I.J. Information Technology and Computer Science, **2015, 06, 52-58**
Published Online May 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.06.07

A New Platform NIDS Based On WEMA

Adnan A. Hnaif
Al-Zaytoonah Private University of Jordan, Computer Networks Department, Amman, 11733, Jordan
Email: adnan_hnaif@zuj.edu.jo

***Abstract*—** The increasing speed of today's computer networks directly affects the performance of Network Intrusion Detection Systems (NIDS) in terms of speed of detection of threads. Therefore, the performance of the existing algorithms needs to be improved to enhance the speed of detection engine used in NIDS applications. Hence, this paper defines a new platform NIDS to enhance the speed of detection engine based on Weighted Exact Matching Algorithm (WEMA). Furthermore, this platform can be run in sequential and in parallel mode, using the pthread techniques, in order to increase the total performance of NIDS applications.

***Index Terms*—** Intrusion Detection Systems, Exact String Matching Algorithms, Multi-Threading Technology.

I. Introduction

Network security is responsible for protecting the information passing through any networks from the intruders. Moreover, network security refers to all hardware and software functionalities that are necessary to provide an acceptable level of protection to the network [1]. For example, a firewall is a system that is used to secure the internal network from the external traffic [2]. However, the traditional firewalls are insufficient to ensure network security from the internal network, and also a firewall filters all unwanted network traffic but allows some of the services (i.e. VoIP) to pass [3]. Therefore, Intrusion Detection Systems (IDS) can be used to detect these intrusions that could affect the network. Meaning, IDS is a system works after the firewall to provide unauthorized people from accessing the network [2].

NIDS is one of the IDS techniques, and it works by matching the string pattern against packet payload using one of the exact string matching algorithms (misuse detection). And, NIDS also can works to monitor and detect special types of events in a system (anomaly detection) [4].

NIDS suffers from a slow speed of its detection engine in linear and in parallel modes, and a lot of overhead costs in some of the parallelized technology. Thus, the main goal of this paper is to propose a new platform NIDS to enhance the speed of the intrusion detection engine based on the packet payload.

This paper contributions can be summarized as follow: First, NIDS based on WEMA: A new platform NIDS based on WEMA to enhance the speed of the detection engine for the packet payload in a liner phase. Second, load balance: A new platform is introduced to increase the speed of the detection engine by utilizing the multi-threading technology.

This paper is organized into six sections. This section presents an introduction and background for the firewall and NIDS. In section two, we discuss the most important exact string matching algorithms used in NIDS, and will also discuss the most current and related works in NIDS. Section three discuss how the proposed solution was designed. The new platform for the packet payload is also described in this section. The design and implementation details and issues are discussed in section four. The illustration of the experimental direction and the implementation of the detection engine are also mentioned. The results obtained from the experiments in section four are the primary content of section five. This section is divided into two parts. The first part reports the results of the detection engine for the sequential evaluation process. The second part reports the result of the detection engine for the load balance evaluation process. Finally, in section 6, the conclusion for this study is presented in some details.

II. Related Works

This section can be divided into two parts. First part will explore the most related exact string matching algorithms that can be used in NIDS applications. Second part will present the load balance techniques which can be used in NIDS applications.

A. Exact String Matching Algorithms

a. Boyer-Moore Algorithm

Boyer-Moore Algorithm has two heuristic phases. First: bad character shift, which starts the comparison between the pattern "P" and the text "T" from the right to the left, and in case of mismatching, the algorithm will shift forward to an "M" character (the pattern length). Second heuristic is good suffix shift, which starts a comparison from the right to the left, and in case of matching, then the algorithm move to the next character in the text "T" with the next character in the pattern "P", until get matching with all string characters; but in case of mismatching, the algorithm is move to the next occurrence that was matched before [5].

