IOSR Journal of Engineering (IOSR JEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 74-77

Memory Efficient High Throughput Pattern Matching Engine for Nids System

D.Betsha Kerlin A.Heerasri B.Hemamalini Ms. P.Sasireka,M.E
UG Student, Final year ECE Department, S.A.Engineering College.
Assistant Professor, Department of Electronics and Communication, S.A.Engineering College.

Abstract : *High speed and always-on network access is becoming common place around the world, creating a demand for increased network security. Network Intrusion Detection Systems (NIDS) attempt to detect and prevent attacks from the network using pattern-matching rules in a way similar to anti-virus software. For the low-cost hardware-based intrusion detection systems, this project work proposed a memory-efficient parallel string matching scheme. In order to reduce the number of state transitions, the finite state machine tiles in a string matcher adopt bit-level input symbols. Long target patterns are divided into sub patterns with a fixed length; deterministic finite automata are built with the sub patterns. Using the pattern dividing, the variety of target pattern lengths can be mitigated, so that memory usage in homogeneous string matchers can be efficient. As an extended work, here adding the all-digital Phase locked loop (ADPLL) for the multi-rate clock synchronization. The input intrusions are divided into pages and each of the incoming pages will be in variable rate. An adaptive reconfigurable PLL based clock divider is used for variable rate pattern match and gated clock system through early mismatch detection is verified through exhaustive test bench simulation.*

I. Introduction

The proliferation of Internet and networking applications, coupled with the wide-spread availability of system hacks and viruses have increased the need for network security. Firewalls have been used extensively to prevent access to systems from all but a few, well defined access points (ports), but they cannot eliminate all security threats, nor can they detect attacks when they happen. State full inspection firewalls are able to understand details of the protocol that are inspecting by tracking the state of a connection. They actually establish and monitor connections for when it is terminated. However, current network security needs, require a much more efficient analysis and understanding of the application data. Content-based security threats and problems occur more frequently, in an everyday basis. Virus and worm inflections, SPAMs (unsolicited e-mails), email spoofing, and dangerous or undesirable data, get more and more annoying and cause innumerable problems. Therefore, next generation firewalls should provide deep packet Inspection capabilities, in order to provide protection from these attacks. Such systems check packet header, rely on pattern matching techniques to analyze packet payload, and make decisions on the significance of the packet body, based on the content of the payload.

1.1 Motivation

Network Intrusion Detection Systems (NIDS) perform deep packet inspection. They scan packet's payload looking for patterns that would indicate security threats. Matching every incoming byte, though, against thousands of pattern characters at wire rates is a complicated task. Measurements on SNORT show that 31% of total processing is due to string matching; the percentage goes up to 80% in 2 the case of Web-intensive traffic. So, string matching can be considered as one of the most computationally intensive parts of a NIDS and in this thesis we focus on payload matching. Many different algorithms or combination of algorithms have been introduced and implemented in general purpose processors (GPP) for fast string matching, using mostly SNORT open source NIDS rule-set. However, intrusion detection systems running in GPP can only serve up to a few hundred Mbps throughput. Therefore, seeking for hardware-based solutions is possibly the only way to increase performance for speeds higher than a few hundred Mbps. Until now several ASIC commercial products have been developed. These systems can support high throughput, but constitute a relatively expensive solution. On the other hand, FPGA-based systems provide higher flexibility and high throughput comparable to ASICs performance. FPGA-based platforms can exploit the fact that the NIDS rules change relatively infrequently, and use reconfiguration to reduce implementation cost. In addition, they can exploit parallelism in order to achieve satisfactory processing throughput.

Several architectures have been proposed for FPGA-based NIDS, using regular expressions (NFAs/DFAs), CAM, discrete comparators, and approximate filtering techniques. Generally, the performance results of FPGA systems are promising, showing that FPGAs can be used to support the increasing needs for network security. FPGAs are flexible, reconfigurable, provide hardware speed, and therefore, are suitable for implementing such systems. On the other hand, there are several issues that should be faced. Large designs are complex and therefore hard to operate at high frequency. Additionally, matching a large number of patterns has high area cost, so sharing logic is critical, since it could save a significant amount of resources, and make designs smaller and faster.

1.2 Scope Of This Thesis

Since string matching is the most computationally intensive part of an NIDS, our proposed architectures exploit the benefits of FPGAs to design efficient string matching systems. The proposed architectures can support between 3 to 10 Gbps throughput, storing an entire NIDS set of patterns in a single device. In this thesis, we suggest solutions to maintain high performance and minimize area cost, show also how pattern matching designs can be updated and partially or entirely changed, and advocate that brute-force solutions can offer high performance, while require low area. Techniques such as fine-grain pipelining, parallelism, partitioning, and pre-decoding are described, analyzing how they affect performance and resource consumption. This thesis proposes a pattern matching algorithm that reduces total memory requirements by sharing common infixes of target patterns. For the pattern
identification, a state should contain its own match vector with a set of bits, where each bit represents a matched pattern in the state.

