www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 6 June 2015, Page No. 12385-12389

Indepth Packet Inspection Using Modified Enhanced Multipattern Matching Algorithm

Hina Naz
[M.tech C.S.E]
Department of computer science and Technology

Abstract. One of the most important requirements in intrusion detection systems (IDSs) is a good pattern matching algorithm. EHMA by Tzu-Fang et al. is an efficient and cost-effective pattern detection algorithm for packet inspection. A few key assumptions in their work were given without enough justification. In this paper, we have tried to verify some key assumptions of this algorithm by testing it on a cluster with real network data. We also introduce some methods to improve the algorithm by choosing the input parameters appropriately. The results show that some assumptions may not correct in some cases, and by applying some changes to the existing algorithm, we can make the performance of the matching process much better.

Introduction

During the last few decades, when more and more network threads are emerging, intrusion detection systems (IDSs) have become a well-known tool in detecting malicious or unwanted traffic. One of the simplest and most common techniques used in IDSs is based on matching a set of signatures against network payload packets. In this kind of system, string matching is always the part that dominates the execution time of the whole system. For this reason, finding an efficient string matching algorithm is a crucial issue when implementing an IDS.
One of the first and most well-known string matching algorithms is proposed by Boyer and Moore [1]. The Boyer-Moore algorithm works by using *bad character heuristics* so it can skip some of the input characters without comparing all of them. Boyer-Moore is proved to have the best average-case performance. Aho-Corasick [2] is one of the earliest multi-pattern matching algorithms, which is able to match string in linear worst-case time in the size of the input. Wu-Manber [3] is also a multi-pattern matching algorithm that based on the *bad character heuristics* similar to that of Boyer-Moore algorithm. Wu-Manber performs a hash on the two-character prefix of the current pattern to index into a group of patterns.
Tzu-Fang et al. [4] argue in their paper that the current matching algorithms are impractical for packet inspection in real time system. The reason is that the performance of processing packets depends not only on the computation time, but also on the number of required external memory references. Some network equipment has a small embedded memory. So we cannot keep all the signature patterns in the internal memory. External memory is required now but this will lead to inefficiency when we need to access to external memory to read the patterns. To deal with this problem, they propose a new algorithm named EHMA.
EHMA is proved to significantly reduce the average number of external memory accesses and improve the matching performance. However, we realized that there are some assumptions in this algorithm that may not be true in some cases. We try to verify these assumptions by testing them on a cluster with real network data. We also suggest some methods to improve the performance of the algorithm.
The rest of the paper is organized as follow. In section 2 we introduce EHMA algorithm and its underlying assumptions. Section 3 describes our method to verify and improve the algorithm. Section 4 includes the experiment results and some discussions. Finally, section 5 summarizes our results and presents the plan for future works.

2 EHMA Algorithm and Underlying Assumptions

EHMA is a two-tier and cluster-wise matching algorithm that reduces the number of external memory accesses, and uses a skipping scan strategy to speed up the scanning process.
When the internal memory of the network equipment is of limited size, the pattern set need to be stored in the external memory. One of the key solutions to speed up the scanning stage is to reduce the number of external memory accesses. EHMA does this by keeping two small index tables in the memory. These two tables will work as a filter that divides the scanning process into two layers. EHMA contains two main phases. We will take a closer look at how these two phases work and some

underlying assumptions that will be verified in the next section.

2.1 Notations

Let *m* be the minimum length of all patterns in the pattern set and *W*, a chosen parameter, be the size of the sampling window. We define the sampling window to be the last *W* characters of the first *m* characters in a pattern (Fig. 1). *F* is a *common gram set* which will be defined later. A string can also be denoted as a set of *B-grams*, where gram is a group of B consecutive characters in a string. For example, a string "abcd" can be represented in 1-gram form as {"a", "b", "c", "d"} or 2-gram form as {"ab", "bc", "cd"}.