Boyer-Moore algorithm suffers from two issues: First: table skips function, which is very complicated and is used only in the case of a short text. Second, Boyer-Moore algorithm depends on the text information and rarely refers to the pattern information [6]. Usually, all the exact strings matching algorithm will determine the number of shifting according to the characters in the pattern, because the pattern is usually shorter than the

text. Therefore, [7] modified Boyer-Moore algorithm to make the searching process working faster.

b. Horspool Algorithm

Horspool algorithm looks like Boyer-Moore algorithm but in slightly different way. The Horspool algorithm works in any order and the average number of comparisons for one text character is lower than Boyer-Moore algorithm [5]. But, the Horspool algorithm has a problem which is the maximum number of shifting depends on the length of the shortest pattern size [8].

c. Weighted Exact Matching Algorithm (WEMA)

WEMA is an exact string matching algorithm, which contains two stages in order to evaluate the matching process between the pattern "P" and the text "T". First: preparing stage, which used to create an alphabetical index matrix weight "M" that is running only once, as long as no update are available in the text "T". Second: matching stage, which can be used to find any possible matching between the pattern "P" and the text "T" [9]. Table 1 summarizes the searching phase complexity for the mentioned algorithm.

Table 1. complexity of some exact string matching algorithms

Algorithm Name	Complexity of searching phase
Boyer-Moore	A: O(mn)
Horspool	A: O(mn)
WEMA	A: O(n)

Where: A is the average case of searching

B. Load balance techniques

The Intel's communication technology laboratory parallelized the SNORT-NIDS on one and four execution cores using pthreads (POSIX) library. As shown by Fig. 1. The first attempt was to run SNORT-NIDS on all four cores, where each thread runs using the same loop [10].

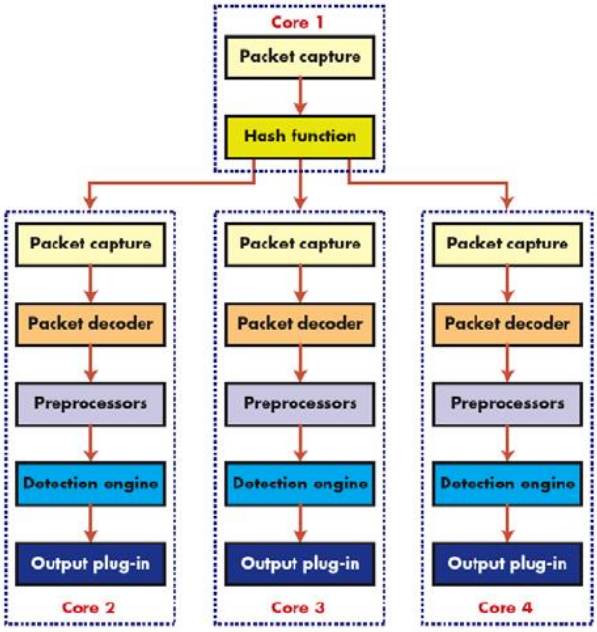

Fig. 1. Snort running on four execution cores in parallel without pipelining

Because of the synchronization between the threads, the performance of the system was very low. To avoid this problem, Intel's communication technology laboratory used a pipelining and flow-pinning technology [10]. The Pipelining and flow-pinning technology are used to divide the applications into successive stages, and assigns these stages to the implementation of the dedicated units, which extends the application to the next phase, (Fig. 2).

On the other hand, [11] designed a multi-designs for a multi-threaded NIDS. The most important design was to parallelize the signature matching process. The incoming packets will be saved in a matching queue, and each incoming packet will be dispatched to the signature matching in a parallel, where every signature matching contains its own rule sets. Fig. 3 depicts a parallel signature machine for multi-threading technology as proposed by [11].

