ELectronics EXpress

IEICE Electronics Express, Vol.5, No.22, 949–953

A fast and scalable string matching algorithm using contents correction signature hashing for network IDS

J. S. Wang[1], H. K. Kwak[1a)], Y. J. Jung[1], H. U. Kwon[1], C. G. Kim[2], and K. S. Chung[1]

[1] *School of Electronics Engineering, Soongsil University, 511 Sangdo-dong, Dongjak-gu, Seoul, Korea*

[2] *Department of Computer Science, Namseoul University, 21 MaejuRi, SeongwhanEub, ChangAn, ChoongNam, Korea*

a) *gobarian@q.ssu.ac.kr*

Abstract: A contents correction signature hashing scheme for fast string matching algorithm in high performance network intrusion detection systems is proposed. The objective is to design a highly scalable pattern matching system suitable for wide spreading similar varieties of malicious behaviors in network. Simulation results show the proposed method maintains the hash table evenly compared with general hash table and shows high scalability as the increase of signatures and 130% of performance gain compared with SNORT.

Keywords: signature hashing, string matching, SNORT, IDS

Classification: Science and engineering for electronics

References

[1] S. Dharmapurikar and J. Lockwood, "Fast and Scalable Pattern Matching for Network Intrusion Detection Systems," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1781–1792, Oct. 2006.

[2] A. Das, D. Nguyen, J. Zambreno, G. Memik, and A. Choudnary, "An FPGA-Based Network Intrusion Detection Architecutre," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 1, pp. 118–132, March 2008.

[3] D. Fava, S. Byers, and S. Yang, "Protecting Cyberattacks Through Variable-Length Markov Models," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 3, pp. 359–369, Sept. 2008.

[4] T. Tao and A. Mukherjee, "Pattern Matching in LZW Compressed Files," *IEEE Trans. Comput.*, vol. 54, no. 8, pp. 929–938, Aug. 2005.

[5] G. Xie, J. Yang, V. Issarny, and A. Conte, "An Accurate and Efficient 3-Phase measurement Method for IP Traffic Flow on High Speed Link," 6th International Conference on Networking, pp. 46–46, April 2007.

DOI: 10.1587/elex.5.949
Received September 05, 2008
Accepted October 17, 2008
Published November 25, 2008

1 Introduction

As the increasing popularity of network applications through internet, the need for network security has been increased due to the wide spreading malicious behaviors such as system hacks and viruses. In order to detect them, also known as signatures, from packet payload, deep packet inspection based on string matching techniques should be performed by intrusion detection systems (IDS) [1]. It has to be done to all incoming bytes, against thousands of pattern characters at wire rate. This process is computationally expensive and affects greatly overall system throughput [2].

String matching in network IDS is characterized as long and large scale patterns because of various types of virus, worm, trojan, and malware day by day. So far, tree based algorithms [3] have been regarded as a base candidate for them with the worst-case time complexity $O(n)$, where n is the length of patterns. However, the time complexity of these algorithms is linear in the size of the input.

Therefore, this paper proposes a novel high-speed and scalable string matching algorithm, contents correction signature hashing (CCSH), for large-scale pattern sets in network. This uses a conventional hash function based on division method and if two or more rules have the same key value, then we build a linked list of all the synonyms of each rule. To find a record in the table, we would first find its home address and then search the list which is associated with that particular home address.

2 Contents Correction Signature Hashing

In the proposed pattern matching system, two separate hashing tables are configured for ports and signatures, respectively. The proposed CCSH is applied to the latter. Here, we choose a hash function $f(x)$ that maps a set of signatures into an integer interval from 1 to n. For the reference value x, a 2-byte value is taken in a signature based on the random position selection scheme. Two bytes can generate 65,536 possible hashing values sufficiently enough for covering all rules existing now. It can minimize the cost of scanning data. This can be justified simply with following equations: $M = n - k + 1$ and $T = M \times k$ where M is the number of hash calculations, n is payload size of bytes, k is the size of reference value; T is total data size required for comparisons. Therefore, the increase of window size may decrease the number of hash calculations; and then it definitely increases the scanning data size required for string matching. So two bytes can allow the minimal scanning result.