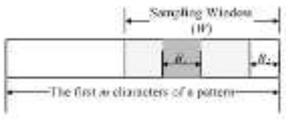

Fig. . The sampling window

2.2 Offline Phase (Table Construction Phase)

From the pattern set, EHMA constructs two index tables H_1 and H_2 that will be used in the matching phase. Fig. 2 is an image of these two tables for the pattern set P = {"teacher", "architect", "firefighter", "farmer", "actress"}. H_1 table works as the first filter. H_1 includes 256 entries, where each entry key is a gram from the character set. Some entries have a link to H_2 table. The set of all entries that have a link to H_2 (in this case {e, h}) is called a *common gram set* F. Have a closer look at this subset {e, h}, we can see that the gram *h* is a representative for the patterns {"teac*h*er", "arc*h*itecture"} while the gram *e* is a representative for the patterns {"fir*e*fighter", "farm*e*r", "actr*e*ss"}.

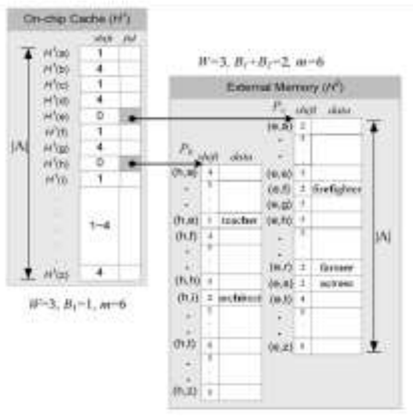

Fig. . The architecture of the hierarchical hash tables H_1 and H_2

EHMA algorithm chooses the *common gram set* ({e, h}) based on the following idea. For each pattern, we will choose a fixed part of it, called the sampling window (the last *W* characters of first m characters in a pattern, Fig. 1). From this part, we construct a B-gram set and choose one gram in that set to be a good representative for that pattern. According to Tzu-Fang et al., a gram is considered to be good when it appears in all patterns most frequently. Choosing the most frequent grams will help in reducing the size of *common gram set*, so we can skip more characters when scanning. H_1 is small enough to be kept in internal memory. On matching phase, each character read from the network packet will be used as a key to fetch an entry in H_1.

Only the entries that belong to F will continue to activate the compare process in H_2. Usually the *common gram set* size |F| is much smaller than the character set size. This will help in reducing the number of accesses to H_2 table, mean accesses to external memory.

H_2 table contains all the patterns and distributes this patterns to clusters based on F. Both H_1 and H_2 table keep a shift list, which helps to safely skip some characters of the input packets without any further processing.

2.3 Online Phase (Matching Phase)

After running offline phase, H_1 and H_2 are built and ready for matching phase. Firstly each gram will be read from network packets and be used as a key to search for an entry in H_1 table. Then the *shift value* of this entry will be checked. If it is 0, it means that this gram belongs to the common gram set; the entry in the H_2 table will be fetched. H_2 entry includes the patterns that related to this gram and we will do matching with these patterns. If shift value is large than 0, it also indicates the number of characters that we can skip without processing. We repeat this step for the next scanning grams.

2.4 Some Assumptions in EHMA Algorithm

In this part, we present three assumptions in EHMA algorithm that may not be verified enough on their paper.

- *Assumption 1 (A1) :*

When extracting the *common gram set* from the pattern set, EHMA algorithm try to find the smallest *common gram set* because according to the authors, a small *common gram set* will help in reducing the number of external memory accesses. But we suspect that in some case, when the smallest common gram set contains the grams that also appear many times in normal traffic, the smallest one may not give the best performance. We will verify this in the next section.

- *Assumption 2 (A2) :*

In EHMA, the sampling window has a fixed position and is the same between patterns. We tried to modify the algorithm to make the sampling window position flexible and can be different from each other.

- *Assumption 3 (A3) :*

In Tzu-Fang et al.'s experiment with the algorithm, some parameters are chosen intuitively. For example, the window size W is fixed to 3 and the minimum pattern length m is 6. We try to change these values and check if these values are good or not.

3 Method to Verify and Improve EHMA Algorithm

This section presents our method to verify above three assumptions as well as improve the algorithm.

3.1 Assumption A1

In EHMA, from the pattern set, they choose the smallest *common gram set* to construct the index table H_1 and H_2. In this paper, we choose another approach. The method we have used is presented in Fig. 3. From the pattern set, instead of generate the smallest common gram set, we try to generate all possible common gram sets, construct index table H_1, H_2 for these sets. Each pair (H_1, H_2) is considered as a version of the detection engine. Next, we construct a simulation to run all the versions with the same network traffic data at the same time. Based on the performance of each version on simulation, we can choose the best gram set.