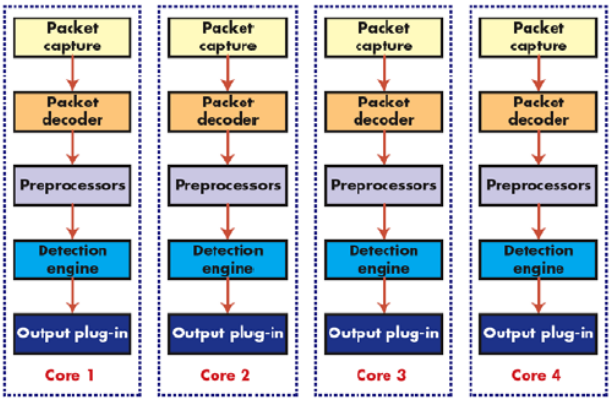

Fig. 2. Snort running on four execution cores in parallel with pipelining

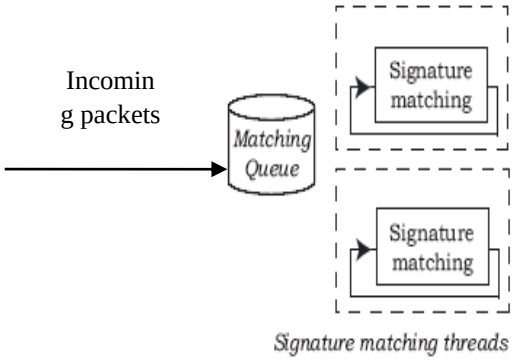

Fig. 3. Parallel signature matching [11]

III. Proposed Methodology

This section presents the design, goal, and methodology of the proposed new platform NIDS based on WEMA. This platform aim to enhance the speed of the packet payload detection engine in NIDS both in sequential and in parallel mode. The proposed novel platform NIDS based on WEMA can detect the intruders trying to gain access into the network using packet payload information. On the Other hand, the proposed platform is able to run on a single core and multi-cores processor to show that the idea could cope up with the

traffic arrival speed and various bandwidth demands. Fig. 4 depicts the entire NIDS platform.

As shown in Fig. 4, the capture engine captures all the packets from the network as the first step. The second step is the preprocess engine, which is used to prepare the captured packets to be passed to the matching engine. Matching engine, checks if the incoming packet payload match with any rule of the payload rule set by using WEMA exact matching algorithm.

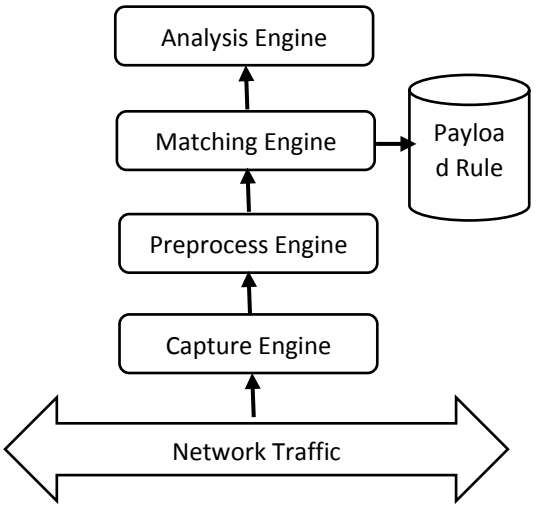

Fig. 4. The entire NIDS platform.

A. Sequential matching process

In general, existing packets payload detection engine uses any exact string matching algorithm to detect the intrusions. These string matching algorithm deal with decimal, hexadecimal, and character values. The proposed new platform NIDS based on WEMA deals with the weight values, and the payload rule sets are converted in "N" weights. Fig. 5 depicts the flowchart diagram of sequential matching process.

