Improved Single Keyword Pattern Matching Algorithm for Intrusion Detection System

K. Prabha,
Ph.D Research Scholar,
Erode Arts and Science College,
Erode, Tamil Nadu, India

S.Sukumaran, Ph.D
Associate Professor of Computer Science,
Erode Arts and Science College,
Erode, Tamil Nadu, India

ABSTRACT
With the spreading of the internet and online procedures requesting a secure channel, it has become an inevitable requirement to provide the network security. It is very clear that firewalls are not enough to secure a network completely because the attacks committed from outside of the network are stopped whereas inside attacks are not. This is the situation where intrusions detection systems (IDSs) are in charge. IDSs are used in order to stop attacks, recover from them with the minimum loss or analyze the security problems.

String matching algorithms are essential for IDS that filter packets and flows based on their payload. This work describes the concept of single keyword pattern matching algorithms. A new improved single keyword pattern matching algorithm is proposed. The new method reduces character comparisons, faster and more reliable in network security applications. The experimental results show that the new algorithm is highly efficient. Its search time is cut down significantly compared with other popular existing algorithms and its memory occupation stays at a low level. Moreover, conclusion on results is made and direction for future works is presented.

Keywords
Network Security, Pattern matching, Intrusion Detection

1. INTRODUCTION
Security attacks through internet have proliferated in recent years. Hence, information security is an issue of very serious global concern of the present time. The need for network security and in particular the need for Intrusion Detection Systems (IDS) have been brought out. IDSs, as originally introduced by Anderson [3] in 1980 and later formalized by Denning [1] in 1987, have received increasing attention in the recent years.

IDS are widely used and heavily depended upon. The continued growth in both network traffic and intrusion signature databases makes the performance of these systems increasingly challenging and important. Intrusions refer to the network attacks against vulnerable services, data-driven attacks on applications, host-based attacks like privilege escalation, unauthorized logins and access to sensitive files, or malware like viruses, worms and trojan horses. Intrusion detection means detecting unauthorized use of a system or attacks on a system or network.

IDS are implemented in software or hardware in order to detect these activities. IDSs collect information from a computer or a computer network in order to detect attacks and misuses of the system. Three types of data are used by IDSs. These are network traffic data, system level test data and system status files [2].

The heart of almost every modern IDS has a string matching algorithm. The IDS uses string matching to compare the payload of the network packet and/or flow against the pattern entries of intrusion detection rules [1, 2].

String matching is an important area in the wider domain of text processing. These algorithms are basic components used in implementations of practical softwares existing under most operating systems. Moreover, they emphasize programming methods that serve as paradigms in other fields of computer science. They also play an important role in theoretical computer science by providing challenging problems[4]. String matching generally consists of finding a substring (called a pattern) within another string (called the text).These string matching algorithms are used to inspect the content of packets and identify the attacks signature in IDS[9]. String matching consists of finding one, or more generally, of all the occurrences of a search string in an input string. In IDS applications, the pattern is the search string, while the payload is the input string. If more than one search string simultaneously matches against the input string, this is called multiple pattern matching. Otherwise, it is called single pattern matching. In this work, we considered only the single keyword pattern matching algorithms.

This paper presents the information about IDS pattern matching algorithms. An improved single keyword pattern matching algorithm is proposed to reduce the number of attempts and character comparison. The main aim for this work is to improve the efficiency of IDS detection engine. In the rest of the paper, we presented string matching algorithms for IDS (Section II), the related works and the proposed method (Section III), the results of implementation (Section IV) and the Conclusion (Section V).

2. SINGLE KEYWORD PATTERN ALGORITHMS FOR IDS
The Boyer-Moore algorithm and its variants are widely used in the string matching. The Horspool algorithm performs the comparison in a simple way, which works for most of the practical cases. The Brute Force algorithm requires no preprocessing of the pattern. In the Karp-Rabin Algorithm has the main idea is that instead of using comparisons it involves mathematical computations which more specifically extends to the notion of hashing[15].

