of computers around the world stay connected to Internet all day long. These computers store very important data such as emails, documents, pictures and videos. Some of these computer may install uncertified programs like Trojan horse, that send sensitive information belong to users such as user name, password and credit card number. Because of the growth of the Internet users and the user data, the computers have become a favourite target for complexity of attacks that requiring more complex analysis to detect it. As security is considered important in the network, The Intrusion Detection System (IDS) becomes an important part of any modern network for guaranteeing the security issue of information systems [1, 2].

There are two approaches of IDS; anomaly detection (also called behaviour based) or misuse detection (also called signature based). The anomaly based is searching for any abnormal behaviour occurs in the network. In this approach, the IDS uses some statistical methods to monitor and analysis the network behaviours. In signature based approach, the IDS compares and matches the content of any incoming packets with a list of well known signatures attack stored in a database; this list forms what is called a rule policy. When the IDS found a matching then an alarm is started to inform the system administrator [1, 3, 4]. By comparing the two approaches for IDS, it is clear that the signature based detection approach is more stable and reliable, but its drawback the limitation of the detecting new attacks. Therefore, it needs to update the rule policy frequently. On the other hand, the anomaly based detection has the ability to discover the new attacks on the networks, but its drawback is giving a high rate of false alarm and consuming more resources. Therefore, most of the networks are running IDSs signature based detection approach because of its accuracy and low recourse consuming [3]. In this paper, we concentrate on IDS based signature. Snort [6] is an example of an open source code IDS based signature which has a very large number of growing rules policies [6].

One of the important processes in IDS is Deep Packet Inspection (DPI) in which it inspects the contents of each packet [7]. After that, the IDS can determine whether these packets are infected with any malicious activities or not. However, this process performs by utilizing of string matching algorithms [8]. String matching algorithms are widely used in computer science and applied to many fields such as bioinformatics, web search engine, pattern recognition, and IDS [9]. In IDS, the string matching algorithms considered the heart of IDS, but unfortunately it needs a long time to perform the searching and matching the packets content and rule policies. However, this process slows down and consumes 70% of IDS processing time [8]. This is because the IDS receives a very large amount of incoming packets. Moreover, the rule policies also grow every time rapidly. Through all these issues, it is difficult for IDS string matching algorithm to differentiate between the normal packets and abnormal packets without dropping the packets or overhead the system performance [10,11,14]. There is a need to accelerate the string matching performance by developing a method enhance and reduce the processing time. In this paper, we added a hash function for Boyer-Moore Horspool algorithm [12] to enhance and reduce the number of character comparison time.

2. Factors Influence on IDS

There are many factors can affect on the IDS performance, and the most important factor of them is the string matching algorithm itself [8]. This is because most of the string matching algorithms are not designed directly to work with IDS, but instead of this, they are designed to work with other applications such searching on dictionaries [13]. Another important factor is the high speed of modern network, because the detection engine is searching and comparing the packets against signatures rule policies. But due to the large amount of incoming packets the IDS will lose control of all these packets and consequently, will drop a number of packets. As result, these drooped packets may contain some malicious pattern that can pass successfully [10]. One last thing, the rule policies are growing rapidly. Snort rule policies are increasing and reaching more than 8000 rules [6]. However, all these factors are affected on IDS performance and increased the processing time.

3. Related Work

Welcome Nowadays the researchers on IDS field are getting more interested due to ability of IDS to monitor the network traffic and detect the abnormal activities. There are many previous works in IDS; some of them are based on Hardware solutions, while others are based on software solutions. However, this research is concentrates only on software based solutions in which it enhances the performance of string matching as well as decreases the processing time.

IDS is utilizing of some familiar string matching algorithms such as Aho-Corassick algorithm (AC) which is multiple string matching algorithm based on the finite state machine [15], Boyer-Moore algorithm (BM) which is one of the most famous exact string matching algorithms based on a single pattern matching [16], Boyer-Moore Horspool Algorithm (BMH) which is a modified version of Boyer-Moore algorithm [12]. These two sections will highlight on BMH algorithm besides other researchers' algorithms enhancement.