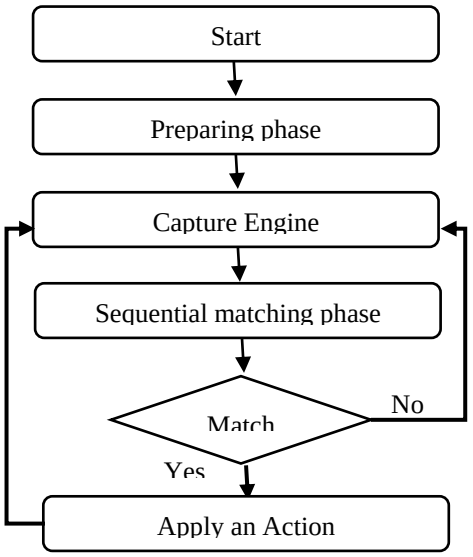

Fig. 5. Flowchart diagram of sequential matching phase

The new proposed platform NIDS based on WEMA applies a different technique to match the payload rule set with the incoming packets payload. This technique works as follows:

(1) Preparing phase: this phase will run only once, as long as no updates are available in the rule set; and also, this phase will create an alphabetical index matrix weight "W" of the payload rule set, which defines about 3000 rules based on SNORT-NIDS rule set. Each character has its own position in the text "T"(indices). Once the rule set has been created, the sequential matching phase will start.

(2) Sequential Matching phase: this phase can be used to find the exact matching between the incoming packet payload "P" and the payload rule set "R.S" as follows:

A. Create array list "L" for each incoming packet payload.

B. Determine the minimum character weight of pattern "P", which refers to the minimum number of occurrences of each character in matrix "M". If the minimum character weight is equal to zero, then stop the matching process, because the pattern "P" does not exist in the text "T". Otherwise, go to step c.

C. Create the attempt matching process of the array list "L" by adding the minimum weight character, (which selected in step b) under the corresponding character position in the array list "L" (index [i]).

D. Compare the next and the previous characters in the pattern "P", which its indices are: index [i+1] and index [i-1], with the corresponding characters of matrix "M", if both exist, then continue matching with index [i+2], and index [i-2] until reaching the end of the pattern "P" or until getting an exact matching.

E. If mismatch has occurred, then read the next occurrence of the minimum character weight of the pattern "P", and repeat from step c.

B. Load balance matching phase

The current NIDS applications are inadequate to detect an intrusion in real time. As such, the main motivation of this paper is to take advantage of the hardware revolution to enhance the speed of the NIDS based on deep inspection of the packet payload. This enhancement should also be able to utilize WEMA to identify the existence of any intrusion within the packet payload. Thus, the new proposed platform NIDS based on WEMA is designed to filter, and load balance the entire incoming packets payload among an optimal number of cores within each processor by utilizing the pthread techniques.

Based on the test bed hardware architecture, the total number of cores in the machine is calculated and create an optimal number of threads accordingly. Fig. 6 depicts the flowchart diagram of load balance matching phase.

From Fig. 6, all of the components works in a sequential process, except the dispatch component. The dispatch component will load balance the incoming packets payload into determined cores, where each core create one thread in the first attempt as an optimal

number of thread on each core. Thus, the dispatch component works as follows:

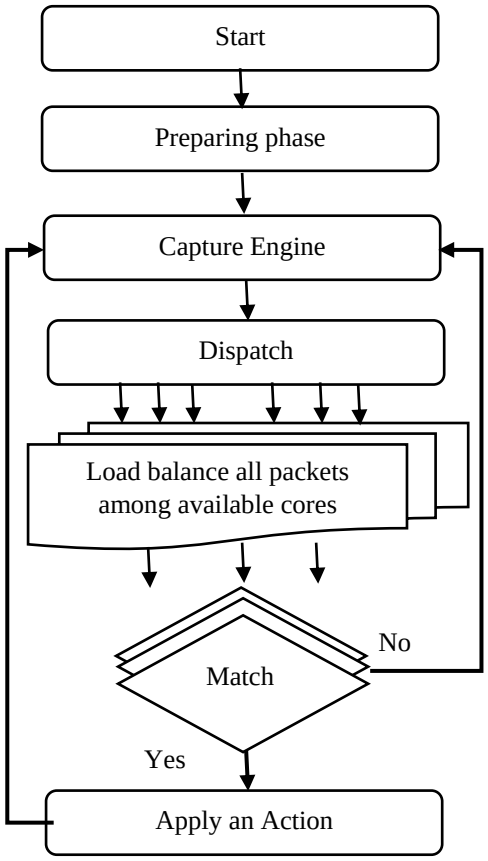

Fig. 6. Flowchart diagram of load balance matching phase.

