International Journal of Engineering & Technology, 7 (4.19) (2018) 329-333

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Page Enabled FSM Model For Multi Rate- High Throughput Regex Pattern Matching System

S. Nagaraju[1], P.Sudhakara Reddy[2*]

Department of ECE, Jawaharlal Nehru Technical University, Ananthapuramu, Andhra Pradesh, India
**Corresponding Author Email: [2]snagaraju02@gmail.com*

Abstract

In recent years demands for high throughput NIDS systems are emerged with compatible Wildcard support for the detection of irregular patterns like ClamAV. In this work we presented a single-compound FSM based state transition controller for regular ASCII based patterns and counter enabled score generation model for regex patterns which contains both repeated characters and don't cares void segments. In many existing digital NIDS systems are token-stream based approaches were used with dedicated memory units to accommodate byte oriented matching with moderate network payload speed. The NIDS efficiency is largely depends on both intrusion byte size and the size of database. To mitigate this problem memory based digital NIDS system requires coordinated pattern matching. In this work, FSM based one hot state encoding model with bit wise state transition controller is proposed which gives both parallel task and high throughput payload validity check. Here during the payload monitoring if input segments are aggregated as tokens, the state transition controller is used to enable the counter for token model and state transitions are carried out based on the regex patterns received and the concurrent matches that are halted in parallel manner. To avoid clock synchronization over concurrent matching process and variable rate matching process page wise integration of each sub groups are carried out which is driven by ADPLL unit. The performance metrics of FSM state controlled payload monitoring is proved in terms of speed and memory efficiency over state-of-art-the-art methods. Here in our proposed NIDS system consumes lesser memory resources and it is verified through comparison with state-of-the-art methods.

Keywords: *Regex patterns, Parallel processing, FSM, PAGE, Strings, tokens.*

1. Introduction

In recent years the demands of network intrusion detection systems which includes both normal string types of intrusions and deep packet inspections based irregular regex types is emerged. And also the system has to deal with wide range of pattern lengths which requires both parallel task and efficient memory handling technique to accommodate a large number of intrusions and rule sets. In most cases the demands over high speed is meeting up with efficient string matching unit.

It has been investigated in many works and proved several methodologies to speed up the string matching process [1], [2], and for effective pattern storage, SRAM based static memory cells are most used [3]. However, there are several problems arises while introducing SRAM cells in NIDS system since off chip data processing will degrade the system performance. To solve these limitations over speed constrains data buffering and network routings are used to moderate the speed during memory reading [4]. On the other hand FPGA-devices are equipped with on chip block RAMs which are readily available, and provides higher flexibility and throughput rate for NIDS system [5], reconfigurable architecture will reduce overall hardware cost significantly. In addition, I/O programming will achieve satisfactory processing throughput. In general, FSM based pattern matching unit always requires large number of states to match a larger size patterns [6], [7], so allocating dedicated memory cells for each character is difficult task to accomplish.

In many previous works considerable performance measures are achieved with some significant memory reduction. Generally around 3.2K regex patterns in the virus database are composed of delimited symbols like *[8]. In [9] they presented hardware efficient memory architectures to detect both regular (strings) and the regex patterns. In [10] the relationship between the repeated sequence in various patterns and its frequency of occurrence weights are assigned to speed up the matching process. In this work, we proposed FSM memory architectures to accommodate both regular and irregular (ClamAV virus database) NIDS patterns.

Here we aim to meet two basic demands of next generation NIDS system such as memory requirement and parallel task. To implement the network detection unit with low hardware cost memory efficient one hot state encoding FSM unit is proposed. To achieve high throughput demands PAGE based bit wise matching unit is proposed. In general, if the patterns are stored as ASCII values then the matching speed is restricted and both complexity and computational time is linearly increased with input pattern length. Moreover, the cost of a memory unit is very high for regex kind of patterns; in general, external memory units are preferred to combat with irregular patterns. Here we also incorporate another major requirement of on-demand variable rate matching updating process using ADPLL based page enabled hardware architecture.

2. Regex Pattern Matching

2.1. Pattern Matching

Pattern matching process is always has trade off complexity over the performance results in terms of operating frequency. Our proposed method is based on the bitwise pattern matching through LSB to MSB matching approach. Instead of using the conventional ASCII-based matching, we transform the regexes into combined token-with multiple segments each contains several bytes of intrusion patterns. A token is considered here as sub-pattern and ID is extracted for each matched items along with the unified patterns. The tokens are represented as alphabet set and the input byte-stream is transformed into a bit-stream using FSM hardware units, where the number of bits will remain same for all the token-stream irrespective to the number of segments in the original payload. Whenever a token is detected, the FSM-based detection system will only need to make a finite number of state transitions.

