International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 16 (2015) pp 36658-36662

A Fast and Scalable Pattern Matching Scheme for NIDS Using Z Algorithm

Dhanesh P
Department of ECE Amrita Vishwa Vidyapeetham University
Amritanagar, Coimbatore, India
e-mail: dhaneshpgopinath@gmail.com

D. S. Harish Ram
Department of ECE Amrita Vishwa Vidyapeetham University
Amritanagar, Coimbatore, India
e-mail: ds_harishram@cb.amrita.edu

Abstract
Network Intrusion Detection Systems (NIDS) have become integral to today's computer networks as the information transferred through the network is highly vulnerable to cyber-attacks. The implementation of the same involves an efficient string matching algorithm which will compare certain parameters and features of the received packets depending on the protocols used for communication. As the system incorporates a huge number of pattern comparisons, the string matching algorithm stands as the heart of entire NIDS system. Software implementations such as SNORT are inadequate for Ethernet backbones with bit rates of hundreds of Gigabits where high throughputs are required. Small improvements at the algorithmic level can boost the performance of the system drastically. In order to handle the large payloads of current network packets, simple and commonly used algorithms such as Aho-Corasick lack performance. In this work, the Z algorithm which is generally used for finding large subsequences in DNA strands is adapted for string matching for intrusion detection. This new algorithm shows noticeable improvements in performance and scalability.

Keywords: Aho Corasick, Boyer Moore, HIDS, IDS, KMP, NIDS, Z Algorithm

I. INTRODUCTION

The network intrusion detection system (NIDS) [1, 5] is a security system which governs the inbound and outbound activities of a network. It is expected to identify suspicious patterns that may point out some intrusion or policy violation in the network or system. The idea of Intrusion Detection is mainly implemented by the pattern classification and pattern matching algorithms. The working of IDS is same as that of the virus detection software or Antivirus software, where a database of defined attacks have been formulated, maintained and updated regularly on a periodic basis. The data or packet of data is identified from a stream of network traffic and different fields in the packet are analyzed according to the rules maintained in the database. Once a particular rule is recognized then that can be modeled as an intrusion.

There are some basic steps involved in Intrusion Detection such as identifying attacks, alert different intrusions, take decisions accordingly for active systems and store the details of attack or event as and when they are detected. The real time performance needed by the rule identification system depends clearly on the efficiency of the pattern matching algorithm used. So performance of the system and efficiency of the algorithms shares a direct relationship between each other. The selection of algorithm has a crucial role in modeling an IDS. The data rates involved in computer networks have increased drastically from 3 Mbps to 100 Gbps with 400 Gbps in the pipeline. As and when the speed increases the payload size also increases to incorporate the transfer rate. According to the IEEE standard 802.3 [2] the payload size may vary from 46 to 1500 bytes depending on the standards followed. In some cases it may contain even 9000 bytes of payload which are known to be the jumbo packets. This study throws some light on the importance of choosing an algorithm capable of handling huge amounts of data. The Z algorithm [3] which is extensively used for pattern matching in large volumes of data such as in DNA sequences can be adapted for the pattern matching scenario followed in the case of IDS also. The rest of the paper is organized as follows: Section 2 reviews the techniques reported in literature on pattern matching for NIDS and also describes the Z algorithm. Section 3 presents an analysis of the results obtained after evaluating the string matching algorithms on different types of packet traffic. Section 4 concludes the paper.

II. METHODOLOGY

This section will briefly explain the various surveys done in the field of pattern matching and the scope of introducing a new algorithm for the improvement in performance and scalability. The upcoming subsections discuss the prior works that has been done in the case of pattern matching algorithms used for the implementation of different type of NIDS, the proposed system architecture, the working of algorithm, and the work done.

A. Prior Works

The field of pattern matching is well researched for a variety of applications such as text matching, bio sequence matching, dictionary finding, big data analysis, security systems etc. As the information data is getting huge the need for efficient string matching algorithms arises. The various string matching algorithms for NIDS reported in the literature are discussed briefly in this section. These algorithms include the famous algorithms like Aho Corasick Algorithm, Boyer Moore Algorithm, Brute Force Algorithm, and KMP Algorithm.

