Applying an Efficient Searching Algorithm for Intrusion Detection on Ubicom Network Processor

Qutaiba Ibrahim and Sahar Lazim
Computer Engineering Department, University of Mosul, Iraq

Abstract: *Intrusion Detection Systems (IDSs) have become widely recognized as powerful tools for identifying, deterring and deflecting malicious attacks over the network. Essential to almost every intrusion detection system is the ability to search through packets and identify content that matches known attacks. In this paper, common searching algorithms (string matching, Native, Boyer Moore and pattern matching algorithms) are examined on Ubicom Network Processor which is intended to be used as Network Intrusion Detection System (NIDS). Afterword, the suitable algorithm for Ubicom network processor is chosen which combine string matching and Native algorithms because these algorithms don't have any type of preprocessing as Ubicom network processor doesn't contain Micro Engine (ME) and doesn't support multithreading which are used to speed the operation of preprocessing.*

Keywords: *Intrusion Detection System, Network Processor, Searching Algorithm*

Received September 9, 2009; Accepted June 28, 2010

1. Introduction

Intrusion Detection Systems (IDSs) are one of the most useful tools to identifying malicious attempts over the network and protecting the systems without modifying the end-user software. Different from firewalls that only check specified fields of the packet *headers*, IDSs detect the malicious information in the *payloads*. An IDS typically contains a database that describes the signatures of malicious behavior. The number of patterns is generally a few thousands and still increasing. The signatures may appear *anywhere* in any packet *payload*. Therefore, IDSs must be capable of in-depth packet inspection even when suffering serious attacks; otherwise the protectorate will not be defended strictly [12].

The core of any intrusion detection system is the string matching algorithm. String matching is the main task in intrusion detection. From a stream of packets, the algorithm identifies those packets that contain data matching the signatures of a known attack. The intrusion detection system then takes action that could vary from alerting the system administrator to dropping the packet in the case of inline IDS. The problem of pattern matching is well investigated, many algorithms exist and they can be classified as either single pattern string matching or multiple pattern string matching. In single pattern string matching the packet is searched for a single string at a time. On the other hand, multiple pattern string matching searches the packet for the set of strings all at once. This paper focuses on single string matching [2].

Network processor (NP)-based network devices are increasing recently. In opposite to ASIC, NP has a possibility of adding of modifying functionality only by use of program update [11]. In this paper, Ubicom Network Processor is used an embedded Network Intrusion Detection System (NIDS). SNORT IDS rules were translated to suite Ubicom platform.

The remainder of this paper is organized as follows: first section 2 displays the related works. Section 3 introduces the overview of Ubicom Network processor from point of view hardware and software based. Section 4 demonstrates the single string searching algorithms that will be examined in this paper. Section 5 presents the results of the comparison between the searching algorithms which are remembered in the previous section. Finally, section 6 discusses the conclusions of the results.

2. Related Works

Researches in tools and methodologies for network processors have focused mainly on modularity, re-usability and ease of programming. The researches in this section show how intrusion detection can be performed on a network processor:

In 2003, I. Charitakis et al. presented a software architecture that enables the use of the Intel IXP1200 network processor in packet header analysis for network intrusion detection. The proposed work consists of a simple and efficient run-time infrastructure for managing network processor resources, along with the S2I compiler, a tool that generates efficient C code from high-level, human readable, intrusion signatures. Therefore implementing intrusion analysis on the IXP1200 becomes a process

that does not require knowledge of architecture internals and the micro-C programming language [6].

In 2004, Herbert Bos et al. described a network intrusion detection system implemented on the IXP1200 network processor. It was aimed to detect worms at high speed by matching the payload of network packets against worm signatures at the lowest possible levels of the processing hierarchy (the microengines of an IXP1200 network processor). The solution employs the Aho-Corasick algorithm in a parallel fashion, where each microengine processes a subset of the network traffic. To allow for large patterns as well as a large number of rules, the signatures are stored in off-chip memory [7].

In 2006, Robin Salim el at. discussed technique to improve packet classification through the use of Bloom Filter and hash table lookup. Because packet classification is an important function to other networking infrastructure, for instance firewall, quality of service, multimedia communication, an improved packet classification scheme could benefit application in related areas [13].

In 2007, Young-Ho Kim el at. introduced an efficient algorithm for content processor to perform multi-pattern signature matching. The proposed algorithm uses software bitmap for each multi-pattern signature without hardware changes, which maximizes flexibility of content processor. From the analysis of *SNORT* which is the widely used intrusion detection system, they observe spatial locality between distances of patterns in the multi-pattern signature. The algorithm makes use of this distance information for adaptive performance optimization. Their techniques show that content processor can be a good solution for multi-pattern processing in intrusion detection systems without hardware modification with reasonable performance [8].

