A Secured Pattern Matching Technique for Intrusion Detection System in Wireless Sensor Network

R.Hamsaveni
Research Scholar, SCSVMV University, Kanchipuram. hamsamurugs@gmail.com, cell:9442212136

Dr.G.Gunasekaran,M.E.,PH.D.(Engg)
Principal, Meenakshi College of Engineering,Chennai.gunaguru@yahoo.com cell:9444177718

***Abstract*-The evaluation of Wireless Sensor Networks (WSN) for performance is the most familiar research area and there are various kinds of possibilities in which security can be breached and enable the hackers to utilize secured information. The Intrusion Detection System (IDS) is one of the major defensive methods used against attack in WSN. There are three types of methodologies used for detect intrusion on the network, they are Signature based, Anomaly based and Stateful protocol analysis. This paper is based on Signature based Intrusion Detection System methodology. 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 Logo Pattern Matching algorithm (LPM) is proposed. The new method reduces character comparisons, faster and more reliable in network security applications. Many pattern matching algorithms have high false alarm. The proposed LPM scheme reduces false alarm percentage to an extensive level and detects the intruder efficiently. The experimental results implemented in SNORT and 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 - Signature Based, Wireless Sensor Network, Intrusion Detection System, Logo Pattern Matching Algorithm.

I. 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) has been brought out. IDSs, as originally introduced by Anderson in 1980 and later formalized by Denning 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 make 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. 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.

String matching is an important area in the wider domain of text processing. These algorithms are basic components used in implementations of practical software's 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. 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. 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. A Logo 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 Related Work for IDS (Section II), the Architecture of SNORT (Section III) Result (Section IV) Conclusion (Section V).

II. RELATED WORK

Before we go in detailed study of our algorithm and its comparisons with other algorithms we would like to give a brief overview of the already existing algorithms along with their complexity analysis and try to explain why our algorithm performs better than them.

Bruce W. Watson (2002) gave a more practical algorithm for a special characterization of 'matching productions', using the transitive closure of a relation to deal with chain rules in the pattern grammar. The algorithm was further improved through the introduction of a reverse trie. The idea of shift distances greater than one symbol (as in the Boyer–Moore and Commentz-Walter algorithms) was also introduced [1].

V. K. Pachghare, Dr. Parag Kulkarni,(2008) proposed an efficient pattern matching Algorithm in order to overcome the troubles in network traffic [2].

Dai Hong (2012) proposed enhanced pattern matching performance using improved Boyer Moore Horspool Algorithm. Moreover the algorithm combines the deterministic finite state; the improved Boyer Moore Horspool Algorithm takes full use of the matching information to skip several characters. The proposed algorithm saves more memory resource, especially adapts to pattern string character sets and text character sets, matching speed increases greatly and pattern length and number influence hardly [3].

Priya jain,Shikha Pandey (2012) have done comparative study on various existing Pattern Matching Algorithms and confirmed that the Bayer Moore Pattern matching algorithm is the most efficient as well as fast and gives the accurate results [4].

Abhay Bhatia,Shashikant, Robin Choudhary(2012) exposed the challenges in pattern matching and shows the comparison between the existing and a proposed Optimized Pattern Matching (OPM) algorithm for finding out the matched links with the given number of links [5].

Lata ,Kashyap Indu and Nagaraju (2013 and 2015) identified intrusion can be possible on the header part or payload part. Intrusion detection algorithms normally have high false alarm. The authors have proposed an algorithm which lowers the false alarm considerably [6].

K. Prabha, S.Sukumaran (2014) derived an ISPMA algorithm for single keyword pattern matching for IDS. They have compared their result with Boyer-Moore Algorithm, Horspool, Karp-Rabin algorithm, Brute force algorithm and illustrated that their ISPMA algorithm is faster and more reliable [7].

Akinul Islam Jony(2014)illustrates a widely used multiple string patterns matching algorithm . A theoretical and experimental result along with the analysis and discussion of the algorithms was presented [8].

Urmila Patel, Mitesh Thakkar(2014), proposed an efficient version of Bidirectional Pattern is An Efficient Exact Single Pattern Matching (EESP) algorithm in which they tried to reduce pre-processing time and also found its all occurrences of the Pattern in to long Text String [9].

Nguyen Le Dang, Dac-Nhuong Le, and Vinh Trong Le,(2016), presented a new algorithm for multiple-pattern exact matching. In their algorithm they reduced character comparisons and memory space based on graph transition structure and search technique using dynamic linked list. Theoretical analysis and experimental results, when compared with previously known pattern-matching algorithms, was highly efficient in both space and time [10].