Aho Corasick [4]

This algorithm was one of the oldest algorithms developed for multi-pattern matching which is capable of matching strings at a worst case time. The Aho-Corasick algorithm constructs a state machine out of the strings or patterns to be matched. It consists of a root node which is considered to be a non-matching state. The root node is the starting node for the algorithm. Each and every pattern considered will add different states and the state machine complexity increases exponentially. The newly obtained state machine is scanned and a pointer called failure pointer is updated with reference to that which will give a way for back tracking in the case of non-detection of a particular substring. The algorithm needs only a single reference to the memory if the pattern data base is fully optimized.

Boyer Moore [6, 7]

The Boyer Moore algorithm is used when the searched pattern is much shorter when compared to the text in which the search is going on. The basic working of this algorithm includes the preprocessing of the pattern or string which is being searched. The preprocessed results are used by the algorithm for discarding some sections of the text and continuing search from the next valid position. The Boyer Moore algorithm mainly works from the back end of the text rather than the front end. As the size of the pattern increases the speed of algorithm also increases because of the number of letters in the section being skipped increases with that.

Brute Force [6]

This algorithm is known to be the exhaustive searching algorithm which works on the systematic enumeration of almost all possible combinations for the solution and finding whether it is producing satisfactory solutions. The algorithm is not commonly used but it is simple in implementation and will always find a possible solution if one exists. The time complexity analysis shows that the algorithm follows a polynomial behavior instead of the linear behavior exhibited by the other algorithms.

KMP Algorithm [6]

KMP algorithm or Knuth–Morris–Pratt algorithm is one of the most popular algorithms used in text searching applications. It generally searches for the pattern P of length n in a text T of length m. The algorithm is capable of finding out the next match with the help of certain information embedded in the pattern itself when a mismatch occurs while parsing the text. This will help bypassing the repeated reexamination of antecedent matched characters. The study of these algorithms gives the idea of how different algorithms behave with the different scenarios in text searching. Some of them are having less complexity, some can have the included substring matching capability and some others show more dependency on patterns that have to be matched. Flexibility, sub pattern matching capability and reduced comparisons are key features demanded from string matching in NIDS.

B. *Overall Architecture for NIDS [1]*

The Network Intrusion Detection System comprises of many different elements or modules which will work together to attain the complete functionality of the system. The modular diagram of an architectural model proposed by Pontarelli et al [1] is shown in the Fig. 1. The network packets from the other networks and hosts are received at the Ethernet interface and same is buffered in an Ethernet buffer as the data rates are much higher in current networks. The buffered packets are then analyzed by a header analyzer in order to identify the protocols used for transfer. IDS generally have separate rules or patterns for different protocols used for communication, that has to be detected in the payload or data part present in each and every packet so as to confirm a malicious activity or a violation of security policy inside the network. After analyzing the packet for identifying the protocol used, a protocol based classifier is used to classify the data part to different buffers as shown in the block diagram.

Depending on the protocol used (TCP, UDP, IP, ICMP, etc.) there exists different String Matching Modules (SMM) that are dedicated to each packet type. A packet routing mechanism is used to route packets between the SMMs. The SMMs are intended to detect different malicious patterns associated with each rules. Once the patterns have been detected a rule identifying mechanism identifies each and every rule with the help of pattern identifier or SMM output. The Decision maker and Controller modules categorize the outputs from rule identifier and with the help of a prediction mechanism arrive at some decisions such as logging an attack, blocking a user, blocking a host from the entire network, raise alarms on certain events and also monitor different intrusions. These logs are used by some organizations to identify problems associated in their cyber security policy, documenting different threats and proactively putting threat mitigation mechanisms in place.