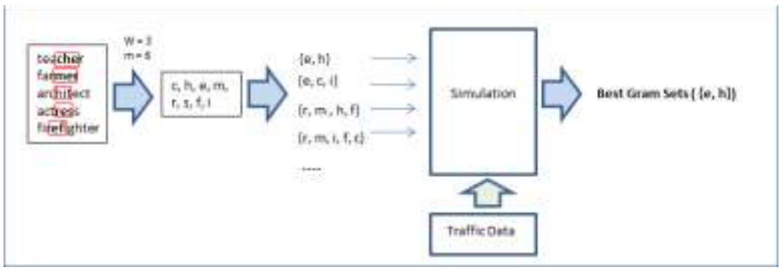

Fig. . Method to verify the assumption about *common gram set* size

3.2 Assumption A2

Fig. 4 presents our method to modify the EHMA algorithm. Fig. 4a) is the way EHMA choose the sampling window. The windows size is 3, and the position of sampling window is the same and fixed to the last W characters of first m characters of each pattern. In Fig. 4b), we try to modify EHMA by letting the sampling windows at different positions. The purpose of making the window position flexible is to find a smaller, then maybe a better common gram set. The algorithm to find the window position for each pattern is as follow. First, we calculate the frequency for every gram appears in all patterns. Then for each pattern, we will choose the sampling window at the position that the sum of the frequency of all grams in the window is maximized. By this way, after the window position is chosen for each pattern, we expect that the *common gram set* generated will be good

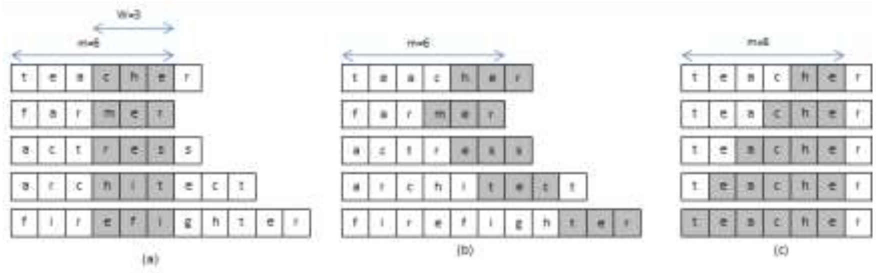

Fig. . Method to improve the scanning performance by changing input parameters. (a) In EHMA, the size and position of sampling window is fixed. (b) We modified EHMA algorithm by making the position of sampling window flexible. (c) Try to test with window size W = 2, 3, 4, 5 or more

3.3 Assumption A3

In Fig. 4c), we also do another experiment by changing the window size *W* to different values 2, 3, 4 or more to choose the best value.

4 Results and Discussion

This section presents the results of our method to verity some assumptions in EHMA. We also discuss the reasons and give the explanations for the experiment results.

4.1 Datasets and Measurement

We tested the algorithm with pattern set from Snort in 2012 [5]. The network traffic was extracted from the Capture-the-Flag contest held at DEFCON9 [6]. When matching the pattern set with the network traffic, we used a simulator to count the number of character comparisons and external memory accesses. The algorithm was evaluated based on these two values.

4.2 Results and Discussion

Fig. 5 compares performance of the algorithm among different common gram sets. The left chart presents the number of character comparisons in corresponding with the common gram size 13, 14, 15 or more while the right chart presents the number of external memory accesses. We can see that these values are the smallest – mean the matching performance is best – when the size of the common gram is 18, not the smallest. So the assumption *A1* may not correct in this case. This can be explained that the fact that the smallest *common gram set* (size = 13 in this experiment) may contain some characters that appear very frequently in the normal traffic. This leads to many unnecessary accesses to the external memory.

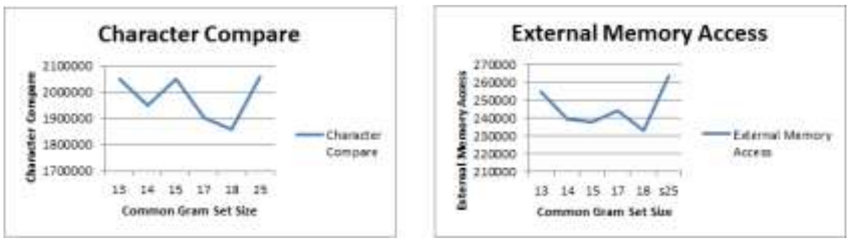

Fig. . Compare performance of the algorithm among different common gram set

About assumption *A2*, in Fig. 6 we compare the performance of EHMA with modified EHMA that allows flexible sampling window position. It shows that choosing the *common gram set* from the windows start at different positions in the patterns can improve the performance of the scanning process. And the improvement also increases with the size of the pattern set.