2.2. Hardware based Approach

In recent years demands of digital NIDS systems to meet the throughput requirements are keep on rising. Fully digitalized NIDS systems can able to meet this demand but requires memory hungry device to hold the intrusion patterns, large size memory blocks, and also prominently need to carry out payload monitoring process as a fully integrated compound blocks in a synchronized manner. In General, finite state machines can only be used when pattern rule sets are evaluated as one hot states and ASCII enabled character matching is widely used which is not preferred for high performance device modelling for the following reasons:

- Each state in FSM machines holds intrusion patterns as ASCII character.
- Irrespective of NIDS rule sets used for payload monitoring, the payload validity check is possible only end of pattern matching process.

2.3. Pattern Matching

The regex features followed in the ClamAV pattern set are shown in Table 1. A major concern in the ClamAV format is the string repetition Y{p,q} (string Y repeats p to q times) and the Kleene closure c* (character Y repeats any number of times-usually unknown). Unlike ASCII-based pattern matching in network intrusion detection system, ClamAV virus signatures are mostly executable code. While both the number of internet users and network traffic rate has been increased every year the demands for high speed is unavoidable. And number of rules sets required for efficient pattern matching is also increasing in its own kind due to advancements in online markets and also its completion evolved over networks. In hardware based NIDS system, entire rule sets and patterns need to be stored in nonvolatile memory. The size of the ClamAV pattern is about 10 MB (excluding the wildcard bytes in the virus signatures). Designing memory unit for NIDS system for the entire set of ClamAV virus patterns is entirely complicated task to accomplish using on-chip memory units available in any FPGA devices even with block RAM based advanced FPGA families. Hardware based NIDS need to be run at very high speed.

2.4. Pattern Matching

The major disadvantage of identifying the virus signature using string based pattern matching is not that simple since the nominal variants of the hard-to-detect virus can easily pass through the anti-virus system. Simple identical way to deal with this kind of virus is that the register values in the matching code are masked off during the signature specification. Hence, both wildcards and nibbles are most commonly found in regex virus signatures. As shown in Table 1 regex virus signatures are mostly covered by nibbles, e.g. Worm.FlyStudio-20. Along with the nibbles, repeated string bytes are also quite common (e.g. Worm.Appaple-315). In some cases extreme repetition of string bytes are found which contain several instances of string bytes. Here all the possible combinations are encountered and the patterns are expanded accordingly.

Table.1: Example virus signatures [8]

Virus name	Signature
Exploit.HTML.ObjectType	*3c6f626a65637420747970 653d222f2f2f2f2f2 f2f2f2f2f2f2f*
Dos. Flip.Gen	*0ebb????????????b2??81c1????eb*
Worm.FlyStudio-20	*5?5?5?5?f85?5?0f83*

In conventional SRAM memory architecture byte information's are stored based on the address given and ASCII values are forwarded as 8 bit and also capable of only one data read at a time. Here this problem over memory efficiency is mitigated with input payload-driven SRAM controller and sub pattern enabled matching followed by bitwise comparison in states. As shown in Fig. 1. In bitwise payload monitoring concurrent matching is done but can be processed only 1 bit at a time from LSB to MSB in each FSM machine. Overall metrics given by bit based matching is used solve memory and throughout related problems associate with a ASCII based matching process as the latency is major cause with throughput rate.

Table 2: Regular expression in the ClamAV virus signatures [8]

Symbol	Meaning
??	match any byte, i.e. wildcard byte
c?	match a high nibble (the four high bits)
?c	match a low nibble (the four low bits)
(aa|bb|cc)	match aa or bb or cc
!(aa|bb|cc)	match any byte except aa and bb and cc (negation)
*****	match any no. of bytes (arbitrary displacement)
{n}	match n bytes (exact displacement by n bytes)
??	match any byte, i.e. wildcard byte

3. Proposed NIDS System Architecture

3.1 Segment Divisions into Tokens

Here if the input rule set is a string, then it is not subdivided and it is considered directly as bytes. In case of short segments patterns detected directly using page enabled FSM units. If a pattern length is large with regex contents, then the segmentation process is carried out in two steps, first sub division of patterns and second generation of trivial tokens. Then later on both these components are merged as a single-byte, and then its corresponding ID is generated as adjacent component which is distinct for each regex patterns.