3.1. Boyer-Moore Horspool Algorithm (BMH)

BMH is a modified version of Boyer-Moore algorithm with a little different change. Unlike, Boyer-Moore algorithm the BMH algorithm uses only one table (bad character shift) whereas, the Boyer-Moore algorithm uses two tables: (bad character shift) and (good suffix shift). BMH searches for the pattern from left to right and the shift value in it is depending on the pattern size determined in the bad character shift table. So the longer patterns size, the longer shift value. This makes BMH faster in its performance in most of the situations that applied on it. It is simple to implement and has a low space complexity and best average case performance. The average of comparisons performed for one text character is between $1/\sigma$ and $2/ (\sigma +1)$. Therefore, it can be implemented in any cases that need the exact string matching algorithm even if the pattern size is small or large [24, 25]. IDS is utilized of this algorithm before, but it still needed little enhancement and improvement in its characters comparison speed. Because of BMH algorithm advantages we choose it through all others algorithms to enhance its performance.

3.2. Other Researchers Development

There are many other researchers are worked to enhance the string matching algorithms to be fully adopted to work with IDS. [17] Presented Wu-Manber algorithm (WM), it is based on the Boyer-Moore algorithm, but it can support multiple pattern matching in one step. However, the problem with this algorithm was the inability to work with a small set of patterns. [18] Designed an algorithm called Set-wise Boyer-Moore-Horspool (SWBMH), which is a modified version of Boyer-Moore to match different rule policies simultaneously. The algorithm was the fastest of both of Aho-Corasick and Boyer-Moore, only for pattern sets with medium size. [19] Designs a multiple string-matching technique, is called E2xB. It offers a higher efficiency and capacity of IDS. Unfortunately, this algorithm works well only for small rule policies as well as packets. [20] Presented Piranha which is an algorithm was designed to work with IDS. The main idea is that when the rarest substring in a pattern mismatch, then the whole substring will mismatch with the text, but it cannot work with a rule set smaller than 4 bytes. [8] Presented a study of IDS workloads problems, they showed that IDS affected by string matching algorithms; rule sets size, traffic characteristics and processor architecture.

[21] Modified AC algorithm by using bitmap node compression and path compression (AC-Bitmap and AC-Path) to increase the storage space and enhance the worst case behaviour. [22] introduced a new idea to enhance the WM algorithm; it is called Dual-Algorithm. This idea is come because there are multiple bytes patterns in a Snort rule sets such as 1 byte, 2 bytes and 3 bytes or more. This 1 byte pattern can slowdown the performance of WM matching phase. The Algorithm is aimed to decrease the dependency on multi byte hash table of WM algorithm and it is only implemented when there is a multi byte pattern (3 bytes or more) and one byte pattern alone. [23] Proposed a quick string matching algorithm (RQS) based on BM algorithm. It consists of two features which develop a bad character. Therefore, it can perform a large shift value. Besides, it improves a good suffix to memorize the matching characters when the algorithm required it. RQS got better worst case performance in comparison to BM algorithm. On the other hand, RQS can work efficiently with longer pattern size without any difficulties. However, our method is added a hash function for Boyer-Moore Horspool algorithm to decrease the number of character comparison.

4. Proposed Method

In this section, we explain the method of enhancing BMH algorithm by adding the hash function to work beside the shift table of the BMH algorithm; the method is called Hash Boyer-Moore Horspool (HBMH). The hash function converts the string into number and it can be calculated in the same manner of Karp-Rabin algorithm [5]. The main benefit of using the hash function is to reduce the number of character comparisons performs by BMH algorithm in each attempt. Thus, it reduces the required comparison time. The example below is taken from [25] which can explain the main idea of using the hash function of this algorithm.

Suppose we have the pattern (GCAGAGAG) and the text (GCATCGCAGAGAGTATACAGT). The implementation of BMH algorithm starts by creating the bad characters table *bmBc* from the pattern. However, this table is used to determine the value of shifting after each comparison performed between the pattern and the text [25]. After that, the algorithm is attempting to find a matching between the pattern and text. If there is no matching between the pattern and the text then the BMH will shift to the next comparison which is 1 (A= 1) by using the value in the *bmBc* table as shown in Fig. 1.