II. SINGLE KEYWORD PATTERN ALGORITHMS

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 concepts.

3.1 Boyer-Moore (BM) Algorithm

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[7]. 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;
}
}
```

Algorithm 1. Boyer-Moore algorithm

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.

3.2 Horspool (HP) Algorithm

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) . Hence, the algorithm is more efficient in practical situations where the alphabet size is large and the length of the pattern is small.

3.3 Karp-Rabin (KR) Algorithm

The Karp-Rabin Algorithm was created by Michael Rabin and Richard Karp. They used a completely different approach than the single keyword methods. 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
t_0 = 0
for i = 1 to m        //preprocessing
{
p = (d*p + P[i]) mod q
t_0 = (d*t_0 + T[i]) mod q
}
For  s = 0 to n-m   //matching
{
if p = t_s
if P[1..m] = T[s+1..s+m]
print "Pattern occurs with shift" s
 if s < n-m
t_{s+1} = (d*(t_s-T[s+1]*h) + T[s+m+1]) mod q
}
```

Algorithm 3. Karp-Rabin Algorithm

3.4 Brute Force (BF) Algorithm

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. 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
}
```

Algorithm 4. Brute Force Algorithm

3.5. Knuth-Morris-Pratt Algorithm (KMP)

Knuth-Morris -Pratt proposed a keyword pattern matching algorithm (herein the KMP algorithm) that runs left to right over its text input in linear time and improves on an algorithm proposed earlier by only Morris and Pratt . This algorithm's pre-computation creates an array with information about how the keyword matches against shifts of itself. For example knowing we have matched exactly r characters somewhere in the input allows us to determine that certain shifts are invalid; thus, avoiding the shifts that the naive brute force algorithm executes only to then fail at a subsequent match attempt.

As a concrete example let us say the keyword pattern x = ababaca and we have just matched 5 characters in y starting at position i only to find a mismatch for the character c at position i + 5. The naive shift of the brute force algorithm would shift the pattern by one position, but of course it would fail because the character b at y[i + 1] would not even match the first character of the keyword x. The KMP algorithm knows what the appropriate shift should be, and may shift over multiple positions without missing any potential matches. The trick to doing this correctly is in the pre-computation step where the prefix function for the keyword is built into an array. If we define x as the first r characters of x that have been matched at any point, the prefix function array table at position r (table[r]) contains the length of the longest prefix of x that is a proper suffix of x $_r$. Using this information the shift to the next position is always possible to calculate as (the current position) + (the number of matched characters before the mismatch (r)) - (table[r]) .

Algorithm 3.5 below shows the KMP algorithm and the pre-computation step of how the prefix function array (table) is created in time and with memory space O(m).In the worst case the matching phase of the KMP algorithm executes 2n − 1 character comparisons .

Procedure KMP(x, m, y, n)

- Input:
- x ← array of m bytes representing the keyword
- m ← integer representing the keyword length
- y ← array of n bytes representing the text input
- n ← integer representing the text length

```
table ←Compute Prefix KMP(x, m)          // Pre-Computation
q ← 0
for i = 1 → n do                          // Matching
while q > 0 and x[q + 1] 6 = y[i] do
q ← table[q]
end while
if x[q + 1] = y[i] then
q ← q + 1
end if
if q = m then
output i − m
q ← table[q]
end if
end for
end procedure
```

procedure Compute Prefix KMP(x, m)

- Input:
- x ← array of m bytes representing the keyword
- m ← integer representing the keyword length