Ubicom Network Processor differs from other platforms in that it has limited memory and relatively low processing power. It is typical to be used in embedded system because of its low power consumption and low cost.

3. Ubicom IP2022 Network Processor Overview

Ubicom IP2022 network processor produced by Ubicom Company, Ubicom provides the whole solution as a fully integrated platform - the RTOS (Real Time Operating System), the protocol stack, and the necessary hardware. Ubicom's IP2022 chip embeds some basic hardware, but it permits combining it with on-chip software to support the most prevalent protocols. The same device can supports Ethernet, Bluetooth wireless technology, IEEE 802.11, and so on. The key to this approach is Software SOC™ (System on Chip) technology.

Ubicom's hardware includes the following components as shown in figure1 [1].

- Designed to support single-chip networked solutions
- Fast processor core
- 64KB (32K x 16) Flash program memory
- 16KB (8K x 16) SRAM data/program memory
- 4KB (4K x 8) SRAM data memory
- Two SerDes communication blocks supporting common PHYs (Ethernet, USB, UARTs, etc.) and bridging applications
- Advanced 120MHz RISC processor
- High speed packet processing
- Instruction set optimized for communication functions
- Supports software implementation of traditional hardware functions
- In-system reprogrammable for highest flexibility
- Run time self-programmable
- Vpp = Vcc supply voltage

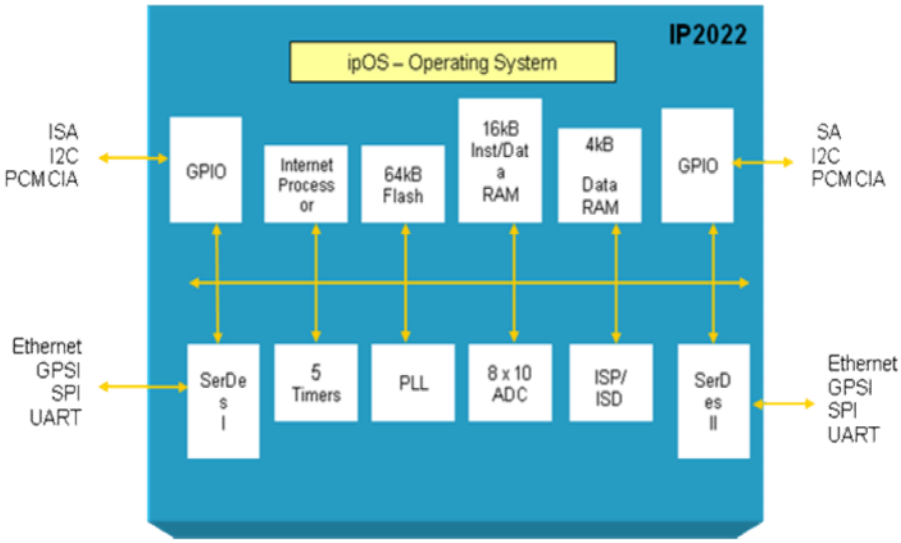

Figure 1. IP2022 block diagram.

The scheme in figure 2 appears the basic components for Ubicom's complete development environment that can separate it to four essential components [2]:

- Ubicom's Software Development Kit (SDK): Ubicom supports many software packages, like ipOS™, ipStack™, ipHAL™, ipModule™ and etc.
- Red Hat GNUPro tools which consist of GCC ANSI C compiler, Assembler, Linker, and GNU debugger.
- Ubicom's Configuration Tool Integrated tool to support rapid development efforts.
- Ubicom's Unity Integrated Development Environment (IDE) contains Editor, project manager, graphic user interface to GNU debugger, device programmer.