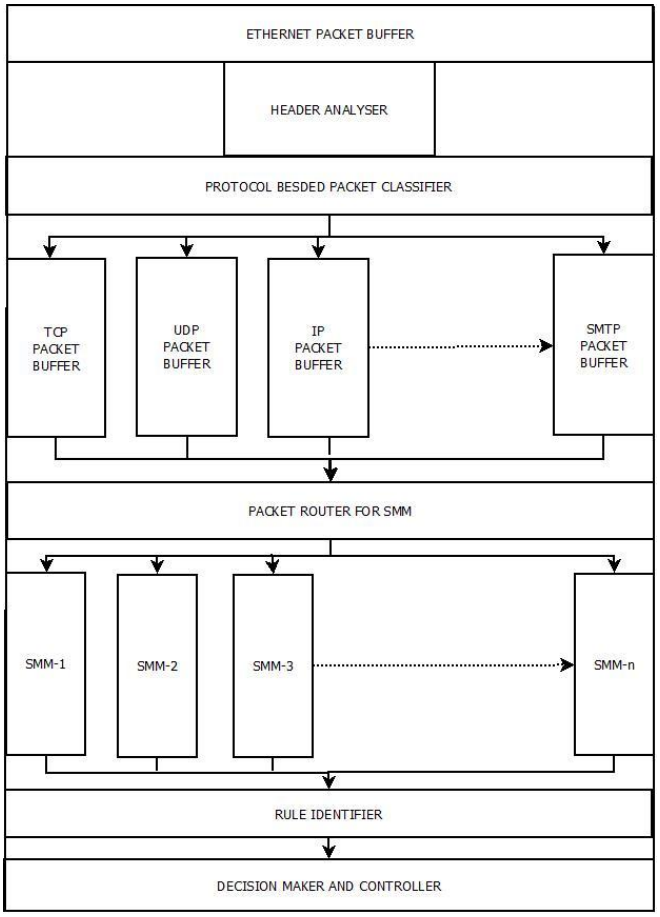

Fig. 1. Overall Architecture of NIDS.

C. ***Z – Algorithm [3]***

The Z – Algorithm was first found to be useful in the field of Bio Sequence detection where the text search field is large. A common application of the Z – Algorithm is in the profile matching of different DNAs. Due to the simplicity in the design and low time complexity exhibited by the Z – Algorithm, it is more convenient to be used in any pattern detection schemes where a huge number of strings have to be compared.

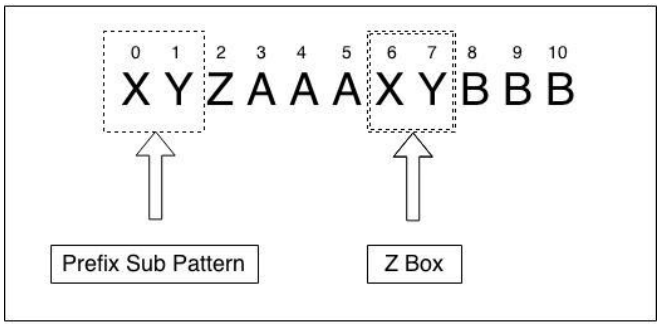

Fig. 2. Z Box formation in a pattern.

There are mainly two concepts up on which the Z – Algorithm is based. They are the *Z box* and *Z value*. The Z-Algorithm can be considered to be a signature based algorithm. The *Z boxes* are the substrings that show a match with the string prefixes with the same number of characters. The *Z values* represent the length of *Z box* i.e. the length of substring. The example in Fig. 2 illustrates the concept of Z boxes. The string "XYZAAAXYBBB" shown in the figure has only one Z box i.e. at the position 6 and 7 which shows a match with the prefix substring "XY". So the Z value for the position 6 will be 2 as there is a match of two positions.

In a combined string *C* of length *n*, the Z Algorithm produces a set of values called *Z values* where *Z[i]* represents the length of the longest sub string located at the position *C[i]* which is also similar to the prefix of string *C*. The condition *Z[i]* = 0 implies that the *C[0]* ≠ *C[i]*. The i value changes from 1 to *n-1* excluding the 0^{th} value in order to avoid the comparison of string with itself. For iterating through the text an interval is maintained as {*l*, *r*} inside where the *i* value is defined such that $1 \leq l \leq i \leq r$. The value of *l* gives the lower bound and the *r* value gives the upper bound of *i*. If there exists a valid interval of {*l*, *r*} for (*i-1*)th position and also if the updated values of *Z* till that position are available, then *Z[i]* and the corresponding interval {*l*, *r*} are calculated by the following steps.