2.1 Boyer-Moore algorithm (BM)
The Boyer-Moore algorithm is one of the exact string matching algorithms that used in single pattern matching. The algorithm uses two tables or functions, which is used to move the sliding window to the right. The first table is called “bad character shift”, while the second table called “good suffix shift”. The algorithm is faster when it is working with small pattern size, but it is slower when it is working with large pattern size[15]. The BM algorithm is given below:

```
Algorithm BoyerMooreMatch(T, P, S)
L = occFunction ()
i = m - 1
j = m - 1
while i > n - 1
{
 if T[i] = P[j]
    if j = 0
       return i //match at i
    else
      i = i - 1
      j = j - 1
  else //character-jump
  l = L[T[i]]
  i = i + m – min(j, 1 + l)
  j = m - 1
 }
 return -1 //no match

void occFunction()
{
 char a;
 int j;
 for (a = 0; a < alphabetsize; a++)
  occ[a] =- 1;
 for (j = 0; j < m; j++)
 {
  a = p[j];
  occ[a] = j;
 }
}
```

The algorithm preprocesses the pattern and creates two tables, which are known as Boyer-Moore bad character (bmBc) and Boyer-Moore good-suffix (bmGs) tables. For each character in the alphabet set, a bad character table stores the shift value based on the occurrence of the character in the pattern. On the other hand, a good-suffix table stores the matching shift value for each character in the pattern. The maximum of the shift value between the bmBc (character in the text due to which a mismatch occurred) dependent expression and from the bmGs table for a matching suffix is considered after each attempt, during the searching phase. This algorithm forms the basis for several pattern matching algorithms.

2.2 Horspool Algorithm (HP)

Horspool algorithm is based on Boyer Moore algorithm. Snort IDS uses a modified version of the algorithm called Boyer-Moore-Horspool algorithm to maintain memory usage and speed up during searching phase. Unlike Boyer-Moore algorithm, which uses two tables; bad character shift and good suffix shift, the Horspool algorithm uses only one table (bad character shift) [14]. Hence, the algorithm is more efficient in practical situations where the alphabet size is large and the length of the pattern is small.

2.3 Brute Force Algorithm (BF)

The Brute Force algorithm requires no preprocessing of the pattern. The comparison can be done in any order either from left to right or from right to left. If all the characters match, then it is said to be a match. If not, the algorithm shifts the pattern by exactly one position to the right [17]. The expected number of character comparisons in Brute Force algorithm is 2n. The algorithm is given below:

```
Algorithm brute (text, pattern)
{
   n = length(text)
   m = length(pattern)
   for i=0 to (n-m)
   {
    j = 0
    while((j<m) and
      (text(i+j) = pattern(j))
       j++
    if  j = m
       return i // match at i
   }
   return –l  // no match
}
```

2.4 Karp-Rabin Algorithm (KR)

The Karp-Rabin Algorithm was created by Michael Rabin and Richard Karp. They used a completely different approach than the single keyword methods[13]. The main idea is that instead of using comparisons it involves mathematical computations which more specifically extends to the notion of hashing. The application of hashing (converting each string into a numeric value) has always been a useful approach when it comes down to string matching. If both words have different hash values then we conclude they are different. But if their hash values are the same we cannot conclude they are the same string and will have to perform further comparisons.

```
Karp-Rabin-Matcher(T,P,d,q)
n = length(T)
m = length(P)
h = d^(m-1) mod q
p = 0
t0 = 0
for i = 1 to m   //preprocessing
{
   p = (d*p + P[i]) mod q
   t0 = (d*t0 + T[i]) mod q
 }
 for s = 0 to n-m   //matching
{
   if p = ts
       if P[1..m] = T[s+1..s+m]
       print “Pattern occurs with shift” s
   if s < n-m
   ts+1 = (d*(ts-T[s+1]*h) + T[s+m+1]) mod q
}
```

3. PROPOSED METHOD

The improved single keyword pattern matching algorithm which is formulated based on the two algorithms Horspool and Karp-Rabin algorithm. Karp-Rabin algorithm is based on hashing approach but not the comparison of characters, which is consider as the advantage of this algorithm. But its weakness is the enormous time needed when long patterns are present [17]. On the other hand, the Horspool algorithm is easy and works in any order. In most situations that it applied on and has a high performance compare to other algorithms. It is easy to implement and has less memory space so, it can be implement in any case that need the exact string matching algorithm for small pattern and large pattern size [14].

3.1 Improved single keyword pattern matching algorithm (ISPMA)

The two phases of the proposed algorithm are (i) preprocessing phase, (ii) searching phase.

Step1 : In the first phase, the ISPMA performs the same preprocessing phase as in the existing two algorithms. It prepares the hash function used in KR algorithm and the *bmBc* table used in HP algorithm for the pattern.