a. Identify the number of cores (n cores). In this algorithm, the number of cores should be 1, 2, 3, or 12, because the hardware that carried the program has 12 cores.
b. Create and manage the threads (threads = m: the number of minimum character occurrences in the matrix "M").
c. Start the searching process in parallel as follows:
   1. Each core is equipped with a detection engine (WEMA) as well as the entire matrix "M". This is achieved by utilizing the multi-cores techniques.
   2. Each character of the minimum character occurrences in the matrix "M" will dispatch among available cores with a maximum of 12 cores (12 character at a time).
   3. Each core will create two internal threads to perform the parallel search as follows:
      1) Thread number 1, match the index [i+1] and thread number 2, match the index [i-1]. This step runs in all cores in parallel. In case of matching, then match the index [i+2], and index [i-2], until getting match, or get mismatch. Fig. 7 depicts the parallel search in all cores simultaneously.
      2) All threads are independent of each other, and if any one of the threads finds matching, the other threads will stop searching process accordingly.
      3) Read another new incoming packet payload.

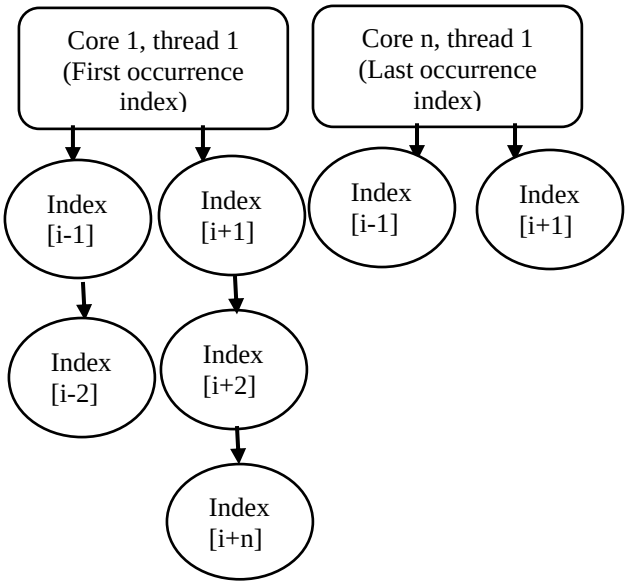

Fig. 7. The parallel search in all cores simultaneously

IV. Design and Implementation

This section explains the design and implementation of the proposed a new platform NIDS based on WEMA in both sequential matching process, and load balance matching process. For sequential matching process, the incoming pack payload, and the payload rule set as an example represents as shown in Table 2 and Table 3 respectively.

Table 2. Example of an incoming packet payload

abcd

Table 3. Example of payload rule set

payload rule sets
aabbdeabdebsabsd/bbaddaeessacceeaabcdddaaaaabes/…
jhfdahjfhfhdhfajhkriequriueqrueqdddjdksjdksjdad

A. Implementation of sequential matching process

The implementation of the sequential matching process has the following steps:

1- Convert the payload rule set into weight. Table 4 depicts the weighted matrix "M" for the first row in the rule set.
2- Create the array list "L" for the incoming packet payload as shown in Table 5.
3- Determine the character in the incoming packet payload, which has the minimum number of occurrences.

Based on matrix "M". The minimum character occurrences in the incoming packet payload is "c" (a=14, b=9, c=3, and d=8).

4- Create the first attempt of the searching process by selecting the first occurrence of the character "c", which is 29, under the corresponding character in the array list "L". Show Table 6 (first attempt).