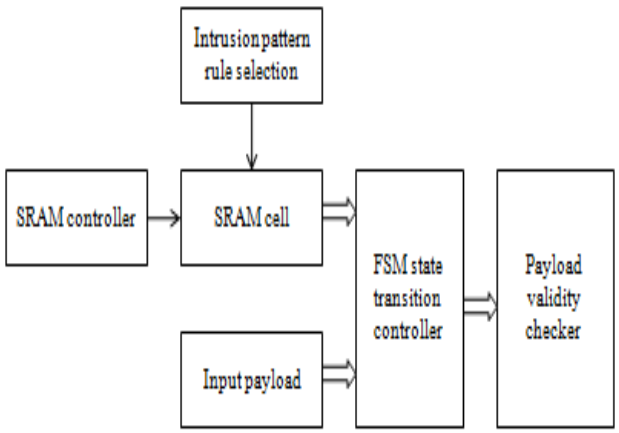

Fig. 1: FSM State Transition Controller

3.2 FSM State Transition Controller

Here FSM core enabled one hot state encoding model is incorporated for bit wise state transition and matching process. Each FSM state store one bit which is driven from LSB to MSB. All payloads are sub divided into sub groups and parallel matching process is accomplished with the help of 8 FSM machines as shown in Fig. 2. Since concurrent operations are evolved in all FSM machines and running in parallel manner whereas each pages monitoring unique intrusion patterns and each FSM machines monitoring payloads through bitwise comparison from LSB to MSB. If any of these bits comparison fails and appropriate matching process can be terminated through assertion, and successive bit comparison is bypassed, payloads are validated.

3.3 Algorithm

The input patterns are divided into sub-patterns and each page is equipped with unique group of intrusion patterns ASCII values. Through configurable rule set selection, page enabled NIDS systems were proposed with improved system performance where the new patterns or rule sets are easily integrated. Finally, bitwise pattern matching and FSM core enabled based bit transitions are carried out between input payload and intrusions pattern length on each page and the finally global match is triggered using partial matching vector (PMV) scheme as shown in Fig. 3.

```
1. Byte stream generation from input serial digital Bits
2. To detect the length of the pattern L
3. Partitioning into L1-L2-L3
   For (i = 1; i < 8; i =++)   // from LSB to MSB
     If (L1[i] = PAGE_1_FSM1[i] && L2[i]
       PAGE_1_FSM2[i] && L3[i] = PAGE_1_FSM3[i])
       Assert PMV1[i] = 1 // FSM holds database
     Else
   Skip to
4. Break points
       Assert PMV1[i] = 0 //  Partial Matching Vector
     End
   End
5. If (& (PMV1) = 1)
     MV = 1                  //  Matching Vector
   Else
     MV = 0
   End
6. Repeat the same process concurrently over
   Multiple PAGEs            // Each PAGE carries
                                different signatures
7. Generate score value for each matching for
   Each MV Vector            // ID will be given to Each
                                Signature
8. Decision making process
```

Fig. 2: Bit based Pattern Matching Algorithm

3.4 Regex System Architecture

The overall system architecture for patterns matching of the virus detection including regex patterns sets is depicted in Fig. 4. Hierarchical modules are built to detect the both types of tokens for regex patterns and general string patterns as shown in Fig 4. The input payload is converted into unified tokens, where the each byte is converted into bit streams. The most simplified regex signatures are divided into group of strings directly and complex signature are divided into multiple segments.

The token detection units are responsible for finding the type of tokens in the input payload stream, and the detected tokens are merged with bit stream. If it is a string pattern, then the FSM state transition will be initiated directly and end results are sent to the output directly, and its reference location from SRAM memory is synchronized for further processing by the memory controller. The FSM unit is responsible for to form segments. If the converted input segment is a NIDS pattern, then its ID is detected based matching scoring results. If the segment has only string part then ID is generated directly by the Scoreboard.

3.5 PAGE Enabled FSM Architecture

Here scalability is achieved using PAGE wised FSM matching units each will represent unique regex patterns which take the advantage of wildcards to realize the parallelism. The number of FSM used for bitwise computation will be updated based on new attributes included in the database and contributes linear performance increase. Compare to core processing element based approaches FSM based core processing consume lesser power and exploits high scalability. FSM also has potential merits of high performances and proportional to the higher frequency of operation. There are two main reasons to limit the maximum number of PEs. The proposed string matching has merits only for conventional pattern sequence. Here with FSM based bit holding states the constraints over a maximum number of core comparison required to match the regex patterns and the problem over regex pattern lengths is solved.