Step2 : The process of computing hash functions for the patterns and text window are exactly the same as the process of creating them in the existing KR algorithm. The bmBc table is the same as it was in the existing HP algorithm.

In the searching phase, the ISPMA performs the comparison between the pattern and the text by utilizing the advantages of the KR and HP.

Step 3: After the preprocessing phase has finished, the comparison start between the text and pattern by comparing the numerical value of pattern hash and window text hash.

Step 4 : Whether, if the two hash value are not identical then the ISPMA perform the shifting.

Step 5: Next Shift to the right based on the values of right most character for the window text in the *bmBc* table. This will speed up the algorithm during the comparison process and it reduced the number of character comparison by using the hash function.

For example, Take the pattern (PQRPRPRP)
After preprocessing phase,

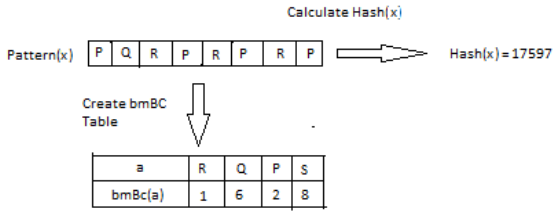

Fig3.1 Preprocessing Phase

For the previous pattern (PQRPRPRP) and the text (PQRSQPQRPRPRPSRSRQRPS). The searching phase for the ISPMA is depicted in below Figure.

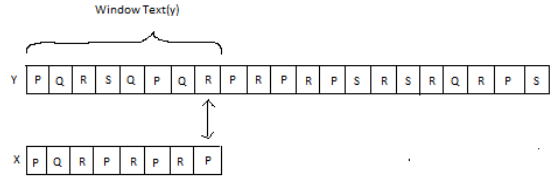

Fig3.2 Searching Phase

Here the Hash (x) = 17597 and Hash (y) = 17819 so the hash value are not equals. The ISPMA shift the value of character R in the *bmBc* table which is 1.
The operation of comparison continues for the next shift in window text as shown in below Figure.

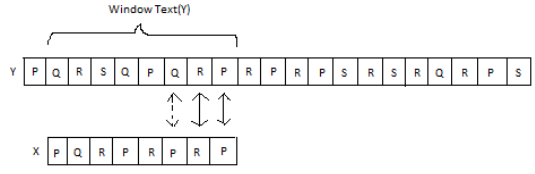

Fig3.3 Searching Phase for next window test

Here the Hash (x) = 17597 and Hash (y) = 17533 so the hash value are not equals. The ISPMA shift the value of character P in the *bmBc* table which is 2. Similarly, the process of ISPMA continues until all characters in the text are being compared and whether the mismatching or matching is found.

4. RESULTS

The proposed approach is implemented using MATLAB. The evaluation of the proposed method is performed based on the factors Efficiency, Runtime, Space and Accuracy.

The result of the experiments are presented below:

Table 4.1 Efficiency Comparison

Algorithm	Character Comparison	Number of Attempts
BM	15	10
KR	12	8
HP	10	8
ISPMA	8	6

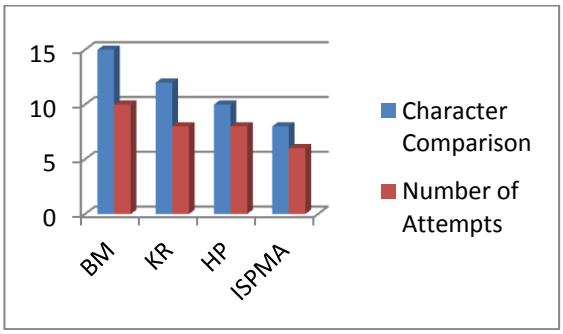

Fig 4.1 Efficiency Comparison

The result shows that the ISPMA reduces the number of character comparison to 8 and reduce the number of attempts to 6. This is because of hashing approach of Karp- Rabin algorithm to perform the character comparison and depends on shift table of Horspool algorithm to perform the movement of pattern.