Even though the information of shared common infixes was stored in match vectors, the number of shared common infixes was limited by the size of the match vectors. In addition, throughput could decrease due to the modified state transition mechanism. The memory requirements for match vectors were reduced by relabeling states and eliminating the match vectors of non-output states. By sharing common infixes of target patterns or relabeling states and eliminating the match vectors of non-output states, the memoryusage in the match vectors could be efficient. However, the variety of target pattern lengths is another serious problem in achieving regularity and scalability with low hardware cost. Each pattern consists of multiple character codes, where the number of character codes is defined as the pattern length. According to the rule sets, the distribution of pattern lengths could be different from each other. In addition, the variation of pattern lengths in each rule set is irregular. If target patterns are to be mapped onto multiple homogeneous string matchers, memory usage cannot be balanced without considering different pattern lengths.

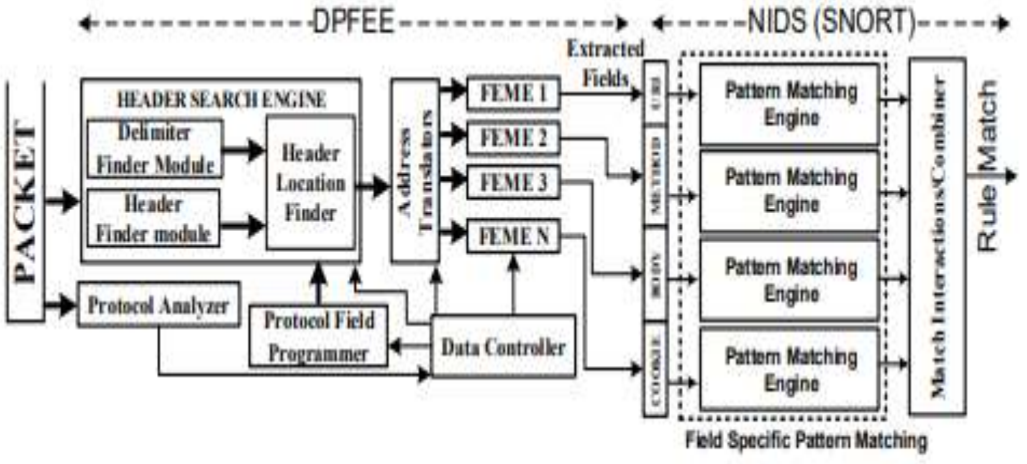
In order to reduce the memory requirements of the DFA-based string matching engine, this proposes a memory-efficient parallel string matching scheme using the pattern dividing approach and its hardware architecture for the pattern identification. Long target patterns are divided into sub-patterns with a fixed length; therefore, the variety of target pattern lengths can be mitigated. By balancing memory usage between the string matchers, unused memory area in homogeneous string matchers decreases. Moreover, the number of shared common states increases due to both the reduced length and the increasing number of sub-patterns, compared with the cases of the string matching with long target patterns. For each string matching, DFAs are built with bit-level input symbols for the bit splitting in order to reduce the number of state transitions from each state. For identifying the original long target patterns, the successive matches with sub-patterns are detected using the proposed two-stage sequential string matching engine. Experimental results show that memory requirements decrease on average by 47.8 percent and 62.8 percent for selected rules Snort and Clam AV, compared with several existing bit-split string matching approaches.

1.3 Dissertation Outline

The rest of the thesis is organized as follows: the next chapter presents brief description of NIDS, offers some statistics about the patterns contained in a NIDS, and present some performance results of software-based NIDS.
Chapter 3, describes hardware-based NIDSs, previous FPGA-based pattern matching architectures, and commercial products.
Chapter 4, proposes a memory-efficient parallel string matching scheme using the patterndividing approach and its hardware architecture for the pattern identification.
Chapter 5, hardware and software used, and
Chapter 6, we present the results and conclusions of this work and discuss
future extensions.

II. PROPOSED SYSTEM

In this work we proposed hardware efficient VLSI architectures to detect the complex NIDS patterns based on the information reduction approach. Here we preprocessed input bytes to transform the byte-oriented matching problem to a token-based matching problem. The input byte stream is converted into a tokenstream using dedicated hardware units which can perform parallel computations for high throughput rate. A NIDS pattern contains one or more segments will be subdivided into multiple non-trivial tokens. Finally the token-stream is processed by a NFA-based aggregation unit to determine the virus to be found. Here FEMEs are finite state machines (FSM) designed to extract field values within a header data of the application layer as shown FEMEs search the header data sequentially. The extraction time is a function of the length of the header and number of bytes searched in a clock cycle.



III. Modules

Simulating wave form
Verilog Module- debugging
Analysis of timing characteristics

Divided Patterns :

A target pattern and a set of its k sub-patterns, which are obtained after dividing the target pattern, are denoted as Pi and Qi ¼ fSPi1; SPi2; . . . ; S Pikg, respectively. The subscript i is the index of the target pattern. A set of k sub-patterns Qi will be called the quotient vector of Pi. The fixed length of k sub-patterns is denoted as f. If the length of a target pattern is shorter than f, the target pattern does not need to be divided, so the pattern is defined as the short pattern. The remnant pattern Ri represents a suffix or residual sub-pattern of the target pattern Pi that succeeds the quotient vector of Pi.

