ISSN:2249-5789

Design and Implementation of a Low Complex Pattern Matching Algorithm for Memory Based Computations

SRI M.L.NAIDU(E.C.E)
Email: mllaxmi@gmail.com

K.NIVEA (MTECH)
Email:kalinivea@gmail.com

ADITYA Institute of Technology and Management
Affiliated to jntu Kakinada

Abstract: Network intrusion detection system is used to inspect packet contents against thousands of predefined malicious or suspicious patterns. Because traditional software alone pattern matching approaches can no longer meet the high throughput of today's networking, many hardware approaches are proposed to accelerate pattern matching. Among hardware approaches, memory-based architecture has attracted a lot of attention be- cause of its easy reconfigurability and scalability. In order to accommodate the increasing number of attack patterns and meet the throughput requirement of networks, a successful network intrusion detection system must have a memory-efficient pat-tern-matching algorithm and hardware design. In this paper, we propose a memory-efficient pattern-matching algorithm which can significantly reduce the memory requirement. For Snort rule sets, the new algorithm achieves 21% of memory reduction compared with the traditional Aho–Corasick algorithm. In addi-tion, we can gain 24% of memory reduction by integrating our approach to the bit-split algorithm which is the state-of-the-art memory-based approach.

Index Terms—Aho–Corasick (AC) algorithm, finite automata, pattern matching.

INTRODUCTION

THE MAIN purpose of a signature-based network intru-sion detection system is to prevent malicious network attacks by identifying known attack patterns. Due to the in-creasing complexity of network traffic and the growing number of attacks, an intrusion detection system must be efficient,flexible and scalable.

The primary function of an intrusion detection system is to perform matching of attack string patterns. Because string matching is the most computative task in network intrusion detection (NIDS) systems, many hardware approaches are pro-posed to accelerate string matching.

The hardware approaches

may be classified into two main categories, the logic and the memory architectures In terms of reconfigurability and scalability, the memory architecture has attracted a lot of attention because it allows on-the-fly pattern update on memory without resynthesis and relayout. The basic memory architecture works as follows.First, the (attack) string patterns are compiled to a finite-state machine (FSM) whose output is asserted when any substring of input strings matches the string patterns. Then, the corre-sponding state transition table of the FSM is stored in memory.For instance, Fig. 1 shows the state transition graph of the FSM to match two string patterns "bcdf" and "pcdg", where all tran-sitions to state 0 are omitted. States 4 and 8 are the final states indicating the matching of string patterns "bcdf" and "pcdg",respectively. In the architecture, the memory address register consists of the current state and input character; the decoder converts the memory address to the corresponding memory location, which stores the next state and the match vector information. A "0" in the match vector indicates that no "suspicious" pattern is matched; otherwise the value in the matched vector indicates which pattern is matched. suppose the current state is 7 and the input character and The decoder will point to the memory location which stores the next state 8 and the match vector 2. Here, the match vector 2 indicates the pattern "pcdg" is matched.Due to the increasing number of attacks, the memory re-quired for implementing the corresponding FSM increases tremendously. Because the performance, cost, and power consumption of the memory architecture is directly related to the memory size, reducing the memory size has become imperative.

REVIEW OF AC ALGORITHM

In this section, we review the AC algorithm. Among all memory architectures, the AC algorithm has been widely adopted for string matching because the algorithm can effectively reduce the number of state transitions and therefore the memory size. Using some example the state transition diagram derived from the AC algorithm where the solid lines represent the valid transitions while the dotted lines represent a new type of state transition called the failure transitions.

BASIC IDEA

Due to the common substrings of string patterns, the

ISSN:2249-5789

compiled AC machine may have states with similar transitions. Despitethe similarity, those similar states are not equivalent states and cannot be merged directly. In this section, we first show that functional errors can be created if those similar states are merged directly. Then, we propose a mechanism that can rectify those functional errors after merging those similar states.

HARDWARE ARCHITECTURE:

Our hardware module which can be configured for matching 16 or 32 patterns with a state machine containing 1024 valid transitions at most. The register,called address_register, is used to store the current state and the input character. The valid_memory is used to store the in-formation of valid_state, path Vec, and if Final corresponding to each valid transition while the failure_memory is used to store the failure_state corresponding to each failure transition. In this prototype, we use a hardwired circuit, called A2P, to translate the content of the address_register to a contiguous scope, called pos, to utilize the valid_memory. The circuit A2P can be implemented using hardwired circuit or CAM . In addition, the signal n_valid is high if there is no valid transition cor-responding to the address_register.called pre-Reg, is used to trace the precedent path Vec in each state. The pre-Reg is initiated to be 1 for all bits and is updated by performing a bit wise AND operation on its current value and the path Vec from the valid_memory. The ns_ctrl unit is used to determine the next state by the value of pre-Reg and n_valid. If the preReg is 0 for all bits or the n_valid is 1, the ns_sel will output low to let the failure_state update the current_state register. On the other hand, if the pre-Reg is not zero and the n_valid is not 1, the ns_sel will output high to let the valid_state update the current_state register.

CONCLUSION:

We have presented a memory-efficient pattern matching al-gorithm which can significantly reduce the number of states and transitions by merging pseudo-equivalent states while main-taining correctness of string matching. In addition, the new algo-rithm is complementary to other memory reduction approaches and provides further reductions in memory needs. The experi-ments demonstrate a significant reduction in memory footprint for data sets commonly used to evaluate IDS systems.

