International Journal of Computational Research and Development (IJCRD)
Impact Factor: 5.015, ISSN (Online): 2456 - 3137
(www.dvpublication.com) Volume 3, Issue 2, 2018

NETWORK INTRUSION DETECTION SYSTEMS USING PATTERN MATCHING ALGORITHM

K. Nithya* & M. Lakshmipriya**
* M.Phil Student, PRIST University, Thanjavur, Tamilnadu
** Assistant Professor, PRIST University, Thanjavur, Tamilnadu

Cite This Article: K. Nithya & M. Lakshmipriya, "Network Intrusion Detection Systems Using Pattern Matching Algorithm", International Journal of Computational Research and Development, Volume 3, Issue 2, Page Number 26-30, 2018.

Abstract:

Intrusion Detection Systems (IDSs) recognized powerful tools identifying, deterring deflecting malicious attacks in network. Every intrusion detection system is ability to through packets and identify content matches known attacks. In this paper, common algorithms are examined of Network Processor which is intended used in Network Intrusion Detection System (NIDS). Afterword, the suitable algorithm for Network processor is chosen which combine string matching Native algorithms because these algorithms processor

Key Words: Intrusion Detection System, Network Processor & Searching Algorithm

1. Introduction:

Intrusion Detection Systems (IDSs) is one of the most useful tools identifying malicious attempts in the network and protecting the systems without modifying end-user. Different from firewalls that check specified fields of packet headers, IDSs detect the malicious information payloads. An IDS typically contains that describes the signatures malicious behavior. The number of patterns generally thousands and still increasing. The signatures appear anywhere in packet payload. Therefore, IDSs must be capable indepth packet even when suffering attacks; otherwise the protectorate is not be defended strictly.

The core of intrusion detection system is string matching algorithm. String matching is the main task intrusion detection. The algorithm identifies those packets that contain data matching signatures of known attack. Intrusion detection system takes action that could vary alerting system administrator to dropping packet in case of inline IDS. The problem of pattern matching is investigated, many algorithms exist they can be classified into either single pattern string matching multiple pattern string matching. In single pattern string matching the packet searched single string at a time. On the other side, multiple pattern strings matches search the packet for the set of strings all at once.

We introduce vectorizable design an exact pattern matching algorithm which nearly doubles the performance when compared, SIMD capable commodity hardware, such as Intel's Haswell processors or Xeon. Vectorization as a technique to increase throughput is gradually taking a more central. For example, architectures with SIMD instruction-sets now provide wider vector registers (256 bits with AVX) and introduce new instructions, such as gathers, that make applicable to wider range applications. Moreover, modern processor designs shifting towards new architectures, like Intel's Xeon that, for example, supports 512 bit vector registers. On those platforms, is not just an option but a must, in order to achieve high performance.

2. Intrusion Detection Systems:

Define Intrusion Detection: An intrusion set of actions that make attempts to challenge the integrity, discretion accessibility of a resource. Generally the practice of intrusion detection involves the tracking of important events which take place in a computer system and analyzing them in order to detect the potential presence for intrusions Comprehensive definition of intrusion detection, describing it as a collection of practices and mechanisms used to detect errors that may lead to security failure with the use of abnormality and misuse restricting and by diagnosing intrusions due attacks. Correspondingly may be added an intrusion detection system is the practical implementation intrusion detection principles and mechanisms over a network.This is a combination of application and/or main hardware components that run on a host machine monitoring the activities of users and programs searching for possible insider threats on the host Manito device and also inspecting network traffic of networks that are connected to the host, looking for outsider threats. The main objective of an IDS is to alert administrators of suspicious activities and in some cases even attempt to circumvent trial the means less attacks. The practices employed in IDSsdo differ from other security techniques such as firewalls, access control or encryption which aim to secure the computer system. Intruder's detection was a form of pre-post analysis, analysis of intrusions and movements in system structure were only identified long after the main event. Researchers developed anIDS that reviewed audit data was produced. This advancement spawned the first version real time IDSs allowing for attack pre-emption through methods of real time response. As the world entered the technological age, the market demand for IT security increased and IDS were further developed and made available to large organizations. New updateable features were developed such as various new alert methods, updates to attack pattern definitions, dedicated user friendly interfacesandpreventiontechniquesthatautomaticallystopped mutual attacks when identified.