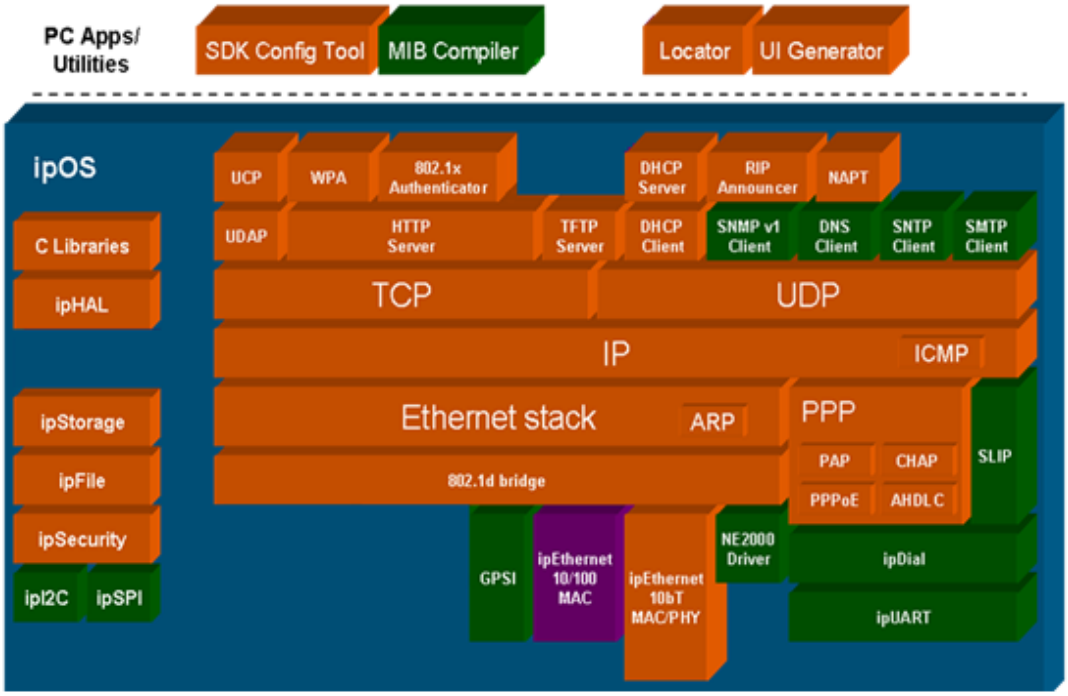

Figure 2. Ubicom's complete development environment.

4. Searching Algorithms

At the heart of almost every modern intrusion detection system is a string matching algorithm. String matching algorithm is crucial because it allows detection systems to base their actions on the content that is actually flowing to a machine. The string identifies those packets that contain data matching the pattern of known attacks. Essentially, the string matching algorithm compares the set of strings in the rule-set to the data seen in the packets that flow across the network [14].

4.1. String Matching Algorithm

In order to make Ubicom's buffer (that used to store the receiving packet) easier to use, a number of functions were provided which behave in a similar manner to standard string/memory functions provided by a standard C library [3]:

- Strstr Function:This function searches for a given string in the netbuf and returns the position (address) pointer to string, where the string was first found, or 0 if not found.
- Strchr Function: This function searches a character in the netbuf. The function returns the position (address) pointer to a character, where the character was first found, or 0 if not found.

4.2. Native Algorithm

The so-called native or brute force algorithm is the most intuitive approach to the string pattern-matching problem. This algorithm simply attempts matching the pattern in the target at successive positions from left to right. If failure occurs, it shifts the comparison window one character to the right until the end of the target is reached. The Native searching algorithm doesn't need any kind of preprocessing on the pattern and requires only a fixed amount of extra memory space.

4.3. Boyer Moore Algorithm

Boyer Moore is the most widely used algorithm for string matching in intrusion detection system, the algorithm compares the string of the input starting from the rightmost character of the string. To reduce the large number of comparison loops, two heuristics

are triggered on a mismatch. The bad character heuristic shifts the search string to align the mismatching character with the rightmost position at which the mismatching character appears in the search string. If the mismatch occurs in the middle of the search string, then there is suffix that matches. The good suffix heuristic shifts the search string to the next occurrence of the suffix in the string [4, 9], see figure 3 below.

```
1{ initialization of Badcharacter and Goodsuffix
tables is omitted }
lastch. ← patttern[m];
i ← m;
while i ⩽ stringlen do
begin
ch ← string[i];
if ch = lastch then
begin
j ← m - 1;
repeat
if j = 0 then return i;
j ← j - 1;
i ← i - 1;
until string[i] ≠ pattern[j];
i ← i + max(Badcharacter [ch] , Goodsuffix [j]);
end
else
i ← i + Badcharacter [ch];
```

Figure 3. Algorithm actions of Boyer Moore algorithm.

4.4. Pattern Matching Algorithm

The Pattern Matching Algorithm can be divided into two phases: preprocessing phase and search phase. The first task of preprocessing phase is to change each byte from signature string to two bytes and put these bytes in converting array. The second task is to generate a two dimensional array called NEXT. This Array is very important which decides how to move to a proper position in the next search. After, array NEXT is generated, its values will be invariable during searching process.

During searching phase, the comparison is performing from right to left at each check point. If a mismatching occurs, the next to the last character of current comparing window is used to execute the next matching [16], see figure 4.