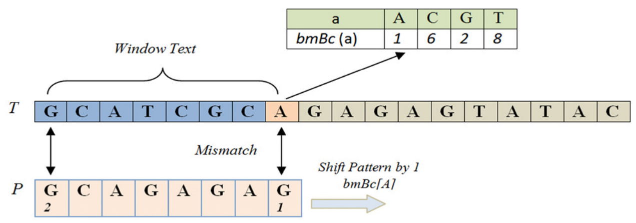

Fig. 1: BMH Matching Phase between the Pattern and Text

The algorithm will compare the character G of the pattern against the character A of the text. If there is no matching between the first characters (G and A), then the algorithm doesn't check the rest of characters and rather it performs shifting by 1 value using *bmBc* table as shown in Fig. 1. The algorithm at this step performs only 1 character comparison. However, the main problem is that when most or all characters of the patterns are matched with characters of the text. In this case, the algorithm performs a large number of character comparisons and consequently consumes more time especially, when the size of the pattern is very large as shown in Fig. 2a. Now from this point, it is clearly notable that there is a time wasted in every comparison between the characters of pattern (GCAGAGAG) and the characters of text (GCAGAGAG) being compared.

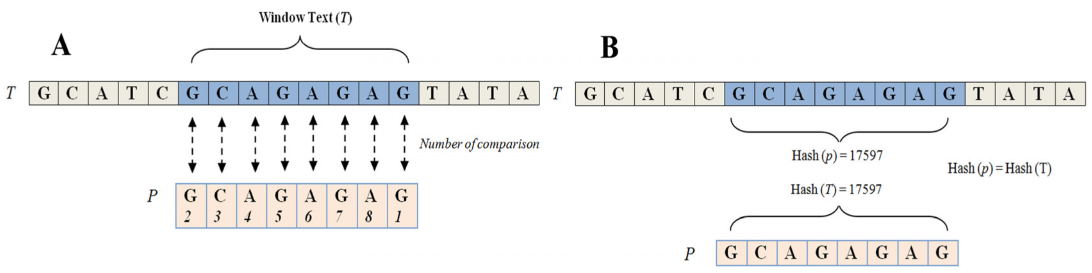

Fig. 2: Character Comparison of BMH. Fig. 2b: Applying Hash Function on BMH

Since IDS has large number of incoming packets as well as rule policies. We need to reduce the comparison time as much as possible. Therefore, we add the hash function in order to take a place in BMH algorithm. The hash function method HBMH will convert all characters in the text and pattern into numbers and it will only compares the numbers instead of characters. Thus, if the numbers being compared are equals, then the characters are also equals as shown in the Fig. 2b. By doing so, we can reduce the number of character comparison into 1 step only, whatever the length of pattern that being matched.

As an example, the hash function for the pattern (GCAGAGAG) is 17597 and for the text (GCAGAGAG) is also 17597. Because the hash value of the pattern and the text are equals. Then, the characters of the pattern are matched with the characters of the text. From the comparison between the original BMH performance and the hash function HBMH, we can notice that the BMH performed 8 character comparisons, but by using HBMH the number of character comparison is reduced to 1 only as shown in the Fig. 2b, and this is what the IDS required.

5. Implementation of HBMH

Every string matching algorithms have two main phases; pre-processing phase and searching phase [25] and HBMH is too. In pre-processing phase, HBMH has two stages. The first one is creating *bmBc* table from the pattern. The second one is calculating the hash function of the pattern as well as text window. However, this is unlike BMH, which only creates *bmBc* table in its pre-processing phase.

In searching phase HBMH performs the searching phase by implementing the hash function which compares the hash value of the pattern against the hash value of the text. If the hash value is not equal then, the algorithm will shift to next comparison by using the value of *bmBc* table, and the hash function will recalculate for the new text window only.