If *l* > *r*, then there will not be an existence of a possible prefix substring that starts before the position *i* and which will end up at a position at or after the *i* value. When this condition occurs, a reset is applied and new {*l*, *r*} values are computed by comparing the sub-strings *C[0….]* to *C[i…]* and *Z[i]* is updated as *(r-l+1)*. If the second condition is satisfied, the current values of left and right indices at least extend to position *i*. Here another variable *j* is defined as *j* = *i-l*. In this case *Z[i]* ≥ min(*Z[j]*, *r-i+1*) since *C[i…]* matches to *C[j…]* up to at least (*r-i+1*)th position. In this case if the Z value exhibits a condition that *Z[j]* < *r-i+1* then there does not exist a sub string of prefix starting at the position *i*. This implies that the Z value *Z[i]* = *Z[j]* and the left and right bounds do not vary. If *Z[j]* ≥ *r-i+1* then it is possible to have new Z value as there is chance of finding match of more than *r-l+1* characters *C[0…]* and *C[i…]*. This condition yields an update to the left and right bounds i.e. {*l*, *r*}. The new *l* value will be the value of *i* and the matching process is conducted from the new position *C[r+1]* forward to find the new *r* value from which the *Z[i]* value will be updated correspondingly.

The Z Algorithm is capable of finding the whole Z values at a single iteration from 0th position up to the (*n-1*)*th* position in the string. This algorithm has a complexity of *O(n)* and is also capable of identifying the prefix as well as the prefix substrings. The different features exhibited by the Z Algorithm make it useful for systems like NIDS where a large number of patterns have to be matched within a huge amount of data in a short time.

III. RESULTS AND EVALUATION

The various algorithms discussed are modeled and the delay incurred by each is estimated. The different values are plotted to evaluate the performance of each algorithm. The algorithms that are considered are Aho-Corasick Algorithm, Boyer Moore

Algorithm, Brute Force Algorithm, Knuth Morris Pratt Algorithm and the Z Algorithm. The complexity, behavior, multiple matches in single iteration and rule dependency of these algorithms are tabulated below in Table. 1. For delay estimation various scenarios are considered involving the application of single patterns and multiple patterns with same length, multiple patterns of same length with substrings included in each set of patterns and nonlinear patterns with random number of substrings. The delay of each algorithm is plotted and compared with the help of graphs which are provided in the figures below.

The plot in Fig. 3 shows the performance evaluation of different algorithms for a given string of size 10 bytes which searches in a text of size 64 bytes. All algorithms except Brute Force experience the same worst case delay. For further analysis multiple patterns with same length of 10 byte are applied with the same number of branches in each set of input patterns. The plotted results in Fig. 4 show that the Aho Corasick and Z algorithms show almost the same behavior with a small reduction in delay in the case of Z algorithm. The Z algorithm shows a delay reduction of 1.33%. The plot in Fig. 5 compares the Aho Corasick algorithm and the Z Algorithm for nonlinear patterns with random number of substrings. The algorithms exhibit a wide variation in performance and notable reduction in delay is observed for the Z algorithm. The estimated reduction in delay is 42.81% for the Z Algorithm. So as far as the randomness in the test patterns increases the Z Algorithm shows a dramatic improvement in performance which makes it highly desirable in an NIDS scenario. The scalability of the system is ensured by adopting memory based approaches and exploiting the flexibility of the Z Algorithm to incorporate random strings.

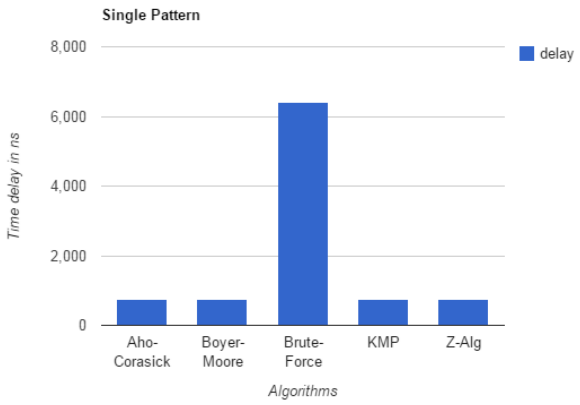

Fig. 3. Comparison of delay for same single pattern for different algorithms.