5. SNORT Rules analysis and Search Algorithm Assumptions

Snort is source code open software programmed by C language, which conforms to GNU. It is a platform independent, light network intrusion detection system, and it also works as network traffic sniffer and log record tool. Snort employs network information search

```
Pattern Matching Algorithm
begin
1 for (n=0 ,k=0 to k<m  k=k+2) do
  convert[k]=left nibble of pattern[n];
  convert[k+1]=right nibble of pattern[n];
 end
2 for (i=0 to16) do
    3 for(j=0 to 16) do next[i,j]←m+1;
4 for (i=0 to 2m-2) do next[convert[i],convert[i+1]]
←m-i/2;
5 j←0;
6 while(j<= stringlen) do begin
7 i←m-1;
8 while(i>=0 and pattern[i]=text[i+j]) do i←i-1;
9 if (i<0) then output(match at location j);return;
10 if (text[j+m-2,j+m-1]=pattern[m-2,m-1]) then j
←j+1;
11 else j←j+next[left nibble of text[j+m-1],right
nibble of text[j+m-1]];
end while
end
```

Figure 4. Algorithm actions of Pattern Matching Algorithm.

technique based on rules. Pattern matching is performed on each packet to conduct intrusion detection [15]. Before the results are displayed, the *SNORT* rules analysis must be described to constrain range of patterns lengths and payload length which will be searched to find the pattern. Reference [5] illustrates that 87% of the rules contain strings to match against the packet payload. The bar chart in figure 5 shows the distribution of the string lengths in bytes. The columns represent the number of rules containing strings of certain length that is marked on the x axis. The columns are clustered around the average string length of 14 bytes. The majority of the strings are shorter than 26 bytes. While, reference [10] shows that 88.6% of all rules are satisfied by first 145 bytes of payload. This means, most of attacks' signatures can be found in first 145 bytes. Figure 6 explains the distribution of the required payload length.

Therefore, this paper proposed to constrain the maximum length of pattern to 20 bytes and the maximum length of payload to145 bytes.

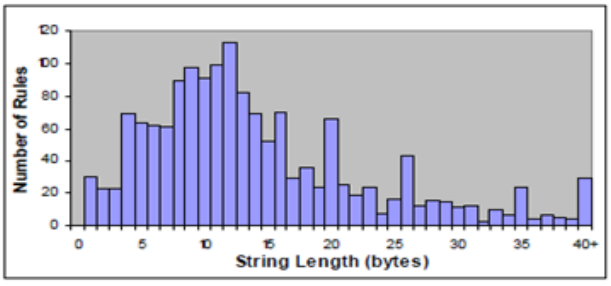

Figure 5. Distribution of the string lengths in the SNORT database.

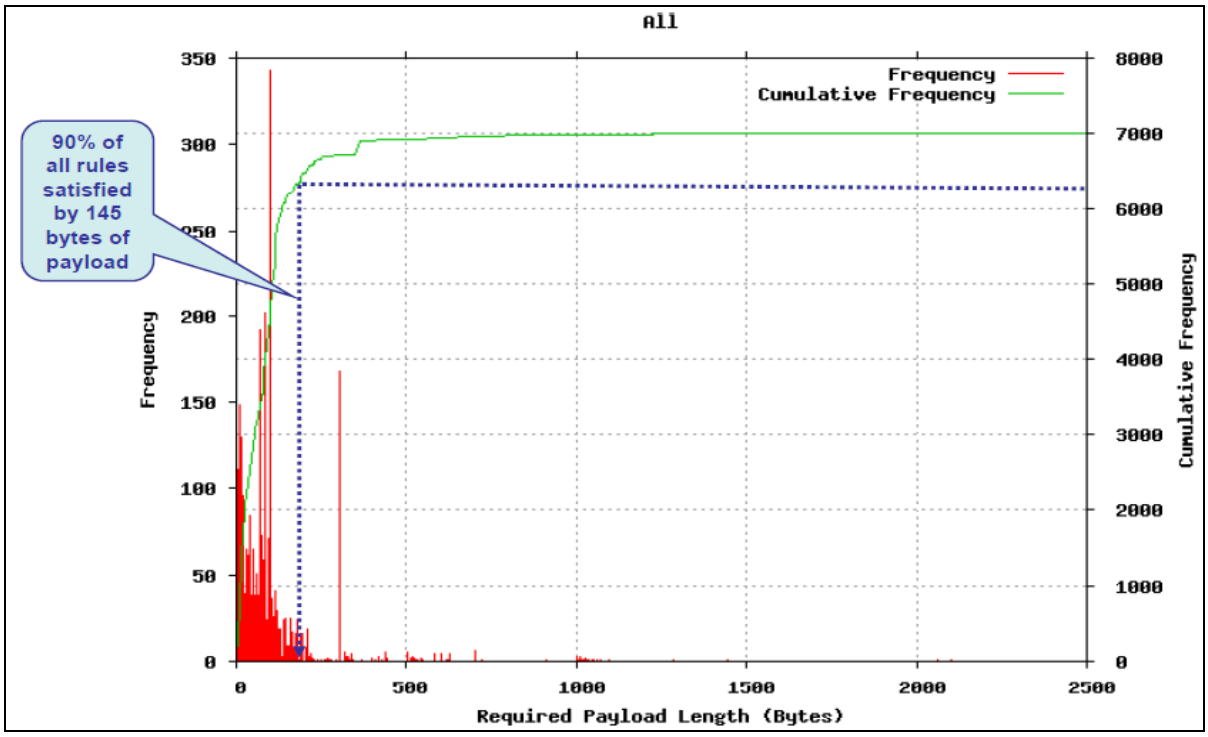

Figure 6. The distribution of the required payload length.