REFERENCES

[1] A. V. Aho and M. J. Corasick, “Efficient string matching: An AID to bibliographic search,” Commun. ACM, vol. 18, no. 6, pp. 333–340,1975.

[2] M. Aldwairi, T. Conte, and P. Franzon, “Configurable string matching hardware for speeding up intrusion detection,” Proc. ACM SIGARCH

Comput. Arch. News, vol. 33, no. 1, pp. 99–107, 2005.

[3] M. Alicherry, M. Muthuprasanna, and V. Kumar, “High speed pattern matching for network IDS/IPS,” in Proc. IEEE Int. Conf. Netw. Protocols (ICNP), 2006, pp. 187–196.

[4] B. Brodie, R. Cytron, and D. Taylor, “A scalable architecture for high-throughput regular-expression pattern matching,” in Proc. 33rd Int. Symp. Comput. Arch. (ISCA), 2006, pp. 191–122.

[5] Z. K. Baker and V. K. Prasanna, “High-throughput linked-pattern matching for intrusion detection systems,” in Proc. Symp. Arch. For Netw. Commun. Syst. (ANCS), Oct. 2005, pp. 193–202.

[6] Y. H. Cho and W. H. Mangione-Smith, “A pattern matching co-processor for network security,” in Proc. 42nd IEEE/ACM Des. Autom.Conf., Anaheim, CA, Jun. 13–17, 2005, pp. 234–239.

[7] Y. H. Cho and W. H. Mangione-Smith, “Fast reconfiguring deep packet filter for 1 + GigabitNetwork ,” in Proc. 13th Ann. IEEE Symp. Field Program. Custom Comput. Mach. (FCCM), 2005, pp. 215–224.

[8] C. R. Clark and D. E. Schimmel, “Scalable pattern matching on high speed networks,” in Proc. 12th Ann. IEEE Symp. Field Program.Custom Comput. Mach. (FCCM), 2004, pp. 249–257.

[9] G. Dan, Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge, U.K.: Cambridge University Press, 1997.

[10] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood,Deep packet inspection using parallel bloom filters,” in Proc. 11th Symp. High Perform. Interconnects, Aug. 2003, pp. 44–53.

[11] S. Dharmapurikar and J. Lockwood, “Fast and scalable pattern matching for content filtering,” in Proc. Symp. Arch. for Netw.Commun. Syst. (ANCS), Oct. 2005, pp. 183–192.

[12] M. Gokhale, D. Dubois, A. Dubois, M. Boorman, S. Poole, and V. H.Granidt, “Towards gigabit rate network intrusion detection,” in Proc.the Eleventh Annual ACM/SIGDA International Conference on Field-Programmable Logic and Applications (FPL ’03), 2002, pp. 404–413.

[13] B. L. Hutchings, R. Franklin, and D. Carver, “Assisting network in-trusion detection with reconfigurable hardware,” in Proc. 10 th Annu.IEEE Symp. Field-Program. Custom Comput. Mach. (FCCM), 2002,pp. 111–120.

[14] H. J. Jung, Z. K. Baker, and V. K. Prasanna, “Performance of FPGA im-plementation of bit-split architecture for intrusion detection systems,"presented at the 20th Int. Parallel Distrib. Process. Symp. (IPDPS),Rhodes Island, Greece, 2006.

[15] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Al-gorithms to accelerate multiple regular expressions matching for deep packet inspection," in Proc. ACM SIGCOMM Comput. Commun. Rev.,2006, pp. 339–350.

[16] C. H. Lin, C. T. Huang, C. P. Jiang, and S. C. Chang, "Optimization of pattern matching circuits for regular expression on FPGA,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 12, pp.1303–1310, Dec. 2007.

[17] H. Lu, K. Zheng, B. Liu, X. Zhang, and Y. Liu, "A memory-efficient parallel string matching architecture for high-speed intrusion detection," IEEE J. Sel. Areas Commun., vol. 24, no. 10, pp. 1793–1804,Oct. 2006.

[18] J. V. Lunteren, "High-performance pattern-matching for intrusion detection," in Proc. IEEE INFOCOM, 2006, pp. 1–13.

[19] J. W. Lockwood, J. Moscola, M. Kulig, D. Reddick, and T. Brooks,Internet worm and virus protection in dynamically reconfigurable hardware," presented at the Military Aerosp. Program. Logic Device(MAPLD), Washington, DC, Sep. 2003, E10.

[20] D. Maier, "The complexity of some problems on subsequences and supersequences," J. ACM, vol. 25, no. 2, pp. 322–336, 1978.

[21] J. Moscola, J. Lockwood, R. P. Loui, and M. Pachos, "Implementation of a content-scanning module for an internet firewall," in Proc. 11th Ann. IEEE Symp. Field-Program. Custom Comput. Mach. (FCCM),2003, pp. 31–38.

[22] P. Piyachon and Y. Luo, "Compact state machines for high performance pattern matching," in Proc. 41nd IEEE/ACM Des. Autom. Conf.,2007, pp. 493–496.

[23] M. Roesch, "Snort- lightweight intrusion detection for networks," in Proc. 15th Syst. Administration Conf. (LISA), 1999, pp. 229–238.

[24] I. Sourdis and D. Pnevmatikatos, "Pre-decoded CAMs for efficient and high-speed NIDS pattern matching," in Proc. 12th Annu. IEEE Symp.Field Program. Custom Comput. Mach. (FCCM), 2004, pp. 258–267.