By referring to Fig. 1, after the comparison between the pattern (GCAGAGAG) and window text (GCATCGCA) is finished, HBMH will shift the pattern by 1 position that is because the value of A in *bmBc* table is 1. Therefore, Shift P by 1 location (*bmBc*[A]). After that, HBMH will calculate the hash value for the new text window (CATCGCAG) and then, the comparison between the hash value of pattern and the new text window will start as shown in Fig.3 and the operation will continue until the search complete anyone at the end of the text.

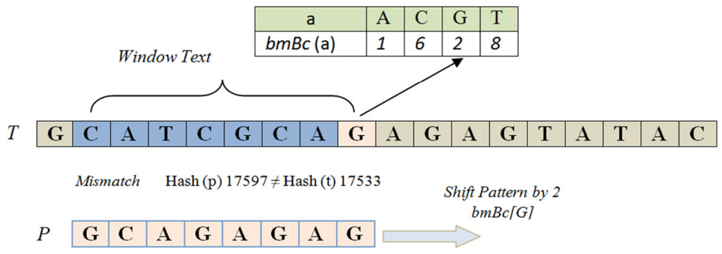

Fig. 3: Searching Phase of HBMH

6. Results

By implementing the example in [25] for both BMH and HBMH algorithms, we can notice that the performance of HBMH outperformed the performance of BMH. This is because, the hash function is really reduced the number of character comparison form 17 comparisons in BMH to only 8 comparisons in HBMH as shown in Fig. 4. This proves that HBMH is faster in its performance and can be very useful in network security applications such IDS.

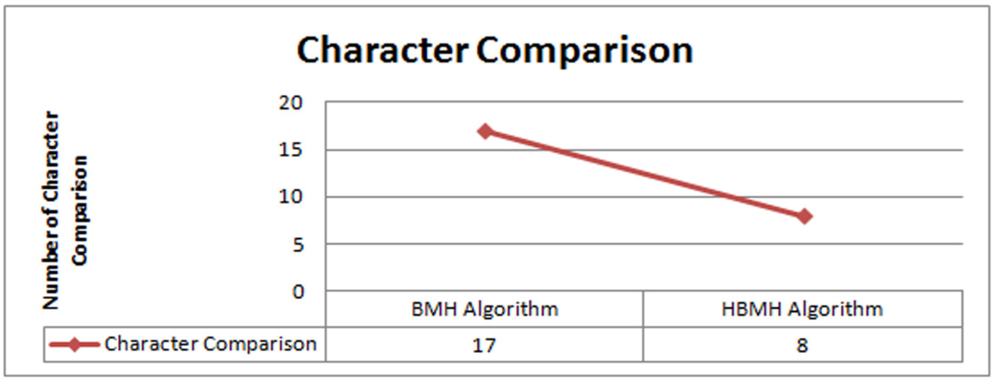

Fig. 4: Performance of HBMH over BMH

7. Conclusion

The Internet users are wildly spread besides; the intrusion activities are also increased. Therefore, there is a need for system to detect the malicious packets. IDS is created and spread in the networks to detect these malicious activities. However, the IDS depends on string matching algorithms to perform its detection, but these string matching algorithms have many weaknesses such as the large time consumed in IDS detection engine. In this paper, we introduced hash function HBMH to enhance the BMH algorithm whereas; BMH spends a long time in character comparison in each attempt. The results were taken from simple example which showed that the HBMH has a less character comparisons. Therefore, it considers faster than the original BMH algorithm. For IDS field we think that HBMH is the best choice for IDS. This is because, IDS receives a very large number of packets, it needs faster method to compare the packets especially, packets have large length. In future work, we intend to apply HBMH in a real network environment in order to, test its ability of handle a very large number of network packets.

8. References

[1] S. Beg, U. Naru, M. Ashraf, S. Mohsin. Feasibility of Intrusion Detection System with High Performance Computing: A Survey. *International Journal for Advances in Computer Science*. 2010, 1(1): 26-35.

[2] C. V. Zhou, C. Leckie, and S. Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *Computers &amp. Security*. 2010, 29 (1): 124-140.

[3] M. Ali Aydın, A. Halim Zaim, and K. Gökhan Ceylan. A Hybrid Intrusion Detection System Design for Computer Network Security. *Computers &amp. Electrical Engineering*. 2009, 35 (3): 517-526.