Traditional Approach Multiple-Pattern Matching: The most commonly used pattern matching algorithm for network-based intrusion detection. It creates a finite-state automaton from the set patterns and reads the input

byte by byte to traverse the automaton and match multiple patterns. Even though it performs a small number of operations for every input byte, it implies– in practice and on commodity hardware – low instruction throughput due to frequent memory accesses with poor cache locality the number of patterns increases, the size of the state automaton increases exponentially and does not fit in the cache. Nevertheless, the method is heavily used in practice; e.g., both Snort one of the best known intrusion detection systems. Besides state-machine based approaches, there is a family of algorithms that rely on filtering to separate the innocuous input from the matches. Recent work focuses on alleviating the problem of long latency lookups on large data structures. Present a novel algorithmic design called DFC (Direct Filter Classification) that replaces the state machine approach of with a series of small, succinct summaries called filters. Such a filter is a summarizes only a specific part of each pattern, e.g. its first two bytes, having one bit for every possible combination of two characters that can be found in the patterns.

In the filtering phase, a sliding window two bytes over the input goes through an initial filter, as described above, quickly evaluate whether the current position is possible starting point a match. The two-byte windows that passed the initial filter fed to other, similar filters, each specializing on family of patterns depending on their length. Since the filters are small (8KB each), they usually fit in L1 cache. Thus, main part of the algorithm differs from uses only cache-resident data structures, resulting in up to 3.8 times less cache misses If window of two characters passed filters, there is a strong indication that starting point of a match. For this reason, the next verification phase, the DFC algorithm performs lookups on specially designed hash tables, containing the actual patterns and performs exact matching on the input and the pattern, to verify the match. Single Instruction Multiple Data (SIMD) execution model for data parallel applications, which utilizes processing units that operate vector of elements simultaneously, instead of separate elements time. SIMD is a desirable goal in computationally intensive, numberapplications, where computation is performed on independent data, sequentially stored in memory. Among several optimizations, use specially designed compact hash tables that are different for different pattern lengths. Translated to our improved filtering design, if the input at some position passed the filtering, in the verification phase the algorithm will perform a match on the compact hash table that stores references to all the patterns of appropriate size. For example, passed the third filter that stores information on patterns that are four bytes or longer, in the verification phase, the algorithm performs a match on the compact hash table that stores patterns of four bytes or longer. Each hash table is indexed with as many bytes as the shortest pattern that the hash table contains. Each bucket in the hash table contains references to the full patterns algorithm has to compare each one of them individually with the input, before reporting a match. Eventually, the algorithm identifies all the occurrences of all the patterns.

Number of patterns increases, the filtering becomes more efficient. Remember that will proceed with the third filter if at least one of the values in the vector register block passes the second filter. With a small number of patterns, we will seldom pass the second filter. When we do, it likely only have a single match, meaning that the rest of the values in the register are disabled and any computation performed for those values is wasteful work. Increasing the number of patterns results in more potential matches in the second filter and, consequence, less disabled values for the third filter and thus more useful work. In we measure this effect and show the average number of useful items inside the vector registers every time we reach the third filter. Clearly, with an increasing number of patterns, the is performed mainly on useful data and.

Filtering Parallelism: Gain better insights about the benefits, we measure the speedup gained in the filtering part in isolation. Figure 6 compares the filtering throughput of the scalar S-PATCH and V-PATCH, for pattern sets S1, S2, as well as the full pattern set (20K patterns). In the same figure, we also report the performance of the filtering, where we exclude the cost of storing the matches in the filtering phase in the temporary arrays. As we can see from the graph, the throughput of the filtering part is increased by up to a factor of 1.84x, on the small pattern set. Storing the matches of the filtering part in arrays comes with a cost; when it is removed, performance increases up to 2.15x for small pattern sets and up to 2.80x for the full pattern setWe have also evaluated the effectiveness of our approach on an architecture with a wider vector processing pipeline. The Xeonco-processor from Intel supports vector instructions that operate on 512-bit registers, thus able to perform two times more operations in parallel, As Xeon threads have much slower clock (1.1 GHz) and the pipeline is less sophisticated it is not surprising that the absolute throughput sustained by a single Phi thread is smaller than that of the single thread performance of the Xeon platform used in the previous experiments. When dealing with multiple streams in parallel, due to the higher degree of parallelism, the aggregated gain will naturally be higher. As Xeon threads have much slower clock (1.1 GHz) and the pipeline less sophisticated (e.g. there is no out of-order execution), it not surprising that absolute throughput sustained by a single Phi thread is smaller that the single thread performance of the Xeon platform used the previous experiments. When dealing with multiple streams parallel, due to the higher degree of parallelism, the aggregated gain will naturally be higherinteresting observation is that the DFC algorithm is slightly slower than on real data, where the number of matches in the input is significantly higher. In the original DFC algorithm, the filters are small and can easily fit L1 or L2 cache, and the hash tables containing the patterns are bigger, but still expected to fit L3 cache. In Xeon there is no L3 cache, so accesses to the hash tables in the verification phase are typically served by the device memory,