The random position selection scheme is devised by taking account of current network environments in which various worms have spread widely and a lot of their varieties have kept appearing day by day. Their common characteristic is that there exists strong similarity between them. This means that the relationship between signatures is extremely high, which results in increasing the possibility that they may have an identical hashing value based on a fixed positioned 2-byte selection method. In this case, a linked

DOI: 10.1587/elex.5.949
Received September 05, 2008
Accepted October 17, 2008
Published November 25, 2008

list connect to a hashing value has the possibility of becoming very long. As a result, the worst case time complexity becomes in proportion to the maximum number of nodes of which have same hashing values.

The proposed CCSH scheme enables the signatures closely correlated with each other to have different hashing values, which comes in effect that the signature hash table becomes distributed evenly rather than being concentrated on a same hashing value. The other effect is to increase the matching speed through the effective distribution. As a result, the proposed CCSH can be used for improving the matching efficiency and guaranteeing the scalability to the number of rules.

3 Algorithm

The proposed CCSH is composed of two stages as shown in Table I: the byte windowing and rule matching. They are expressed in *while* and *for* loop statements. In general hashing tables, full-text matching is required between the data part of inspecting packet and an actual signature. However, in CCSH, a hashing value is taken from two bytes from a signature such that the information, how far a 2-byte is away from the beginning part of a signature, is kept in the information of a rule connected to the signature hash table.

Byte Windowing: In order to inspect packet data area being compared with a 2-byte reference value, windowing should be executed based on 2-byte stride. The hashing value calculated through byte windowing is compared with the values in the signature hash table. This process continues until either finding a matching value or reaching at the end of the table. Here, the actual number of rules to be compared with can be decreased because they are reduced by port number and protocol and then hash table is used to reduce the number further.

Rule matching: Once the byte windowing is finished or there is a matching case, the rule matching stage will be executed to find whether or not a packet accords with real rules. For this stage more detailed information are prepared such as the information on origin, destination, and location in signature. That information will be compared prior to the full matching between a signature and data area. Therefore, it can reduce not only the number of matching numbers but also the possibility of false detections.

Table I. The proposed CCSH algorithm

```
WHILE offset >= sizeof(packet)
  READ window                          // Byte windowing: read 2 bytes for hash
  h = HASH(window)                     // Byte windowing: get hash value
  IF HashTable[h] is not empty         // Byte windowing: find signature link in hash table
    FOR
      COMPARE (packet, signature)      // Rule matching: compare signature with packet
      IF COMPARE is true
        MATCH OK                       // Rule matching: signature matched
      ENDIF
  ENDIF
  INCREMENT offset                     // Byte windowing: move match window
ENDWHILE
```


DOI: 10.1587/elex.5.949
Received September 05, 2008
Accepted October 17, 2008
Published November 25, 2008

4 Experimental Results

In order to evaluate the proposed algorithm, we have implemented a real-time network IDS using a 2.6.16.18 version Linux kernel and the packet was processed using REROUTING hook point in Netfilter, an open source Linux kernel module for intercepting and manipulating network packets. For performance comparison, we select Jump Aho-Corasick algorithm [4] in 2.4.5 version SNORT and Smartbit [5] as a packet generator.

First, the effectiveness of 2-byte random position selection scheme in CCSH is verified by the level of even distribution of hash table. Figure 1 shows the distribution ratio of 1,000 rules between general hash table and CCSH. When using conventional hash table, the distribution of hashing values are concentrated on certain value intervals, for example, between 20 and 30. As mentioned before, this is a serious cause of performance degradation and bad influence on performance estimation on an algorithm. However, when taking the proposed method, the distribution is even, which shows the effectiveness of the proposed algorithm. Moreover, this guarantees the performance when there is a strong correlation between signatures and even distribution in accordance with increasing number of rules.