6. Results

The results of the algorithms that were presented in section 4 are discussed and a comparison between them through variant values of signature and payload lengths is given.

Results are discussed as two steps: In the first one, results are discussed by comparing the various algorithms at similar signature lengths. Two metrics were used to evaluate the performance of the various algorithms: Searching time is the amount of time to find a signature inside the payload, and Total searching time is the overall time for the searching algorithm, which includes the searching time and preprocessing time if existed. At second one, some matrices are specified to constrain the best algorithm to be adopted by the platform. The signature position is
assumed to be at the end of the payload at each length during computing the total searching time for the algorithms.

6.1. Algorithms Comparison using various Signature lengths:

In this section, comparisons between various algorithms were made. The range of the signature lengths is constrained between 1 to 20 bytes, and the range of the payload length is constrained between 22 to 145 bytes. as mentioned earlier in Section 5.

6.1.1. Signature Length of 1 Byte.

Figure 7 shows the results of searching phase for all algorithms. this figure shows that the string algorithm (using the strchr function which is specified for one character) has the minimum searching time. But an increased searching time of string function (using strstr function) is observed, because it checks the byte which is read from Ubicom's buffer if it is NULL or not. The cause for the relatively long searching time of Boyer Moore algorithm is this algorithm works in backward direction. This means, it reads the values of payload from netbuf to find signature from right to left on opposite to the native and strstr function which read forwarding from left to right. The pattern matching algorithm is better than Boyer Moore because its shifted value is larger than that of the last one, the shifted value in the pattern matching algorithm is two byte, while the shifted value in the Boyer Moore algorithm is one byte.

Figure 8 shows the Total searching time of the four algorithms. The String Matching (using strstr and strchr functions) and Native Algorithms weren't affected by the preprocessing time because they didn't have any type of preprocessing. Pattern matching and Boyer Moore algorithms have preprocessing time, their values can be extracted from Table 1. Although, the preprocessing time of pattern matching algorithm is higher than that of Boyer Moore, the last one has higher total searching time. Also, Table 1 show the first hit of the four algorithms.

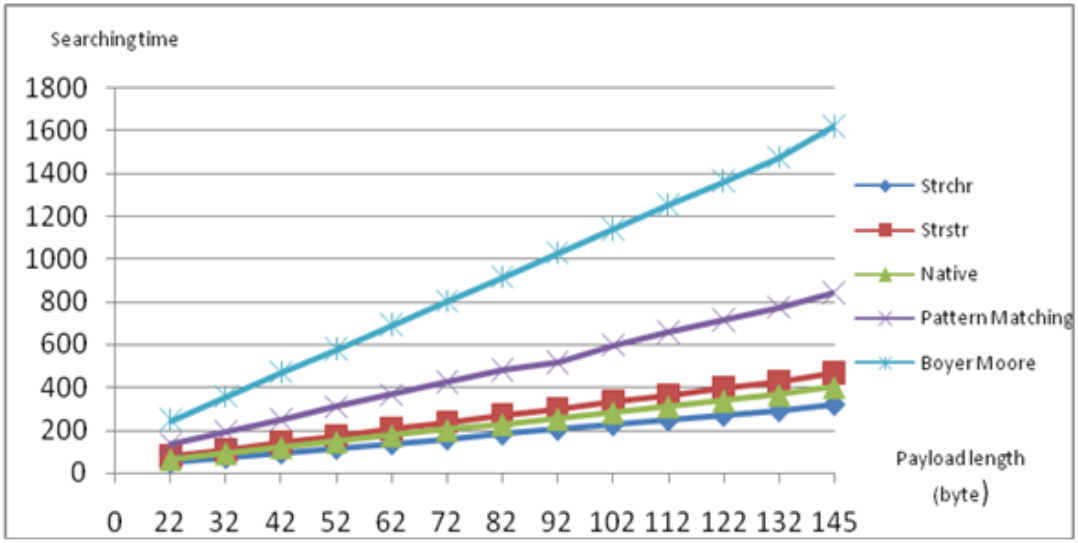

Figure 7. Variation of searching time for 1 Byte signature length.