3.6 Data Driven Rate Controller

Bit based pattern matching is the optimal method for a string matching process as the overall performance rate is not depends on the pattern length and the pattern type. Here only 8 FSM machines are used and each one equipped with only three one hot states. In the case of page enabled model, the probability of early detection is also high which is also related to the ratio between network data rates over number of patterns in data base. However, parallel-processing is used to eliminate latency problems with improved throughput rate.

To handle pattern with large sizes it is divided as group of sub patterns. Consequently the incoming data is also divided multiple streams and each one is forwarded into the subgroup as a sequence of ASCII values. To improve the re-configurability and adopt network traffic data rate of each incoming payloads page enabled parallel matching schemes is proposed where the matching is done in bit wise manner.

It has following merits,

- Complexity is greatly reduced by replacing 8 bit comparator by 1 bit comparator.
- Maximum of 3 FSM state transitions are used and overall 8 FSM machines only required irrespective of patterns length.

4. Evaluation Results

Here PAGE enabled NIDS system is used for matching regex patterns with fully digitalized FSM based bitwise comparison and sub pattern wise grouping for better optimization in digital NIDS system. The hybrid memory controller is used to regulate both string and regex type patterns for all patterns matching process and its metrics over parallel matching is validated through exhaustive pattern inputs using functional simulation. Input stimulus driven FPGA synthesizer is used to prove the area efficiency in terms of memory requirements. In this paper, pipelined mechanism is not used to mitigate latency related problems and still achieves moderate throughput rate several Gbps which is validated through FPGA QUARTUS II EDA tool synthesizer and its design complexity reduction is also proven. However, the accuracy and complexity reduction is largely depending on patterns types used as shown in Table 2.

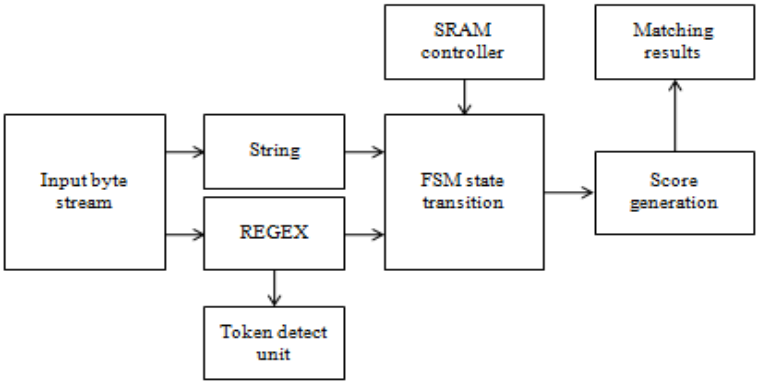

Fig. 3: Proposed Pattern matching system for regex patterns.

Table 3: Throughput Performance Comparison of PAGE enabled FSM model

NIDS type	Parallel task merits	Fmax(Hz)]
Multi-Stride String matching[11]	1 byte	230MHz
LUT model [12]	3 byte	144MHz
Proposed PAGE enabled model	6 byte	783MHz

Table 4: State-of-the-art comparison for memory efficiency of FSM based state holding model

Methods	Virus signatures	Memory requirements
Or, N.L et al.[12]	88.9K strings + 3.2K regexes	3.74 MB
Pao et al.[11]	82K strings	2.4 MB
Proposed FSM model	88.9K strings + 9.6K regexes	1.39 MB

Token based model not required any comparison to monitor the repeated sequence and don't care types in regex patterns during the counting process. Our FSM based core design consumes about 1.39 MB on-chip memories as shown in table 3 and table 4 for accommodating both types of patterns. Compared to the multi threading and LUT based approaches FSM engine model requires fewer memory spaces which equipped both the strings (88.9K) and regexes (9.6K). Hence, it is possible to detect both types of intrusions concurrently with our design.

4.1 ADPLL Synthesizer

Here, memory controller driven variable rate pattern matching system is validated for all types of regex signature sets and its efficiency through PAGE enabled bitwise parallel matching process is proved through exhaustive test bench simulation with ClamAV virus database version. Input payload driven all digital PLL synthesizer is also validated through hardware synthesis and its performance level is also proved to be in several GHz to supportnext generation networks (5G). As shown in Figure 3 the best optimal operating speed of FSM state transition is determined incoming payload rate of the projected samples and adoptively optimization is configured using variable delay lines for adapting the different bit-rate of input data streams.