Memory-Based Bit-Split Dfa:

DFA is an FSM where there is one and only one transition to a next state according to each pair of state and input symbols. DFA can be represented with a five-tuple: a finite set of states (Q), a finite set of an initial state (q0), and a set of output states (F Q). The identification index of a target pattern is an individual keyword used to distinguish the target pattern match. The memory requirements of DFA are proportional to the size of Q and.

Pattern Identification:

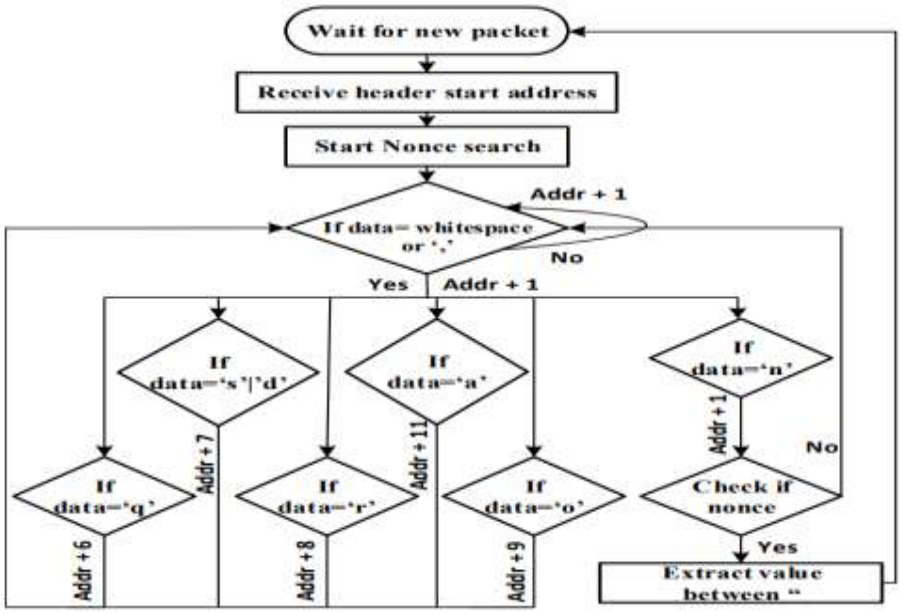
For each target pattern, a unique identification index should be provided in order to distinguish its pattern match from other pattern matches. If multiple target patterns are mapped onto a DFA, it is possible that a target pattern can be a subpattern of other target patterns. For example, it is assumed that four target patterns {“abc,” “abcd,” “ac,” “bcd”} are mapped on a DFA, where target pattern lengths range from 2 to 4. The fourth target pattern is a suffix of the second target pattern. If the second target pattern is matched, the fourth target pattern is always matched, but not vice versa.

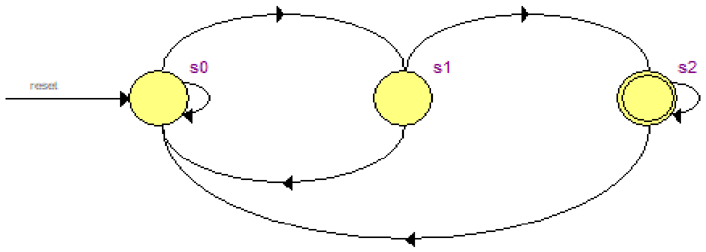
Virus Detection Engine:

The overall organization of the virus detection engine is depicted in Fig. 4.1 Hardware modules are built to detect the three types of tokens, namely string tokens, PACX tokens and MX-NFA tokens. The input byte stream is converted into a token stream, where the number of tokens is much less than the number of bytes. An example based on virus Worm.Allaple-11 is shown in Fig. 4.1 to illustrate idea. The simplified virus signature is divided into 3 tokens. The token detection units are responsible for finding the corresponding type of tokens in the input byte stream, and the detected tokens are merged into a token stream. A token can be part of a segment, a single token segment, or a pattern. If it is a pattern, then the token ID is sent to the output interface directly, otherwise the token ID together with its reference location and control flags are inserted into the token queue for further processing by the AU.

Nfa Token Detection Unit:

The NFA regex detection method was initially developed to support intrusion detection. In principle NFA can also be used to detect the full virus patterns but the cost can be very high for long patterns and patterns with large exact-count and range count. In this study, the virus signatures are divided into tokens, and the NFA is used to process tokens that cannot be detected by other methods. The design of the NFA detection unit is refined for virus detection. In the Snort regex, the counting block corresponds to the repetition of a symbol by the given number of times. Here each FEME has additional error handling capability to check for malformed headers.





IV. Conclusion

Here we verified the functionality proposed novel Deep Packet Field Extraction Engine-based parallel string matching scheme with minimized memory requirements for NIDS virus database. The problem of various pattern lengths with wild cards can be mitigated by dividing long target patterns into tokens with a fixed length. The memory-efficient architectures were proposed for both string matching and complex NIDS pattern matching which can reduce the total memory requirements. Considering the reduced memory requirements for the real rule sets, it is concluded that the proposed matching scheme is useful for reducing total memory requirements of parallel string matching engines.