4.1 Time performance

The running-time performance, also referred to as time complexity, is measured in number of machine steps, and in this case we are primarily concerned with character or byte comparisons. To present the results of the running time of algorithms, we vary the input size, where the input is the English words. The number of patterns to be matched remains the same. The running time (in milliseconds) for the algorithms are recorded in the following table:

Table 4.2 Runtime Comparison

Input Size	Running Time (in milliseconds)			
	BM	***KR***	***HP***	***ISPMA***
20000	15	15	17	13
60000	40	45	46	38
100000	68	73	79	65
140000	102	102	108	101
180000	119	132	139	115
200000	133	144	159	123

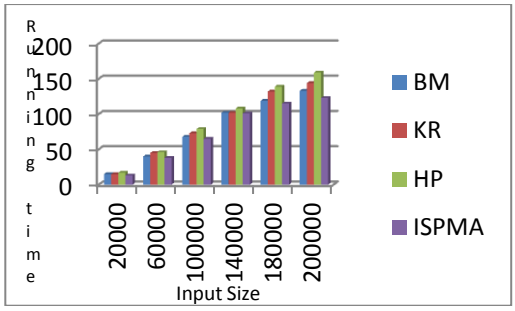

Fig 4.2 Runtime Comparison

4.2 Space performance

The amount of memory consumed while the algorithm runs, is considered only in addition to the necessary space to store the keyword and input. The keyword and the keyword set must always be stored. The space performances of proposed algorithm and the three algorithms were compared using one pattern. The results are shown in Fig4.3. The results show that ISPMA size is smaller than that of the other algorithms, for the same pattern.

Table 4.3 Space Comparison

Algorithm	Pattern Length (Byte)	Memory Space(MB)
BM	20	180
KR	20	160
HP	20	140
ISPMA	20	100

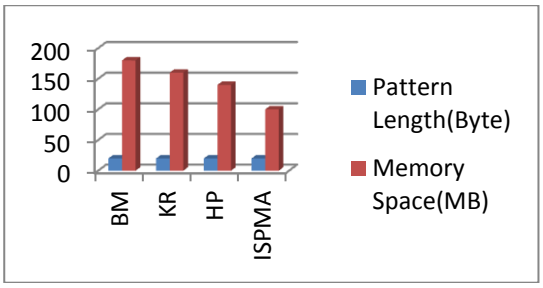

Fig 4.3 Space Comparison

4.3 Accuracy Performance

The number of patterns are vary, the accuracy for the four algorithms are shown in Figure 4.4. Horspool and Boyer-Moore have the minimum accuracy because the shifted values are affected by increasing signature length. The difference between them is very small. Karp-Rabin and improved single keyword pattern matching algorithms were not affected by the increased signature length because their shifted values are always one byte.

Table 4.4 Accuracy Comparison

Number of patterns	Accuracy (%)			
	BM	***KR***	***HP***	***ISPMA***
50	92	95	95	99.5
100	91.8	94.5	94.5	99
150	91	94.3	94.3	98.5
200	90.6	94	93	98.2
250	90.3	93.7	92.5	98

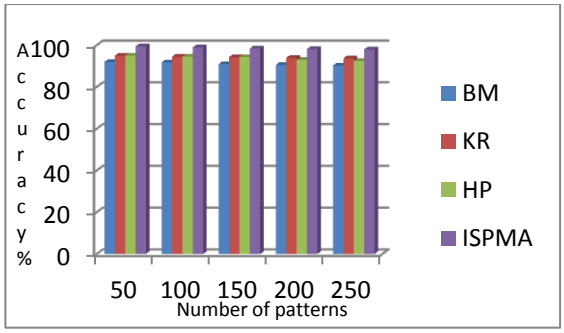

Fig 4.4 Accuracy Comparison

5. CONCLUSION

This work identifies the number of promising algorithms and provides an overview of recent developments in the single keyword pattern matching for IDS. Boyer-Moore Algorithm uses two tables and matching starts with right to left, but in Horspool uses only one table and the matching is faster than the Boyer-Moore. The Brute force algorithm requires no preprocessing of the pattern. Karp-Rabin algorithm is based on hashing approach. The proposed ISPMA algorithm is compared with the exiting algorithms and the result shows that the algorithm is faster and more reliable in network security applications. The results of algorithm show an improvement in average comparing, faster than the original algorithms, less character comparison and performs less number of attempts compared to the exiting algorithms. In future work, we will enhance the method by moving toward the parallel computing to reduce the workload of system and consequently improve the speed and accuracy of the detection of malicious activities.

6. REFERENCES

[1] Apostolico and M. Crochemore. String pattern matching for a deluge survival kit. Handbook of massive data sets, 2002.

[2] B. Kim, S. Yoon and J. Oh, "Multi-hash based Pattern Matching Mechanism for High-Performance Intrusion Detection," International Journal of Computers. Vol. No. 3. Issue1, 2009.