TABLE I. ANALYSIS OF DIFFERENT ALGORITHMS USED FOR PATTERN MATCHING PROBLEMS

Algorithm	Time complexity	Behavior	Multiple Matches	Rule Dependency
Aho Corasick	$O(m+n+z)$	Linear	Yes	Yes
Boyer Moore	$O(m+n)$	Linear	No	Yes
Brute Force	$O(m*n)$	Polynomial	No	No
Knuth Morris Pratt	$O(m+n)$	Linear	No	Yes
Z Algorithm	$O(m+n)$	Linear	Yes	No

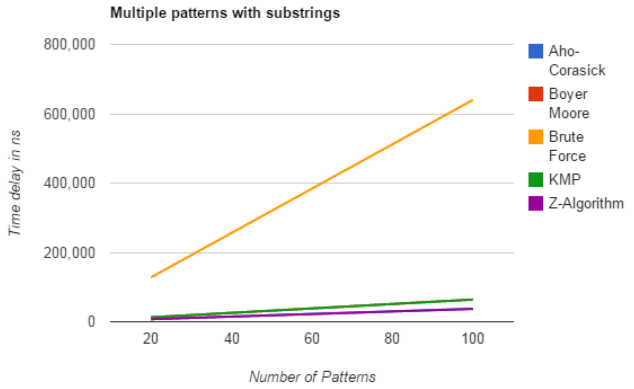

Fig. 4. Comparison of delay for multiple strings of same length with substrings included.

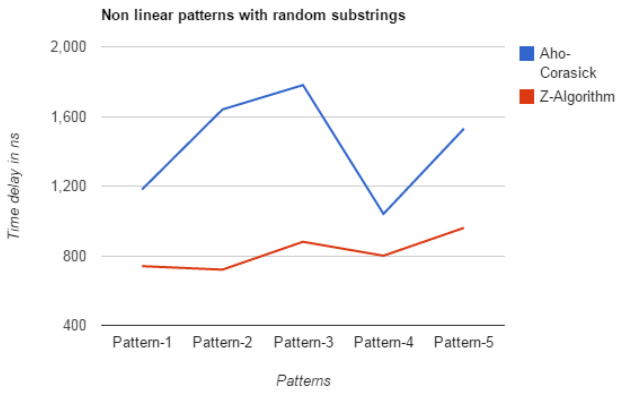

Fig. 5. Comparison of Aho - Corasick and Z Algorithm for nonlinear patterns with random number of substrings.

IV. CONCLUSION

This paper proposes the adaptation of the Z algorithm for fast and scalable intrusion detection in IDS. The Z Algorithm is capable of incorporating substring matching and its

performance is not pattern dependent. The pattern-independent nature of Z Algorithm helps to improve the scalability and flexibility of the String Matching Modules present in the NIDS architecture. The analysis results of different algorithms with various pattern matching scenarios show that the Z Algorithm can have more than 40% improvement in delay compared to the next best algorithm. The performance improvements observed for the Z algorithm will become more significant with higher degree of randomness in packet traffic and higher data rates.

REFERENCES

[1] Salvatore Pontarelli, Giuseppe Bianchi, and Simone Teofili.: Traffic-Aware Design of a High-Speed FPGA Network Intrusion Detection System. In: IEEE transactions on computers, vol. 62, no. 11, November 2013

[2] IEEE Standard for Ethernet. IEEE Std 802.3™-2012 Revision of IEEE Std 802.3-2008

[3] Dan Gusfield.: Algorithms on Strings, Trees, and Sequences Computer Science and Computational Biology. University of California, Davis.

[4] Nathan Tuck, Timothy Sherwood, Brad Calder, George Varghese, "Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection", IEEE INFOCOM 2004

[5] Chris Clark, Wenke Lee, David Schimmel, Didier Contis, Mohamed Koné, Ashley Thomas, "A Hardware Platform for Network Intrusion Detection and Prevention", Center for Experimental Research in Computer Systems (CERCS) Georgia Institute of Technology, Atlanta, GA, USA

[6] K. Prabha and Dr. S. SukumaranCzajkowski.: Single-Keyword Pattern Matching Algorithms for Network Intrusion Detection System. International Journal of Computer and Internet Security. ISSN 0974-2247 Volume 5, Number 1 (2013)

[7] S. S. Sheik, Sumit K. Aggarwal, Anindya Poddar, N. Balakrishnan, and K. Sekar.: A FAST Pattern Matching Algorithm. J. Chem. Inf. Comput. Sci. 2004, 44, 1251-1256