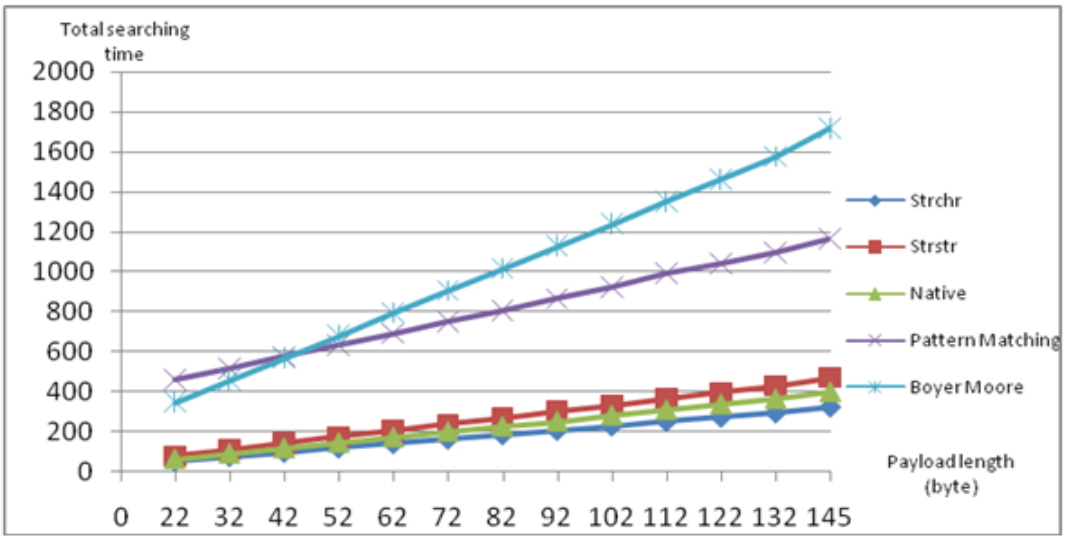

Figure 8. Variation of total searching time for 1 Byte signature length.

Table 1. The first hit of the four algorithms.

Algorithm	String Matching	Native	Boyer Moore		Pattern Matching	
Signature length (byte)	Total Searching Time (µsec)	Total Searching Time (µsec)	Searching Time (µsec)	Total Searching Time (µsec)	Searching Time (µsec)	Total Searching Time (µsec)
1	8.898	4	8.2588	109	8.837	330.83
5	19.133	10	21.176	169	22.312	368.12
10	31.211	17.5	35	247	39.941	414.22
15	45	25	52.087	324	57.062	460.92
20	57	33	67.031	400.13	74.775	508.42

6.1.2. Signature Length of 20 Bytes.

The signature length is increased to 20 bytes, the searching time for the four algorithms are shown in Figure 9. Pattern matching and Boyer Moore algorithms have the minimum searching time because their shifted values are affected by increasing signature length. The difference between them is very small. String and Native algorithms weren't affected by the increased signature length because their shifted values are always one byte.

Figure 10 shows the total searching time of the four algorithms. Native algorithm is the best one against other algorithm because it has the minimum total searching time in all cases. The Pattern Matching algorithm is the worst in this case because it has very large preprocessing time, and this time is affected by increased signature length. The difference of preprocessing time between Boyer Moore and Pattern Matching algorithms is constant and nearly smaller than the difference in the previous case because Boyer Moore algorithm is affected by the increased signature length.

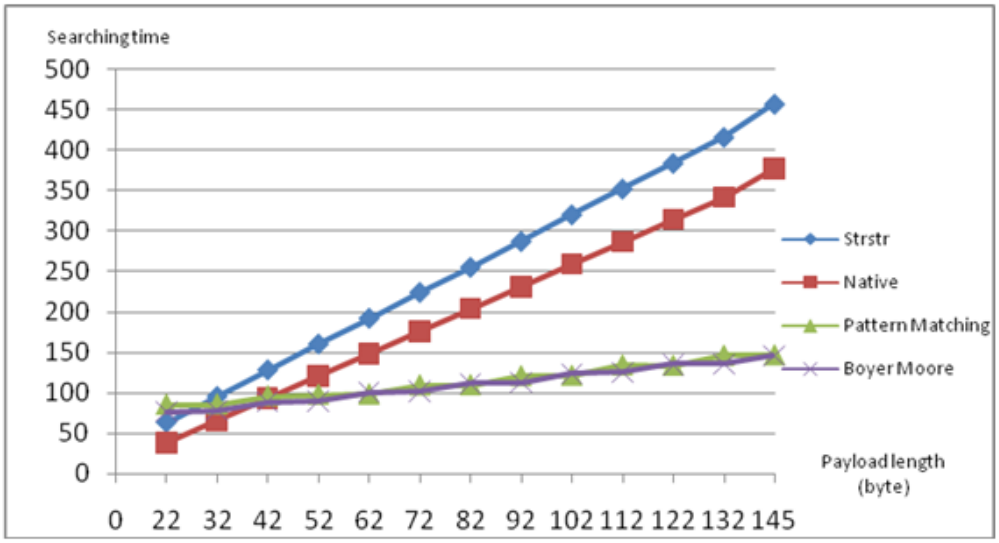

Figure 9. Variation of searching time for 20 Bytes signature length.