onautomation.

3. Pattern Matching Algorithms:

Pattern matching has been an active field of research for many years and there are numerous proposed approaches. Explained before one of the fundamental algorithms in the fields. There are variants of that decrease the size of the state transition by changing the way it is mapped in memory, but they come at an increased search cost, compared to the standard version of used in our evaluation. Other approaches apply heuristics that enable the algorithm to skip some of the input bytes without examining them at all, such as where a table is used to store information of how many bytes one can skip in the input. The main issue with these approaches is that they perform poorly with short patterns. For the problem domain investigated here, the patterns can be of any length and the algorithm must handle all of them gracefully. Moreover, in both algorithms, there is no data parallelism because there are dependencies between different iterations of the main loop over the input.

Proposed algorithm is fusion of Berry-Ravindran and Raita algorithms. The Berry-Ravindran bad character function proved in many cases effective in preprocessing phase. So same has been implemented with slight modifications. The searching location is almost remains asRaita algorithm. This new sectional algorithm compares the right most character of the pattern and the sliding window, if it is matched after that leftmost character of the pattern and the sliding windows are compared, if it also matched then the middle character of the pattern is mutually compared to the corresponding position of the inserting window, if it is matched then it's repeats again compare the characters string from the second last character of the pattern to the text window. In case of match or mismatch the skip of the window is achieved by the Berry-Ravindran shift value for the character that is placed next to the window.

In a standard problem, we are required find all occurrences of a pattern in a given input text, known single pattern matching. Suppose, if more than one pattern are matched against the given input text simultaneously, then it known multiple pattern matching. Whereas the method of single pattern matching is most widely used in network security environments. Multiple pattern matching algorithms can search multiple patterns in text at the same time. Absolutely it have a high performance and good practicability, and are more useful than the single pattern matching algorithms.

Matching patterns in a NIDS (Network Intrusion Detection System) a problem more specialized than the general patterns matching problem. In the context signature matching in a NIDS the signature database corresponds pattern set and the network packets, which the system scans, correspond to the text input for a pattern matching algorithm. Pattern-matching problems

4. Pattern Matching Algorithms to Intrusion Detection:

The Address spoofing is a method which is used to hide the real address of the sender of a network packet, particularly the intruder. This could be used to bypass the firewall and gain unauthorized access to a network or computer. Conditional firewalls have in-built mechanisms to avoid this fraud. They are, practically, not deceived by this kind of delegating address spoofing. But the protocol of address substitution, in itself, remains an urgent issue that needs to be addressed. An interested intruder can mask her own address with the address of a most likely trusted network address or with the address of a trusted host in the network and send packets containing malicious fills of data which may adversely affect the network computers and data. This method is different from the attack by trusted host and network problem since in this ruder case only the configure address of the trusted host is used rather than like masquerading as the trusted host in the former case. Network intrusion detection systems are widely used and heavily depended upon. The continued growth in both network traffic and intrusion signature databases makes the performance of these systems increasingly challenging and important. The impact of an under-performing intrusion detection system is severe: a passive system will drop large amounts of network traffic and may miss attacks, while an in-line system acts as a bottleneck to network performance. Therefore the performance weaknesses os intrusion detection systems create denial of service vulnerabilities Per-packet string matching is important to a class of applications beyond intrusion detection that includes firewalls scanning for viruses, layer seven switches doing web load balancing based on URLs and even cookies [1], and content distribution networks. Thus we believe that string matching over packet content is an important problem to be studied.