Next, we evaluate the packet processing performance. The performance comparisons are conducted on different rule sizes in order to verify the rule scalability and network stability. We vary the number of rules up to 1,000 with 1 Gbps transmission rates, as shown in Figure 2. Here we can easily find that the number of rules affect on the overall performance in SNORT; however, CCSH shows high scalibiltiy as the increase of signatures without degrading performance. At the same time, it achieves 130% of performance increase compared with SNORT. The performance gap is extended further because it is programed at kernel-level while SNORT at the application-level.

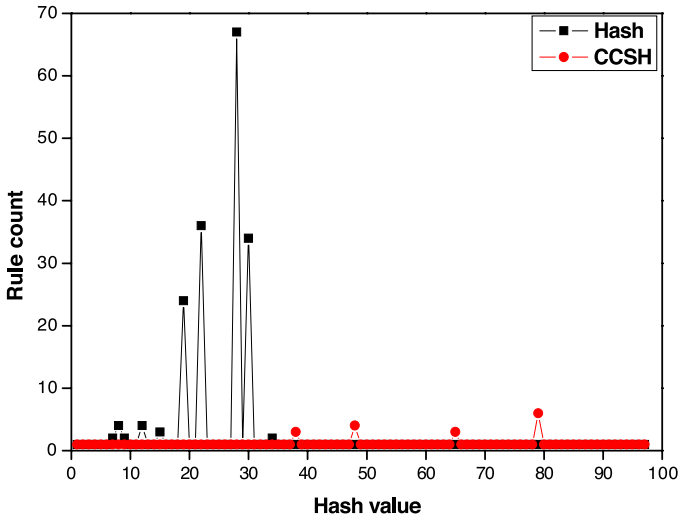

Fig. 1. The distribution ratio of 1,000 rules between general hash table and CCSH

DOI: 10.1587/elex.5.949
Received September 05, 2008
Accepted October 17, 2008
Published November 25, 2008

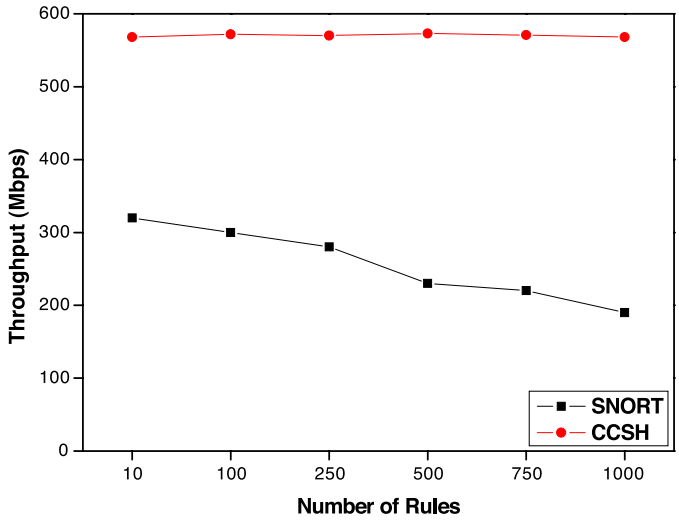

Fig. 2. The packet processing performance with the number of rules

5 Conclusion

This paper proposes a novel string matching algorithm, called contents correction signature hashing (CCSH), to detect malicious packets in network with scalability and high speed. Simulation results show the proposed method maintains the hash table evenly compared with general hash table and shows high scalability as the increase of signatures and 130% of performance gain compared with SNORT. Especially, considering on the network environment of wide spreading malicious behaviors such as system hacks and viruses day by day, the proposed algorithm can achieve much higher performance improvement.

Acknowledgments

This work was partially supported by the Basic Research Program of the Korea Science and Engineering Foundation grant No.R01-2006-000-11167-0 and partially by the Soongsil University Research Fund.

DOI: 10.1587/elex.5.949
Received September 05, 2008
Accepted October 17, 2008
Published November 25, 2008