```
table ← newArray[m + 1]   // Prefix Shift Table
table[1] ← 0
k ← 0
for q = 2 → m do
while k > 0 and x[k + 1] 6 = x[q] do
k ← table[k]
end while
if x[k + 1] = x[q] then
k ← k + 1
end if
table[q] ← k
end for
return table
end procedure
```

Algorithm 5. Knuth-Morris-Pratt Algorithm

Main points- The main points of the Knuth-Morris-Pratt algorithm are outlined below :

- Performs the comparisons from left to right.
- Preprocessing phase in O (m) space and time complexity.

IV. Architecture of Signature Based Network Intrusion Detection System using SNORT

4.1 Method of Data Collection

Marly Roesch who is a developer of the Snort systems defines a given network at lightweight network intrusion detection system" when the network traffic and the packets on the IP network can be analyzed and logged in real-time. Like a network snipper who is based on the network packet collecting system library called "libpcap". Since the libpcap will be log into a database for my project and then with the help of pattern matching algorithm m compare the content of the packet with the rule set present in the database .

4.2 Source of Data Collection

SNORT is a Free and open source NIDS. It utilizes the rule driven language to perform the pattern matching. Since the detection process of SNORT is heavily depend on the rule set present in the database.

4.3 Structure of IDS Rule

All IDS rules have two logical parts: rule header and rule options. This is shown in Figure 4. 1. The rule header contains information about what action a rule takes. It also contains criteria for matching a rule against data packets. The options part usually contains an alert message and information about which part of the packet should be used to generate the alert message. The options part contains additional criteria for matching a rule against data packets. A rule may detect one type or multiple types of intrusion activity. Intelligent rules should be able to apply to multiple intrusion signatures.

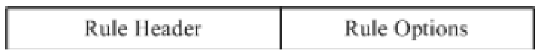
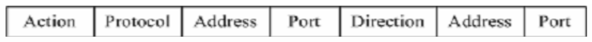


Figure 4.1 Basic structures of IDS rules.

The general structure of rule header is shown in Figure 4. 2. The action part of the rule determines the type of action taken when criteria are met and a rule is exactly matched against a data packet. Typical actions are generating an alert or log message or invoking another rule.



Action	Protocol	Address	Port	Direction	Address	Port

Figure 4.2 Structure of IDS rule header.

The protocol part is used to apply the rule on packets for a particular protocol only. This is the first criterion mentioned in the rule. Some examples of protocols used are IP, ICMP, UDP etc. The address parts define source and destination addresses. Addresses may be a single host, multiple hosts or network addresses. We can also use these parts to exclude some addresses from a complete network. There are two address fields in the rule. Source and destination addresses are determined based on direction field. As an example, if the direction field is —- >□, the Address on the left side is source and the Address on the right side is destination. In case of TCP or UDP protocol, the port parts determine the source and destination ports of a packet on which the rule is applied. In case of network layer protocols like IP and ICMP, port numbers have no significance. The direction part of the rule actually determines which address and port number is used as source and which as destination. For example, consider the following rule that generates an alert message whenever it detects an ICMP1 ping packet (ICMP ECHO REQUEST) with TTL equal to 100, as follow:

alert icmp any any -> any any (msg: "Ping with TTL=100"; \ttl: 100;)

The part of the ru le before the starting parenthesis is called the rule header. The part o f the rule that is enclosed by the parentheses is the options part. The header contains the following parts, in order:

• *A rule action*: In this rule the action is —alert□, which means that an alert will be generated when conditions are met. The packets are logged by default when an alert is generated. Depending on the action field, the rule options part may contain additional criteria for the rules.