The work of describes a shift based algorithm that uses a network processor with a memory based hashing engine. It uses a prefix sliding window of length w, which shifts from the leftmost byte to the rightmost byte of the text. Their algorithm only supports simple patterns, with no identification of correlation of patterns. The algorithm using a rotational shift table (of size (28)w) that includes all possible w bytes combinations. At the time of the introduction of this algorithm A TCAM of size M, can be configured rows, where w is the TCAM width. The TCAM is populated in an offline process with two phases. In the first phase, we split the rule's signatures (patterns) to fit in the chosen w. Patterns longer than occupy more than one row.

The pattern in the first row is marked as the 'pattern's prefix". Short patterns, marked as prefixes, are padded to the size of w by using the TCAM "don't care" bit. Every TCAM row shows a corresponding shift value that states the number of bytes we can safely shift in the packet when a match occurs. In this phase all

TCAM rows have a shift value of 0. In the next phase, a set of shifted sub-patterns is created for each pattern prefix, by shifting the prefix to the right, losing the rightmost character and adding. Since a TCAM lookup returns the first matched row, the TCAM rows are ordered according to their shift values in an ascending order. The last row corresponds to the maximum shift value and contains w don't care bytes, thus providing the default match row manufacturer can easily implement the rotating effect by only storing the prefix pattern and utilizing internal clock cycles with a slightly modified TCAM logic.

Basically there are two primary types of IDS: A Network Intrusion Detection system (NIDS) transparently monitors network traffic, looking for patterns indicative of an attack on a computer or network device. Testing the network traffic, a network based intrusion detection system can detect suspicious activity such as a port scan or Denial of Service (DOS) attacks.

The General Infrastructure Design IDS, data acquisition module is responsible for capturing various types of hardware frames from network flow and handing these hardware frames to data pretreatment module and then the data pretreatment module strips off hardware frame heads and checks the integrity of messages, after that, according to the application protocols messages belong to, these hardware frames are sent to response protocol analyzing and processing modules respectively.

5. Conclusion:

In the new pattern matching algorithm is proposed in this paper, first sequence letters in the pattern string from low appearance probability to high appearance probability in natural, and then match one by one according to the algorithm, in this way, you can find as many mismatches as you can, thereby reducing the comparison times. Intrusion prevention system (IPS) is software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents; this can be put forward as a Future work. The current generations of IDS (HIDS and NIDS) are quite effective already; as they continue to improve they will become the backbone of the more flexible security systems we expect to see in the not-too-distant future. Finally, on the basis of BM algorithm, an intrusion detection system model is put forward.

6. References:

1. United States of Americe. US Govermnent Accountability Office (2015) (2015) Report on Cyber Security - Actions needed to Address Challenges facing Federal Systems. Washington: GAO-15-573T.
2. L. Morgan (2014), List of Cyber Attacks and Data Breaches in 2014. IT Governance, 23 Dec. Available from:http://www.itgovernance.co.uk/blog/list-of-the-hacks-and-breaches-in-2014/ [Accessed on 13 November2015]
3. M. Watson, (2014). JP Morgan suffers data breach affecting 76 million customers. IT Governance, 23 Dec. Available from:http://www.itgovernanceusa.com/blog/jp-morgan-suffers-data-breach- affecting-76-million-customers/ [Accessed on 14 November2015]
4. "Report and response regarding Leakage of Customers‟ personal Information." (10 September 2014). Last accessed on 17 February 2015, http://blog.benesse.ne.jp/bh/en/ir_news/m/2014/09/10/uploads /news_20 140910_en.pdf.
5. S Tobak. (18 December 2014). Fox Business. "3 Revelations from the Sony Hack" Last accessed on 29 January 2015, http://www.foxbusiness.com/technology/2014/12/18/revelations-from-sony-hack/.
6. A, Peterson. (5 December 2014). The Washington Post. "Why it's so hard to calculate the cost of the Sony Picture shack" Last accessed on 29 January 2015, http://www.washingtonpost.com/blogs/the-switch/wp/2014/12/05/why-its-so-hard-to-calculate-the-cost-of-the- sony-pictures-hack/.
7. Trend Micro Incorporated, (22 December 2014). Simply Security. "The Reality of the Sony Pictures Breach." Last accessed on 29 January 2015, http://blog.trendmicro.com/reality-sony-pictures-breach/.
8. R. Heady, G.F. Luger, A. Maccabe and M. Servilla, "The architecture of a Network Level Intrusion Detection System," Department of Computer Science, College of Engineering, University of New Mexico, 1990, pp.1-17.
9. R. Bace and P. Mell, "NIST Special Publication on Intrusion Detection Systems," Booz-Allen and Hamilton Inc, Mclean VA, 2001, pp.5-22.
10. R.A. Kemmerer and G. Vigna, "Intrusion Detection: A brief History and Overview," Computer, 2002 [supplement to security and privacy magazine], pp.27-30.
11. J. Allen, A. Christie, W. Fithen, J. McHugh and J. Pickel, "State of the practice of intrusion detection technologies," vol. CMU/SEI-99-TR-028, Canergie-Mellon Univ Pittsburgh PA Software Engineering Institute, 2000 , pp.3-23.
12. Abhaya, K. Kumar, R. Jha and S. Afroz, "Data Mining Techniques for Intrusion Detection: A Review," International Journal of Advanced Research in Computer and Communication Engineering, vol. 3, June 2014, pp. 6938-6941.
13. R.J. Manish, and H.T. Hadi, "A review of network traffic analysis and prediction techniques" unpublished.
14. S. Choudhury and A. Bhowal, "Comparative Analysis of Machine Learning Algorithms along with Classifiers for Network Intrusion Detection." In: Smart Technologies and Management for Computing,