[4] A. Singhal. Data Warehousing and Data Mining Techniques for Cyber Security. *Advances in Information Security*. 2007, 31, pp. 69-82.

[5] R. M. Karp, and M. O. Rabin. Efficient Randomized Pattern-Matching Algorithms. *IBM Journal of Research and Development*. 1987, 31 (2): 249-260.

[6] Snort on http://www.snort.org

[7] P. Lin, Y. Lin, Y. Lai, and L. Lee. Using String Matching for Deep Packet Inspection. In: *IEEE Computer Society*. 2009, pp. 23-28.

[8] S. Antonatos, K.G. Anagnostakis, M. Polychronakis, and E.P. Markatos. Performance Analysis of Content Matching Intrusion Detection Systems. *In: Proc of the 4th IEEE/IPSJ Symposium on Applications and the Internet (SAINT)*.2004.

[9] B. Kun, G. Nai-jie, T. Kun, L. Xiao-Hu, and L. Gang. A Practical Distributed String Matching Algorithm Architecture and Implementation. In: *Proc of world academy of science, engineering and technology*. 2005, pp. 1307-6884.

[10] W. Yang, B.-X. Fang, B. Liu, and H.-L. Zhang. Intrusion Detection System for High-Speed Network. *Computer Communications*. 2004, 27 (13): 1288-1294.

[11] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo. Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. *Information Security and Cryptology. Springer Publisher Berlin / Heidelberg*. 2006, pp. 153-167.

[12] R. N. Horspool. Practical Fast Searching In Strings. *Software-Practice and Experience*. 1980, 10 (6): 501-506.

[13] J. V. Lunteren. High-Performance Pattern-Matching For Intrusion Detection. In: *Proc of 25th IEEE International Conference on Computer Communications*. 2006, pp. 1409-142.

[14] H. Lu, K. Zheng, B. Liu, and C. Sun. A Robust Approach for Matching Mixed Case-sensitive and Case-insensitive Patterns. In: *Proc of Third International Conference on Networking and Services*. 2007, pp. 72-72.

[15] A. V. Aho, and M. J. Corasick. Efficient String Matching: An Aid to Bibliographic Search. *Communications of the ACM*. 1975, 18 (6): 333-340.

[16] R. S Boyer, and J. S. Moore. A fast string searching algorithm. *Communications of the Association for Computing Machinery (ACM). 1970,* 20 (10): 762-772.

[17] S. Wu, and U. Manber. A Fast Algorithm for Multi-Pattern Searching. *Tech. Rep. TR94-17. University of Arizona*, 1994.

[18] M. Fisk, and G. Varghese. An Analysis Of Fast String Matching Applied To Content-Based Forwarding And Intrusion Detection. *Tech. Rep. CS2001-0670 (updated version). University of California - San Diego*, 2002.

[19] K. G. Anagnostakis, S. Antonatos, E. P. Markatos, and M. Polchronakis. E2xb: A Domain-Specific String Matching Algorithm for Intrusion Detection. In: *Proc of 18th IFIP International Information Security Conference (SEC).* 2003, pp. 217-228.

[20] 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 of 20th IFIP International Information Security Conference* (SEC), 2005.

[21] N. Tuck, T. Sherwood, B. Calder, and G. Varghese. Deterministic Memory Efficient String Matching Algorithms for Intrusion Detection. In: *Proc of the IEEE Infocom Conference*. 2004, pp. 333–340.

[22] P.S. Wheeler. Techniques for Improving The Performance Of Signature Based Intrusion Detection Systems. *Master's thesis. University of California Davis*, 2006.

[23] J. Yu, and Y. Xue, Robust Quick String Matching Algorithm for Network Security. *International Journal of Computer Science and Network Security (IJCSNS)*. 2006, 6 (7B): 180-184.

[24] C. Lovis, and R. H. Baud. Fast Exact String Pattern-matching Algorithms Adapted to the Characteristics of the Medical Language. *Journal of the American Medical Informatics Association. 2000,* 7 (4): 378-391.

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