• *Protocol:* In this rule the protocol is ICMP, which means that the rule will be applied only on ICMP-type packets. In the IDS detection engine, if the protocol of a packet is not ICMP, the rest of the rule is not considered in order to save CPU time. The protocol part plays an important role when you want to apply IDS ru les only to packets of a particular type.

• *Source address and source port.* In this example both of them are set to —any□, which means that the rule will be applied on all packets coming from any source. Of course port numbers have no relevance to ICMP packets. Port numbers are relevant only when protocol is either TCP or UDP.

• *Direction.* In this case the direction is set from left to right using the -> symbol. This shows that the address and port number on the left hand side of the symbol are source and those on the right hand side are destination. It also means that the rule will be applied on packets traveling from source to destination. You can also use a <- symbol to reverse the meaning of source and destination address of the packet. Note that a symbol <> can also be used to apply the rule on packets going in either direction.

• *Destination address and port address.* In this example both are set to any, meaning the rule will be applied to all packets irrespective of their destination address. The direction in this rule does not play any role because the rule is applied to all ICMP packets moving in either direction; due to the use of the keyword "any" in both source and destination address parts. The options part enclosed in parentheses shows that an alert message will be generated containing the text string "Ping" with TTL=100□ whenever the condition of TTL=100 is met. Note that TTL or Time To Live is a field in the IP packet header.

SNORT is a signature based NIDS. SNORT can be divided into five major components that are each critical to intrusion detection.

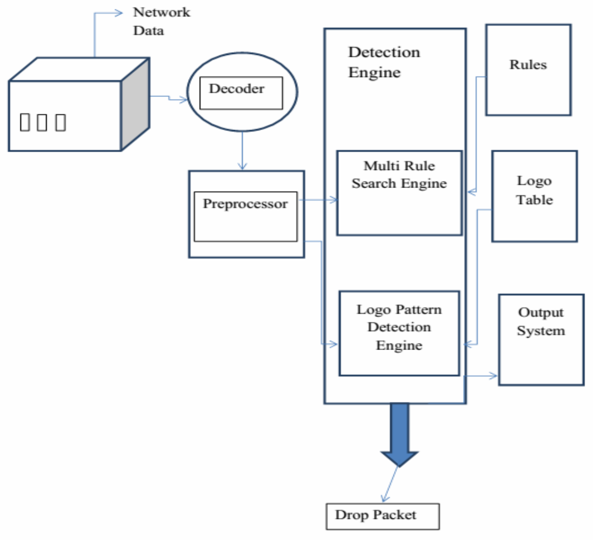


Figure 4. 3. SNORT Architecture.

The first is the packet capturing mechanism. SNORT relies on an external packet capturing library (libpcap) to sniff packets. After packets have been captured in a raw form, they are passed into the packet decoder. The decoder is the first step into SNORT's own architecture. The packet decoder translates specific protocol elements into an internal data structure. After the initial preparatory packet capture and decode is completed, traffic is handled by the preprocessors. Any numbers of pluggable preprocessors either examines or manipulate packets before handing them to the next component: the detection engine. There are two steps in detection engine. The First is Multi Rule Search Engine which fetches the rules from Rules table. The rules are in predefined format, stored in a table. The detection engine performs simple tests on a single aspect of each packet to detect intrusions. The Second step is Logo pattern detection engine which senses the logo in every incoming packet. The logo is mapped with the logo table, if the logo matches, the detection engine confirms there is no intrusion or else intrusion is detected. The last component is the output plug-in, which generates alerts to present suspicious activity.

V. RESULTS

The proposed approach is implemented using SNORT. The evaluation of the proposed method is performed based on the factors Efficiency, Runtime, Space and Accuracy. The result of the experiments is presented below:

5.1. Efficiency Performance

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

Table 1. Efficiency Comparasion

Algorithm	Character Comparison	Number of Attempts
BM	20	15
KR	18	14
HP	16	14
BF	13	12
KMP	10	8
LPM	8	6

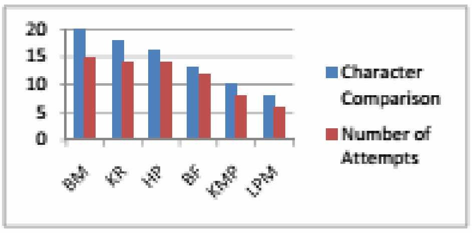


Fig 1. Efficiency Comparasion

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

5.2. 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 is recorded in the following table.

Table 2. Runtime Comparison

	Running Time (in milliseconds)				
Input Size	BM	KR	HP	KMP	LPM
20000	20	20	18	16	12
60000	45	40	41	39	30
100000	70	75	79	81	66
140000	104	104	108	110	100
180000	120	132	135	139	112
200000	134	142	140	159	123

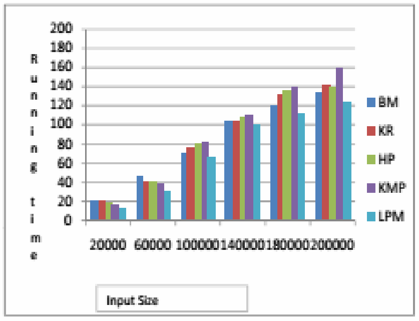


Fig 2. Runtime Comparison

5.3. 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 five algorithms were compared using one pattern. The results are shown in Fig 5.3.The results show that LPM size is smaller than that of the other algorithms, for the same pattern.