5- Create the second attempt of the searching process by checking the index [i-1], and index [i+1] in the array list "L".

Table 4. Weighted matching matrix "M"

Alpha. Char.	Indices of the rule set characters								
	1	2	3	4	5	6	7	…	n
a(14)	1	2	7	13	20	23	28		44
b(9)	3	4	8	11	14	18	19		45
c(3)	29	30	36						
d(8)	5	9	16	21	22	37	38		
e(7)	6	10	24	25	31	32	46		
s(4)	12	26	27	47					

Table 5. Array list "L"

	Incoming packet payload = abcd			
Index	1	2	3	4
Array list "L"	a	b	c	d

6- Reference to the matrix "M", "b" does not exist in the index [i-1]. See Table 6 (second attempt). Therefore, no match can get in this attempt, and then we will examine the next occurrence of the character "c", which is 30. See Table 6 (third attempt). Also no match can get in this attempt. Thus, examine the last occurrence which is 36 as shown by Table 6 (fourth attempt).

7- As shown by Table 6 (fourth attempt), we can get an exact matching between the incoming packet payload and one of the payload rule set (full sequence: 34, 35, 36, and 37).

Table 6. Attempts of the searching process

	Incoming packet payload = abcd			
Index	1	2	3	4
Array list "L"	a	b	c	d
First attempt			29	
Second attempt		b ≠ 28	29	?
Third attempt		b ≠ 29	30	?
Fourth attempt	a=34	b = 35	36	d=37

B. Implementation of load balance matching process

To enhance the overall performance of the searching and matching process, the proposed NIDS based on WEMA runs on a multi-threading technology. Pthread techniques is used to load balance the incoming packet payload among available cores.

As we have mentioned, all of the components works in a sequential process, except the dispatch component. For instance, to apply the multi-threading technology on the previous incoming packet payload, the following steps are incorporated:

1- Dispatch the weighted matrix "M", which depicts in Table 4 among needed cores.

2- Determine the character in the incoming packet payload, which has the minimum number of occurrences based on matrix "M".

3- In each core, create the array list "L".

4- Create the first attempt of the load balance matching process on each core, where each core will select one different index of the minimum character occurrences as shown in Table 7.

5- In each core, two internal threads will be created, thread number 1 check matching with index [i-1], and thread number 2 check matching with index [i+1] in parallel. See Table 7 second attempt.

As shown in Table 7, an exact matching has been occurred after only one attempt comparing with sequential matching process, which needed four attempts.

V. Results

The proposed NIDS based on WEMA is evaluated in a sequential and in load balance manner. Both of these evaluations are explained in details as follows:

A. Sequential evaluation process

Based on the hardware architecture, we read 450 to 10000 packets from the file. The payload rule sets had a size of 50 KB. In order to evaluate the effectiveness of the proposed NIDS based on WEMA, a comparison is made with Boyer-Moore algorithm. The comparison results can be viewed in Fig. 8.