This research advances the state-of-art in string matching process in a number of ways such as bit wise computation, ADPLL based synchronization for variable rate NIDS system and regex pattern matching process. An attempt was made to integrate all the above process to build PAGEs for combined auto string matching followed by token and ID generation. Integration of all the modules yields promising results in string matching process. The major contributions of the proposed FSM core enabled NIDS model are summarized as follows:

- Design of FSM technique for bit wise computation.
- Design of an ADPLL technique followed by FSM one hot state handling transition leads some prominent features such as promising delay and memory reduction.
- PAGE enabled integrations to achieve a framework for scalability and re-configurability as shown in Fig.4.

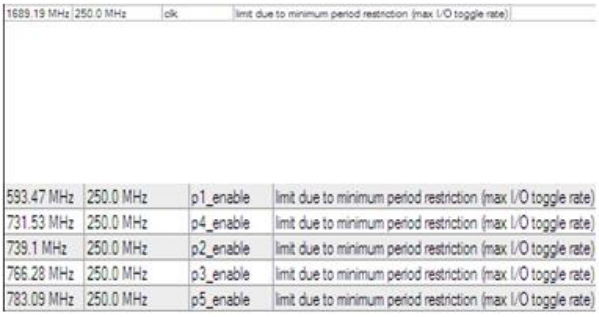

Fig. 4: Adoptive variable payload match bit-rate report

5. Conclusions

Here we verified the functionality proposed FSM-based parallel string matching scheme with minimized memory requirements for compound string and regex patterns. The problem of moderate pattern lengths with wild cards is successfully mitigated by dividing the segments into small byte oriented tokens with a fixed length. The memory-efficient architectures were proposed for both staring matching and complex regex pattern matching and the actual throughout requirements are maximized with PAGE based parallel processing. The total memory requirements for the real time regex rule sets is compared with state-of-the-art methods it is concluded that the proposed FSM based model is useful for reducing memory requirements and ADPLL equipped multi rate matching is useful for moderate throughput rate in any NIDS engines.

References

[1] Dharmapurikar, S.; and Lockwood, J.W. (2006). Fast and scalable pattern matching for network intrusion detection systems. IEEE Journal on Selected Areas in Communications, 24(10), 1781-1792.

[2] Aho, A.V.; and Corasick, M.J. (1975). Efficient string matching: an aid to bibliographic search. Communications of the ACM, 18(6), 333-340.

[3] Tuck, N.; Sherwood, T.; Calder, B.; and Varghese, G. (2004). Deterministic memory-efficient string matching algorithms for intrusion detection. In INFOCOM Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, 2628-2639.

[4] Xu, J.; Kalbarczyk, Z.; Patel, S.; and Iyer, R.K. (2002). Architecture support for defending against buffer overflow attacks. In Workshop on Evaluating and Architecting Systems for Dependability.
[5] http://www.stoimen.com/blog/2012/03/27/computer-algorithms-brute-force-string-matching.
[6] Boyer, R.S.; and Moore, J.S. (1977). A fast string searching algorithm. Communications of the ACM, 20(10), 762-772.
[7] Sourdis, I.; and Pnevmatikatos, D. (2003). Fast, large-scale string matches for a 10Gbps FPGA-based network intrusion detection system. In International Conference on Field Programmable Logic and Applications, 880-889.
[8] PCRE – Perl Compatible Regular Expressions, http://perldoc.perl.org/perlre.html
[9] Pao, D.; Wang, X.; Wang, X.; Cao, C.; and Zhu, Y. (2011). String searching engine for virus scanning. IEEE Transactions on Computers, 60(11), 1596-1609.
[10] Gupta, A.; Thakur, H.K.; Gupta, T.; and Yadav, S. (2017). Regular Pattern Mining (With Jitter) On Weighted-Directed Dynamic Graphs. Journal of Engineering Science and Technology, 12(2), 349-364.
[11] Pao, D.; and Wang, X. (2012). Multi-stride string searching for high-speed content inspection. The Computer Journal, 55(10), 1216-1231.
[12] Or, N.L.; Wang, X.; and Pao, D. (2016). MEMORY-based hardware architectures to detect ClamAV virus signatures with restricted regular expression features. IEEE Transactions on Computers, 65(4), 1225-1238.