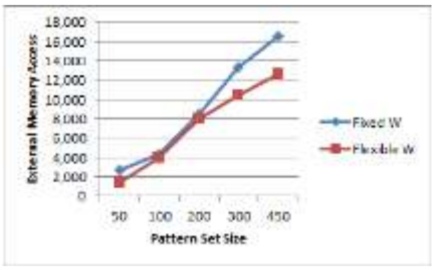

Fig. . Compare between EHMA algorithm and modified EHMA algorithm which change Window Position

Fig. 7 shows the result for assumption *A3* when we test the window size *W* with different values 2, 3, 4 or more. In this case as well as in almost other experiments, value *W*=3 usually gives the best result. *W*=3 is also the value that was chosen by EHMA's authors. This can be explained like this. If *W* is too small (*W*=2), the number of the common gram chosen is too large, this is not good. On the other hand, if *W* increases, the number of shift value in H_1 and H_2 is significantly decreased (from Safety Shift Calculation algorithm [4]). This affects the scanning process. Experiments show that *W*=3 is the best value in most cases. In the future, we are going to do more experiments relating to changing W together with other parameters like gram size *B1* … to get a more certain conclusion about this problem.

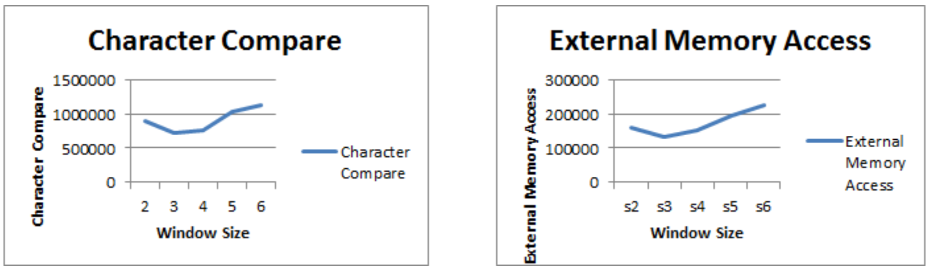

Fig. Changing the size of sampling window

In conclusion, after running the experiment, we can see that the assumption *A1* may not be correct in some cases. And in the assumption *A2*, instead of fixing the position of the sampling window, making it change between the patterns can also improve the performance. Finally, our experiment shows that the window size *W*=3 chosen in assumption *A3* is the optimized value in the current set of experiments.

5 Conclusion and Future Work

EHMA is an efficient single-pattern matching algorithm based on two-tier and cluster-wise matching strategy. In this paper, we verify a few key assumptions in this algorithm and conclude that in some cases, these assumptions may not be correct. We also present some rule in how to choose the input parameter to make the performance better. Combining all the observation that we have made can help to significantly improve the original algorithm.

In the future, we will try to have a look at other parameters of the algorithm, for example the gram size B1. Currently, in both Tzu-Fang et al. experiment and our experiment, B1 is fixed to 1 because of the computational complexity.

References

1. R.S. Boyer, J.S. Moor Waterman: A Fast String Searching Algorithm. Comm. ACM, vol. 20, no. 10, pp. 762--772 (1977)
2. A.V. Aho, M.J. Corasick: Efficient String Matching: An Aid to Bibliographic Search. Comm. ACM, vol. 18, no. 6, pp. 330-340 (1975)
3. S. Wu, U. Manber: A Fast Algorithm for Multi-Pattern Searching. Technical Report TR94-17, Dept. Computer Science, Univ. of Arizona (1994)
4. T.-F. Sheu, N.-F. Huang, H.-P lee: In-Depth Packet Inspection Using a Hierarchical Pattern Matching Algorithm. IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 2 (2010)
5. Snort, http://www.snort.org, 2012
6. C. Cowan: Defcon Capture the Flag: Defending Vulnerable Code from Intense Attack. Proc. DARPA Information Survivability Conf. and Exposition (DISCEX III'03) (2003)