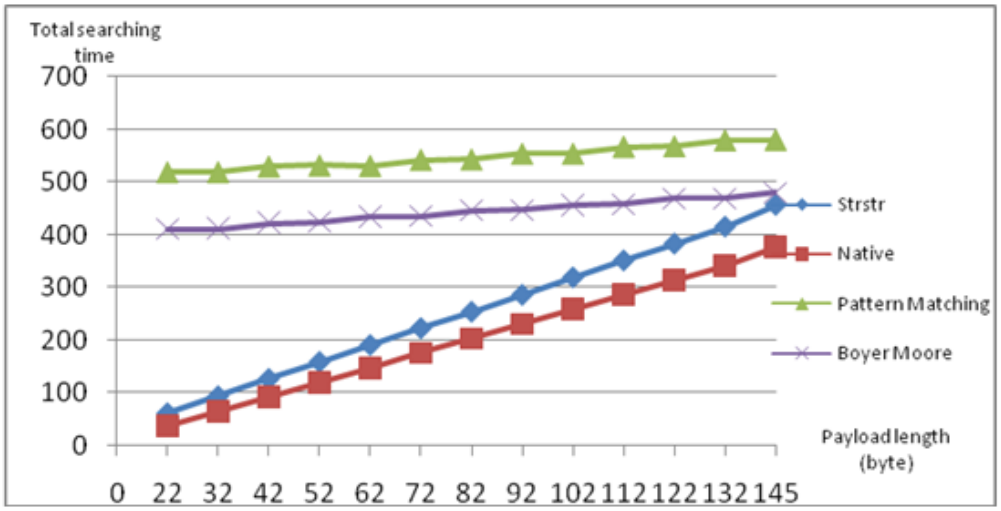

Figure 10. Variation of total searching time for 20 Bytes signature length.

6.2. Suggesting a New Searching Algorithm

In order to summarized the comparison results, searching algorithms features were compound as listed on Table 2.

According to Ubicom platform criteria, native algorithm is candidate to be the best searching algorithm. Also, string algorithm using strchr function has the higher performance when searching for (1 byte) signature. In this paper, the adopted searching algorithm is a combination of the above two algorithms. The use of strchr function is suggested in the case of (1 byte) signature. When searching for a higher signature value, a switching procedure is made to the operation

of the native algorithm. The performance of the suggested algorithm can be seen in figure 11.

7. Conclusion

This paper deals with investigating a set of searching algorithms intended to be used on an embedded NIDS platform. Ubicom Network Processor supplied with SNORT IDS rules was examined against a set of common searching algorithms (string Matching, Native, Boyer Moore and Pattern Matching algorithms). It was found that combining both String Matching and Native algorthims was the optimum solution to build a high response embedded NIDS.

Table 2. Comparison algorithms based on metrics.

Algorithms				
Metrics	String	Native	Pattern Matching	Boyer Moore
Searching time	High	High	Low	Low
Check for NULL	yes	No	No	No
Space of memory	Low	Low	High	High
Preprocessing time	*N/A	N/A	High	Medium
Direction of comparison	Left to right	Left to right	Right to left	Right to left
Shifted value	One byte	One byte	Variant	Variant

*N/A: not available

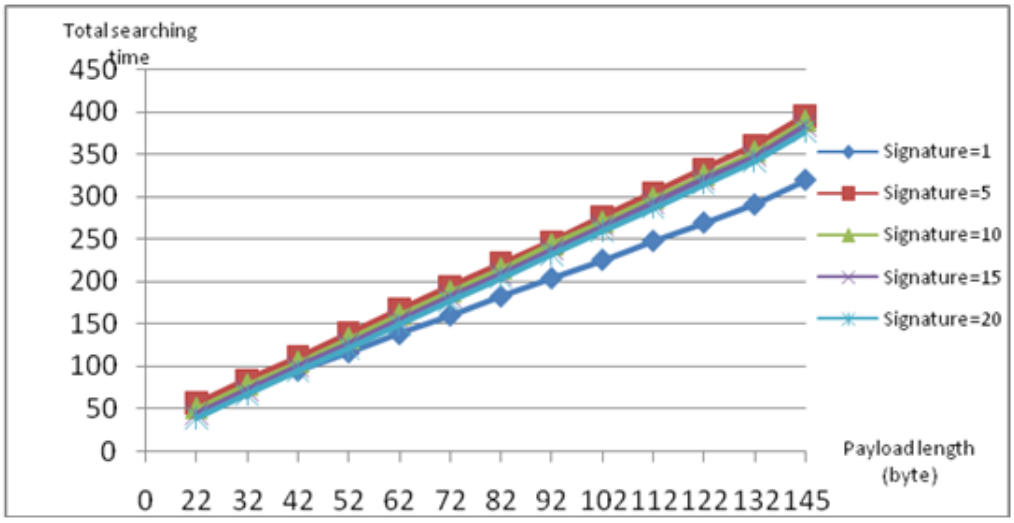

Figure11. The performance of the Suggested Algorithm.