Table 3. Space Comparison

Algorithm	Pattern Length (Byte)	Memory Space (MB)
BM	20	190
KR	20	170
HP	20	150
BF	20	130
KMP	20	125
LPM	20	100

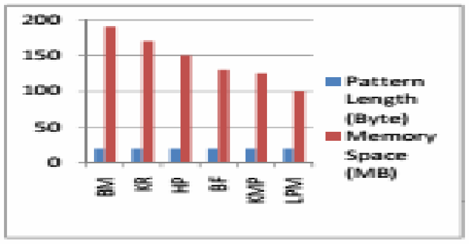


Fig 3. Space Comparison

5.4. Accuracy Performance

The number of patterns are vary, the accuracy for the five algorithms are shown in Figure 4. Horspool,Karb-Roubin,and BoyerMoore and Brute-Force Algorithms have the minimum accuracy because the shifted values are affected by increasing signature length. The difference between them is very small. Knuth-Morries-Pratt and Logo Pattern Matching algorithms were not affected by the increased signature length because their shifted values are always one byte.

Table 4. Accuracy Comparison

Number of Patterns	Accuracy(%)					
	BM	KR	HP	BF	KMP	LPM
50	93	96	96.4	95.3	95.1	99.5
100	92.6	95.8	95.5	94.8	94.8	99.3
150	92	95.8	95.3	94.6	94.6	99.2
200	91.2	94.2	94.2	94.1	93.5	98.8
250	90.6	93.9	93.8	93.8	92.4	98.5
300	90.2	92	93.4	93.6	92	98.2

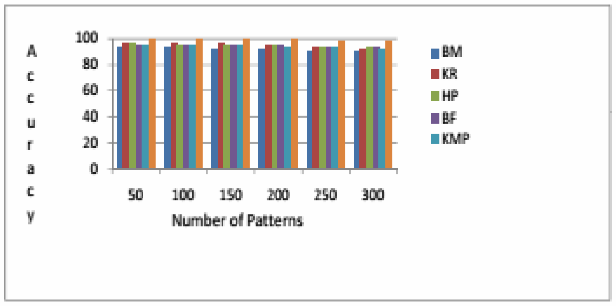


Fig 4. Accuracy Comparison

VI. 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. The Knuth -Morris -Pratt algorithm performs the comparisons from left to right. Karp-Rabin algorithm is based on hashing approach. The proposed LPM 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.

VII. REFERENCES

[1] Bruce W. Watson “A new regular grammar pattern matching algorithm” 2002 Elsevier Science, Theoretical Computer Science 299 (2003) 509 – 521.
[2]V.K.Pachgharel, Paragkulkarni,” Network Security Based On Pattern Matching: An Overview “, International Journal of Computer Science and Network Security, VOL.8 No.10, October 2008
[3] Dai Hong, Anshan, Liaoning,” Enhanced Pattern Matching Performance Using Improved Boyer Moore Horspool Algorithm”, Journal of Convergence Information Technology, Volume7, Number4, March 2012
[4] Priya jain, Shikha Pandey,” Comparative Study on Text Pattern Matching for Heterogeneous System”, International Journal of Computer Science & Engineering Technology , ISSN : 2229-3345 Vol. 3 No. 11 Nov 2012.
[5]Abhay Bhatia, hashikant, Robin Choudhary, “Comparative Study of Pattern Matching Using Text Mining”, International Journal of Research Review in Engineering Science and Technology (ISSN 2278- 6643) | Volume-1 Issue-1, June 2012
[6]Lata, Kashyap Indu,” Novel Algorithm for Intrusion Detection System”, International Journal of Advanced Research in Computer and Communication Engineering,Vol. 2, Issue 5, May 2013
[7]K. Prabha, S.Sukumaran, “Improved Single Keyword Pattern Matching Algorithm for Intrusion Detection System”, . International Journal of Computer Applications (0975 – 8887) Volume 90 – No 9, March 2014
[8] Akinul Islam Jony,” Analysis of Multiple String Pattern Matching Algorithms”, International Journal of Advanced

Computer Science and Information Technology ,Vol. 3, No. 4, 2014, Page: 344-353, ISSN: 2296-1739

[9]Urmila Patel, Mitesh Thakkar,"An Efficient Exact Single Pattern Matching Algorithm", International Journal of Advanced Research in Computer Engineering & Technology Volume 3 Issue 5, May 2014

[10]Nguyen Le Dang, Dac-Nhuong Le, and Vinh Trong Le," A New Multiple-Pattern Matching Algorithm for the Network Intrusion Detection System "International Journal of Engineering and Technology, Vol. 8, No. 2, April 2016

ABOUT AUTHORS

R.Hamsaveni received B.Sc Computer Science from Madras University and M.Sc Computer Science Degree from Bharathidasan University, Trichy and M.Phil in Mother Tersa Women's,Kodaikanal. She is pursuing Ph.D degree in Computer Science at SCSVMC University. She has 16 years of teaching experience. She is working as Assistant Professor of Computer Science in SVCET, Chittoor. A.P. Her research interests include Network Security.

Prof. Dr. G. GUNASEKARAN graduated in 1989 with a Bachelor of Engineering degree in Computer Science. He obtained his Master of Engineering Degree in Computer Science in the year 2001.He received the Ph.D degree in Computer Science and Engineering from the Jadavpur University, Kolkata in the year 2009. He has 25 years of teaching experience starting from Lecturer to Associate Professor. At present he is working as Principal of Meenakshi College of Engineering, Chennai, Tamilnadu. He has guided for more than five 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 Data Mining and Network Security.