Communication, Controls, Energy and Materials, IEEE, 2015, pp. 89- 95.

15. S.B. Kotsiantis, I.D. Zaharakis and P.E. Pintelas, "Machine Learning: a Review of Classification and combining Techniques," Artificial Intelligence Review, vol. 26, November 2006, pp.159-190.
16. Witten, E. Frank and M. Hall, "Data mining: Practical Machine Learning Tools and Techniques." 3rd ed., Burlington, MA: Morgan Kaufmann, 2011, pp5-40.
17. M. Masud, L. Khan and B. Thuraisingham, "Data mining tools for malware detection," Boca Raton, FL: CRC Press, Febuary 2012, pp. 15- 38.
18. O. Maimon and L. Rokach, (2010) "Data Mining and Knowledge Discovery Handbook," 2nd ed., New York: Springer Science & Business Media, 2010, pp.1-43.
19. A. Lazar, "Heuristic Knowledge Discovery for Archaeological Data using Genetic Algorithms and Rough Sets1," In Heuristic and Optimization for Knowledge Discovery, H. Abbass, C. Newton, & R. Sarker, Eds. Hershey, PA: Idea Group Publishing, 2002 , pp.263-278.
20. R. S. Wahono, "A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks," In Journal of Software Engineering, vol. 1, April 2015, pp.1-16.
21. P. Amudha, S. Karthik and S. Sivakumari, "Classification Techniques for Intrusion Detection–An Overview," In International Journal of Computer Applications, vol. 76, 2013, pp.33-40.
22. M. H. Haratian, "An Architectural Design for a Hybrid Intrusion Detection System for Database," unpublished.
23. A. I. Abubakar, H. Chiroma, A. S. Muaz and L.B. Ila, "A Review of the Advances in Cyber Security Benchmark Datasets for evaluating Data- Driven Based Intrusion Detection Systems," In Procedia Computer Science, vol. 62, 2015, pp.221-227.
24. K. Bajaj and A. Arora, "Dimension Reduction in Intrusion Detection Features using Discriminative Machine Learning Approach." In IJCSI International Journal of Computer Science Issues, vol. 10, 2013, pp. 324-328.
25. Y. Wahba, E. Elsalamouny and G. Eltaweel, "Improving the Performance of Multi-class Intrusion Detection Systems using Feature Reduction." In IJCSI International Journal of Computer Science Issues, vol. 12, issue 3, May 2015, pp.355-368.
26. A. Tesfahun and D.L. Bhaskari, "Intrusion Detection using Random Forests Classifier with SMOTE and Feature Reduction," In: Cloud & Ubiquitous Computing & Emerging Technologies, 2013 International Conference, 2013, pp.127-132.