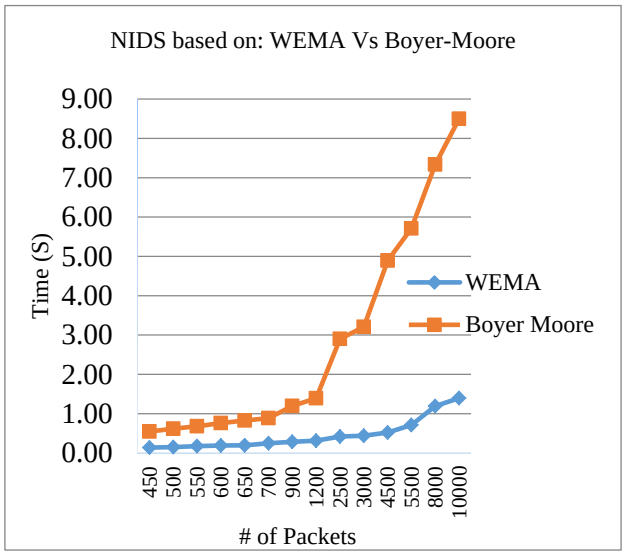

Fig. 8. NIDS based on WEMA Vs Boyer-Moore algorithm

From Fig. 8, it can be seen that the performance of the proposed NIDS based on WEMA and Boyer-Moore algorithm are very close for the first 1200 packets. After this point, it seems that the proposed NIDS based on WEMA has a little bit advantage than Boyer-Moore algorithm. However, it is clear that the proposed NIDS based on WEMA is faster after 1200 packets as compared to Boyer-Moore algorithm. This is a very encouraging enhancement. The achieved improvement was between 74% - 87%.

Table 7. Attempts of load balance matching process

	Incoming packet payload = abcd											
	Core 1				Core 2				Core 3			
	Thread 1				Thread 1				Thread 1			
Index	1	2	3	4	1	2	3	4	1	2	3	4
Array list "L"	a	b	c	d	a	b	c	d	a	b	c	D
First attempt			29				30				36	
Second attempt		b≠28		d≠30		b≠29		d≠31	a=34	b=35	36	d=37

B. Load balance evaluation process

We parallelized the proposed NIDS based on WEMA within the same environment by using the Pthread techniques. 450 to 10000 packets were read from the file. The payload rule set was 50 KB size. The result of the parallelized NIDS based on WEMA is shown in Fig. 9.

As depicted by Fig. 9, the performance of the first 900 packets is low, because in some cases there will be no need to use the load balance process since the number of the packets is small (the process of distributing the packets over the threads needs time, which lead to slowing the load balance process). When the number of packets increased from 1200 packets to 10000 packets it can be observed that the load balance process had been enhanced from 1% to 47%.

In addition, the efficiency, overhead, and speedup of the proposed NIDS based on WEMA of the load balance evaluation process is in Fig. 10.

As shown by Fig. 10, the efficiency and overhead of the load balance evaluation process are acceptable and very close of each other. Concerning the speedup issue, the speedup increase whenever the number of packets increases.

