TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 13/19 pp585-593
Volume 12, Number 5, October 2007

Memory Efficient String Matching Algorithm for Network Intrusion Management System*

YU Jianming (余建明)[1,2], XUE Yibo (薛一波)[2,3], LI Jun (李 军)[2,3,**]

1. Department of Automation, Tsinghua University, Beijing 100084, China;
2. Research Institute of Information Technology, Tsinghua University, Beijing 100084, China;
3. Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China

Abstract: As the core algorithm and the most time consuming part of almost every modern network intrusion management system (NIMS), string matching is essential for the inspection of network flows at the line speed. This paper presents a memory and time efficient string matching algorithm specifically designed for NIMS on commodity processors. Modifications of the Aho-Corasick (AC) algorithm based on the distribution characteristics of NIMS patterns drastically reduce the memory usage without sacrificing speed in software implementations. In tests on the Snort pattern set and traces that represent typical NIMS workloads, the Snort performance was enhanced 1.48%-20% compared to other well-known alternatives with an automaton size reduction of 4.86-6.11 compared to the standard AC implementation. The results show that special characteristics of the NIMS can be used into a very effective method to optimize the algorithm design.

Key words: string matching; network intrusion management system (NIMS); Aho-Corasick (AC) algorithm

Introduction

Network intrusion management systems (NIMSs) are fundamental security applications that are growing in popularity in various network environments. The heart of almost every modern NIMS has a string matching algorithm. The NIMS uses string matching to compare the payload of the network packet and/or flow against the pattern entries of intrusion detection rules[1,2].

String matching requires significant memory and time costs. For example, the string matching routines in Snort account for up to 70% of the total execution time and 80% of the instructions executed on real traces[3]. The size of the string matching data structure is more than 150 MB when using the Aho-Corasick (AC) algorithm[4] and the Snort rule set distributed on July 27, 2005. Moreover, as the number of potential threats and their associated patterns continues to grow, the memory and time costs of string matching are likely to increase as well.

These challenges motivate research on the design of string matching algorithms specific to NIMS applications[5-12]. However, most previous algorithms have not utilized the specific characteristics of NIMS patterns to improve the string matching performance. The E^2xB[8] algorithm utilized the characteristics of NIMS input based on the observation that the input size is relatively small (on the order of packet size) and the expected matching probability is also small (which is common in network intrusion detection system (NIDS) environments).

Hardware applications have also been proposed including field programmable gate array (FPGA) and application specific integrated circuits (ASIC)[13-20].

Received: 2005-12-23; revised: 2006-06-30
* Supported by the Juniper Research Grant and Intel IXA University Program
** To whom correspondence should be addressed.
E-mail: junl@tsinghua.edu.cn; Tel: 86-10-62796400

The hardware methods can certainly achieve higher string matching performance, but the rule set cannot be easily updated, especially with the ASIC method. Software algorithms are less expensive and more flexible. With special-purpose, programmable chips tailored to network devices such as network processors (NPs), software algorithms can also achieve high performance and can combine the low cost and flexibility of commodity processors with the speed and scalability of custom silicon (ASIC chips).

In this work the characteristics of NIMS patterns are used to design a faster string matching algorithm that takes less memory. An improved AC algorithm, the character indexed AC (CIAC), was developed to dramatically reduce the memory requirement.

1 Snort and String Matching

1.1 Snort

Snort is the most popular open source network intrusion detection system and its detection model is used for reference by many commercial products.

Snort captures packets from a network interface which are preprocessed before sending to the detection engine. The preprocessing includes layer three IP fragment reassembly, layer four transmission control protocol (TCP) session reconstruction, and so forth. The detection engine checks packet payloads against the intrusion detection rules. If one or more rules match, an attack is detected and the corresponding response functions are launched.

The detection rules form a rule set with all the pattern entries of the rules forming a pattern set. For newer versions above Snort version 2.0, the detection rules are divided into many groups, referred to as sub-rule sets in this paper. The pattern entries of each sub-rule set form a sub-pattern set. For example, the TCP and user datagram protocol (UDP) rules are divided into sub-rule sets by the source and destination port numbers. When a TCP or UDP packet arrives, its destination and source port number are used to find the appropriate sub-rule sets to be checked. Then a string matching algorithm, such as AC, is used to compare the packet payload with the corresponding sub-pattern sets. If there are matching patterns, the rules that contain the matching patterns are checked to confirm whether an attack is occurring.

1.2 String matching algorithm

String matching consists of finding one, or more generally, of all the occurrences of a search string in an input string. In NIMS 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.

1.2.1 Boyer-Moore (BM) algorithm

The BM algorithm[21] is the most well-known single pattern matching algorithm. The BM algorithm utilizes two heuristics, bad character and good suffix, to reduce the number of comparisons. Both heuristics are triggered on a mismatch. The BM algorithm takes the far most shift caused by the two heuristics.

Horspool proposed a variation of the BM algorithm, the BM-Horspool (BMH) algorithm[22], which utilizes only an improved bad character heuristic. BMH is simpler to implement than BM, which preserves the average performance of BM.

1.2.2 Maximum weighted matching (MWM) algorithm

The MWM algorithm[23] uses the bad character heuristic like the BM algorithm but with a two-byte shift table. The MWM algorithm also performs a hash on the two-byte prefix of the current substring of the input string to index into a group of search strings. The MWM algorithm can efficiently deal with large amounts of search strings. However, its performance depends on the length of the shortest search string and the characteristic of the input string.

1.2.3 AC_BM and SBMH algorithms

The set-wise Boyer-Moore-Horspool (SBMH) algorithm[5] is regarded as the first NIDS-specific string matching algorithm. This algorithm adopts heuristics like BM to simultaneously search for multiple search strings. Coit et al.[6] independently proposed a similar algorithm called AC_BM.

1.2.4. E^2xB algorithm

The E^2xB algorithm[8] is an exclusion-based algorithm specific to NIDS applications. This algorithm is based on the observation that if there is at least one character of the search string that is not contained in the input string, then the search string is not a substring of the input string. E^2xB first checks the input string for missing fixed size substrings of the search string. If all the substrings of the search string can be found, a standard