[3] Bace R. An introduction to intrusion detection and assessment for system and network security management. ICSA Intrusion Detection Systems Consortium Technical Report, 1999.

[4] Christian Charras, Thierry Lecroq, "Handbook of Exact String Matching Algorithms", King's College Publications, 2004, ISBN :0954300645.

[5] Coit C J, Staniford S, McAlerney J, "Towards faster string matching for intrusion detection or exceeding the speed of Snort", Proceedings of the DARPA Information Survivability Conference and Exposition II (DISCEX'01). Los Alamitos, CA, USA: IEEE Comput. Soc., 2001.

[6] Denning, Dorothy E.: Information Warfare and Security. Addison Wesley Longman, Inc., Reading, 1999.

[7] Fisk M, Varghese G, "An analysis of fast string matching applied to content-based forwarding and intrusion detection", Technical Report CS2001-0670. San Diego: University of California, 2002.

[8] Martin Roesch, "Snort-Lightweight Intrusion Detection for Networks", Stanford Telecommunications, Inc, 13th LISA conference, 1999.

[9] Meier, Michael; Holz, Thomas: Intrusion Detection Systems List and Bibliography. http://wwwrnks.informatik.tucottbus.de/en/security/ids.html, 2003.

[10] M. Fisk, and G. Varghese, "An analysis of fast string matching applied to content-based forwarding and intrusion detection", Technical Report CS2001-0670 (updated version), University of California - San Diego, 2002.

[11] N. Tuck, T. Sherwood, B. Calder, and G. Varghese, "Deterministic memory-efficient string matching algorithms for intrusion detection", Proc. IEEE Infocom, vol. 4, March 2004.

[12] RRehman RU, Intrusion detection systems with snort. Upper Saddle River, New Jersey, Publishing as Prentice Hall PTR, 2003.

[13] R. M. Karp and M. O. Rabin. "Efficient randomized pattern-matching algorithms", IBM Journal of Research and Development, Vol.31, no.2, 1987.

[14] R.N.Horspool,"Practical fast searching in strings", Software-Practice and Experience, Vol. 10, no. 6, 1980

[15] R. S. Boyer and J. S. Moore, "A fast string searching algorithm", Communications of the ACM, Vol.20, no.10, 1977.

[16] S. Dharmapurikar, J.W. Lockwood, "Fast and Scalable Pattern Matching for Network Intrusion Detection Systems", IEEE Journal on Selected Areas in Communications, vol. 24, 2006.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms, Second Edition. The MIT Press and McGraw-Hill Book Company, 2002.

[18] W. Yang, B-X. Fang, B. Liu, and H-L. Zhang, "Intrusion detection system for high-speed network," Computer Communications. Vol 27, 2004.

[19] W. Lee, J. D. Cabrera, A. Thomas, N. Balwalli, S. Saluja, and Y. Zhang, "Performance adaptation in real-time intrusion detection systems," in RAID, 2002.

[20] YU Jianming, XUE Yibo, LI Jun, "Memory Efficient String Matching Algorithm for Network Intrusion Management System", Tsinghua Science and Technology, ISSN 1007-0214, October 2007.

7. ABOUT AUTHORS

K.Prabha received B.Sc Computer Science and M.Sc Computer Science Degree from Bharathiar University, Coimbatore and M.Phil in Periyar University, Salem. She pursuing Ph.D degree in Computer Science at Bharathiar University. She has 7 years of teaching experience. She is working as Assistant Professor of Computer Science in Erode Arts and Science College, Erode, Tamilnadu. Her research interests include Network Security and Data Mining.

Dr. S. Sukumaran graduated in 1985 with a degree in Science. He obtained his Master Degree in Science and M.Phil in Computer Science from the Bharathiar University. He received the Ph.D degree in Computer Science from the Bharathiar University. He has 25 years of teaching experience starting from Lecturer to Associate Professor. At present he is working as Associate Professor of Computer Science in Erode Arts and Science College, Erode, Tamilnadu. He has guided for more than 40 M.Phil research Scholars in various fields and guided one Ph.D Scholar. Currently he is Guiding 5 M.Phil Scholars and 8 Ph.D Scholars. He is member of Board studies of various Autonomous Colleges and Universities. He published around 15 research papers in national and international journals and conferences. His current research interests include Image processing, Network Security and Data Mining.