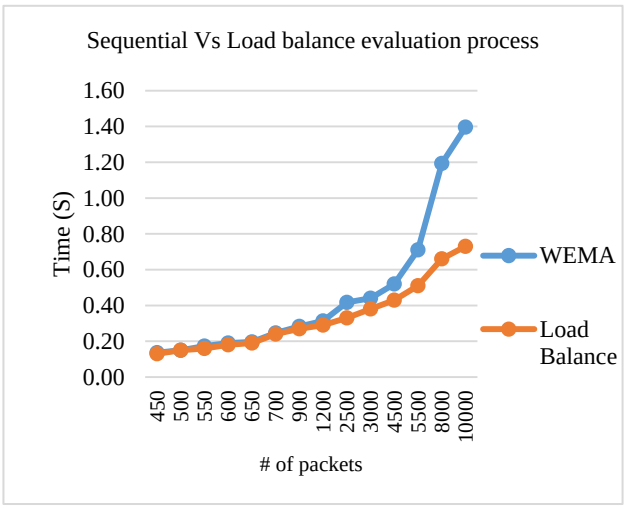

Fig. 9. Sequential Vs load balance evaluation process

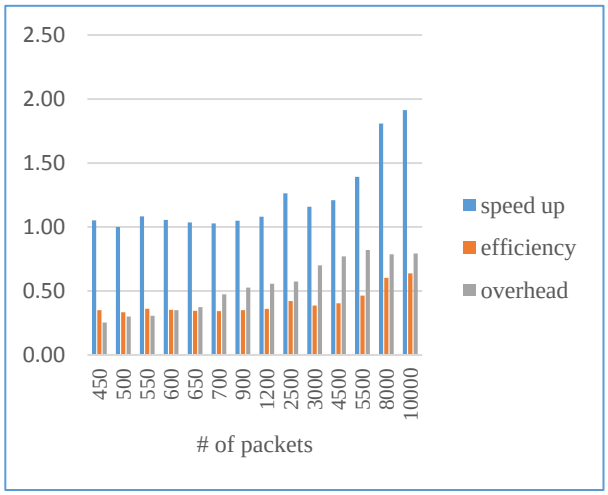

Fig. 10. Efficiency, overhead, and speedup of the load balanced scenario

VI. Conclusion

The performance evaluation of the proposed NIDS based on WEMA has been improved comparing with the traditional NIDS based on Boyer-Moore algorithm. The evaluation shows promising results in terms of processing speed. Furthermore, it shows the ability of the proposed NIDS based on WEMA to cope with large size file with minimal required time processing.

On the other hand, we have fulfilled the objective in enhancing the speed of detection engine in both sequential and in load balance modes using multi-threaded technology in a complete and scalable manner.

Acknowledgement

I would like to thank Al-zaytoonah University of Jordan for supporting this research paper.

References

[1] Stallings, w. (2006). Cryptography and Network Security

[2] Muhammad Abedin, S. N. (2006), "Detection and Resolution of Anomalies in Firewall Policy Rules, "IFIP International Federation for Information Processing (pp. 15-29). E. Damiani and P. Liu (Eds.): Data and Applications Security 2006, LNCS 4127,.

[3] Li, W. (2004), "Using Genetic Algorithm for Network Intrusion Detection, " In Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference (pp. 1-9). Mississippi : Mississippi State University, Mississippi State, MS 39762.

[4] Sutapa Sarkar, and Brindha.M (2014), "High Performance Network Security Using NIDS Approach, " I.J. Information Technology and Computer Science, 2014,07, 47-55 Published Online June 2014 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2014.07.07

[5] Lecroq, C. C. (2004), "Handbook of Exact String Matching Algorithm," King's College Publications ISBN:0954300645.
[6] Zhou, Y. X. (2008), "Using Multi-core Processors to Support Network Security Applications, " 12th IEEE International Workshop on Future Trends of Distributed Computing Systems (pp. 213-218). IEEE-Computer Society.
[7] Rafiq, A. E.-K. (2004), "A fast String Search Algorithm for Deep Packet Classification," ELSEVIER-SCIENCE DIRECT , 1524-1538.
[8] Rong-Tai Liu, N.-F. H.-N.-H.-C. (2004), "A Fast Pattern-Match Engine for Network Processor-based Network Intrusion Detection Systems," Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) (pp. 1-5). IEEE Computer Society.
[9] Abdallah A. Hlayel, Adnan A. Hnaif, "A New Exact Pattern Matching Algorithm (WEMA), "Journal of Applied Science, 13(24), 2013. ISSN 1812-5654 / DOI:10.3923/jas. Vol 14. 2013
[10] Edwin Verplanke, E. (2007), "Understand Packet Processing With Multi-core Processors, ". EETimes-indea, April 2007, pp.1-5.
[11] Bart Haagdorens1, T. V. (2004), "Improving the Performance of Signature-Based Network Intrusion Detection Sensors by Multi-threading, "Springer-Verlag Berlin Heidelberg , 188–203.

Author's Profiles

Dr. Adnan Hnaif is an Assistance professor at the computer networks department, Faculty of Science and information technology, Al Zaytoonah Private University of Jordan. Dr. Hnaif received his PhD degree in Computer Science from University Sains Malaysia – National Advanced IPv6 Centre and Excellence (NAV6) in 2010. He received his MSc degree of Computer Science in 2003, and obtained his Bachelor degree of Computer Science in 1999/2000. His researches focus on the network security, exact matching algorithms, and parallel processing. E-mail: adnan_hnaif@zuj.edu.jo