References

[1] "IP2022 Wireless Network Processor Features and Performance Optimized for Network Connectivity IP2022 Data Sheet", UBICOM, Inc., 22 Jan. 2009, Web Site: http//www.ubicom.com.

[2] "IP2022 Internet Processor Product Brief", UBICOM, Inc., 2001, Web Site: http//www.ubicom.com.

[3] "IP3000/IP2000 Family Software Development Kit Reference Manual", UBICOM, Inc., 28 June 2005, Web Site: http//www.ubicom.com.

[4] "Boyer-Moore algorithm", Web Site: http://www-igm.univ-mlv.fr/~lecroq/string/node14.html.

[5] Aldwairi M., "Hardware-Efficient Pattern Matching Algorithm And Architectures For Fast Intrusion Detection", Dissertation, Computer Engineering Dept., North Carolina State University, 2006.

[6] Charitakis I., Pnevmatikatos D., Markatos E., Anagnostakis K., "Code Generation for Packet Header Intrusion Analysis on the IXP1200 Network Processor", Appears in 7th International Workshop on Software and Compilers for Embedded Systems, Vienna, Austria, Sep. 2003.

[7] Herbert B., Kaiming H., "A network intrusion detection system on IXP1200 network processors with support for large rule sets", Technical Report, Leiden University, Netherlands, 2004.

[8] Kim Y., Jung B., Lim J., Kim K., "Processing of Multi-pattern Signature in Intrusion Detection System with Content Processor", The 6th International Conference on Information,

Communications & Signal Processing, 10-13 Dec. 2007.

[9] Matyas A., Moore J., "String Searching over Small Alphabets", Technical Report, Department of Computer Sciences University of Texas, Austin, 11 Dec. 2007.

[10] Münz G., Weber N., Carle G., "Signature Detection in Sampled Packets", The 2nd Workshop on Monitoring, Attack Detection and Mitigation, Toulouse, France, 5-6 Nov. 2007.

[11] Ryu S., Chung B., Kim K., "Incorporating Intrusion Detection Functionality into 1XP2800 Network Processor based Router", Electronics Telecommunication Research Institute, 20-22 Feb. 2006.

[12] Sheu T., Huang N., Lee H., "A Time- and Memory- Efficient String Matching Algorithm for Intrusion Detection Systems", Global Telecommunications Conference, 1 Dec. 2006.

[13] Salim R., Radha G., "Software-based Packet Classification in Network Intrusion Detection System using Network Processor", IEEE Region 10 Conference on TENCON, 14-17 Nov. 2006.

[14] Tuck N., Sherwood T., Calder B., Varghese G., "Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection", Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, VOL.4, PP.2628 –2639, 7-11 March 2004.

[15] Wang Y., Kobayashi H., "High Performance Pattern Matching Algorithm for Network Security", International Journal of Computer Science and Network Security, VOL.6 No.10, PP.83–87, Oct. 2006.

[16] Zhao K., Chu J., Che X., Lin L., Liang H., "Improvement on Rules Matching Algorithm of Snort Based on Dynamic Adjustment", The 2nd International Conference on Anti-counterfeiting, Security and Identification, 20-23 Aug. 2008.

Qutaiba Ali was born in Mosul, Iraq, on October ,1974. He received the B.S. and M.S. degrees from the Department of Electrical Engineering, University of Mosul, Iraq, in 1996 and 1999, respectively. He received his Ph.D. degree (with honor) from the Computer Engineering Department, University of Mosul, Iraq, in 2006. Since 2000, he has been with the Department of Computer Engineering, Mosul University, Mosul, Iraq, where he is currently an assistance professor. His research interests include computer networks analysis and design, embedded network devices and network security. Dr. Ali instructed many topics (for Post and Undergraduate stage) in computer engineering field during the last ten years and has many publications in numerous journals and conferences. He acquires many awards (form National Instrument INC.) and appreciations form different parties for excellent teaching and extra scientific research efforts. Also, he was invited to join many respectable scientific organizations such as IEEE, IENG ASTF, WASET and many others. He was participate (as technical committee member) in ten IEEE conferences in USA, Malaysia, China, South Korea and Egypt and joined the editorial board of five scientific international journals.

Sahar Lazim was born in Mosul, Iraq, on 1985. She received the B.S. & MS degree from the Department of Computer Engineering University of Mosul, Iraq, in 2007 and 2009 respectively. Her research interests include Network security & Network processor architecture.