Remodeling the NetFPGA architecture for content processing and filtering

Bokil Kanchan
Centre for Development of Advanced
Computing (C-DAC)
68, Electronics City
Bangalore, India
91-80-28523300

kanchan@ncb.ernet.in

ABSTRACT

The networks today operating at speeds of gigabits per second are getting more and more vulnerable to network attacks. There is a need for a smart, high speed, reliable and scalable system to prevent network intrusions. This paper presents our approach to provide a robust solution by remodeling NetFPGA reference architecture for deep packet inspection such that the packet processing delay is highly negligible. We discuss our implementation of devising high speed FSMs in a pipelined architecture that has been validated for maintaining throughput of 1 Gbps with a set of SNORT based signatures.

General Terms

Algorithms, Security, Design.

Keywords

SNORT, intrusion prevention system, NetFPGA.

Conference'10, Month 1–2, 2010

1.INTRODUCTION

A typical network-based intrusion prevention system sits on the perimeter of a network to be protected. The IPS sensor logs the alerts by performing either (a) Simple pattern matching of known signatures against the contents of individual packets or (b) Stateful pattern matching of an entire session by tracking the state of each communication transaction. It also blocks the malicious traffic within the network keeping it secure.

An IPS sensor constitutes of a large database of signatures to be matched, a buffer for incoming traffic and a rule engine.

Given a rule set of signatures, it is the rule engine that decides the performance of the sensor. Performing the signature matching at line rate effectively, with a huge set of patterns has always been a challenging task while designing an IPS sensor. The biggest bottleneck lies in number of patterns the sensor can operate against the speed at which it performs.

In this paper we present a solution that comprises of dividing Snort based signatures [1] into subsets of protocol based signatures accompanied by high speed NetFPGA card [2] with a pipelined architecture. We discuss our approach, design specifications, data flow and the architecture for IPS sensor implemented on NetFPGA platform. The key part is to choose only those signatures (to be matched by rule engine) whose protocol category matches with the protocol field of incoming packets. For e.g., for a packet with protocol field tcp (06), match only the tcp signatures and ignore the remaining ones.

The major contributions of this paper are, (a) devising an FSM technique for hardware based content matching (b) remodeling the reference architecture of NetFPGA for intrusion analysis (c) validating our solution by using Layer 4-7 equipment.

Further sections are organized as follows: Section 2 briefly reviews the related work. Section 3 sheds light on design

specifications. Section 4 discusses system architecture in detail. Section 5 shows the results obtained upon the tests conducted. The paper is concluded in Section 6.

2.RELATED WORK

There have been various algorithms including Boyer-Moore [3], Aho-Corasick [4] and Wu-Manber [5] for content matching algorithms. We analyzed the techniques used in these algorithms to model them on fpga.

We studied various mechanisms used till now to implement string matching algorithms on variety of platforms including FPGAs [6] and ASICs [7] . We also find research papers on Bloom filters [8] to speed up the string matching algorithms. We also find ideas presented for regular expression matching [9]. The paper [10] discusses use of content addressable memory for intrusion detection.

3.THE DESIGN

Our approach :

Our goal is to develop a system that sits at the edge of a LAN and protects the internal network from being attacked. We have used the SNORT signatures to design the system. There exist many network architectures and mechanisms such as honeypots and demilitarized zone to know the suspicious activities. We took the SNORT ruleset as the skeleton for its commercial applicability and suitability for any small to large networks.

SNORT signatures :

SNORT is an open source software by Sourcefire which refers to a large signature database for intrusion detection. SNORT [1] releases its signature set which is referred by many intrusion prevention systems. It also updates the database for newly found network attacks.

A SNORT signature typically describes various parameters the packet has to be tested upon, like the protocol, port number, from or to internal network, content in the payload. It also describes the action to be taken upon matching of all the parameters within one signature. There exists a huge set of signatures to be tested for incoming packet with more that one such contents with much complexity. The challenges involved in designing the system are,

(a) Performing the signature matching at line speed.

(b) Including the entire large set of signatures in the system.

Design Specifications :

An application for intrusion detection or prevention system running on top of an operating system generally adds delays in the process of content matching. As we go down from application level to kernel level and that to hardware level, we observe improvement in the throughput of the system. We found NetFPGA a suitable platform to develop the sensor by implementing a pipelined architecture in it.

As per the goals of the project, the system should have two interfaces to enable the connection to external and internal networks. Please refer to Figure 1.

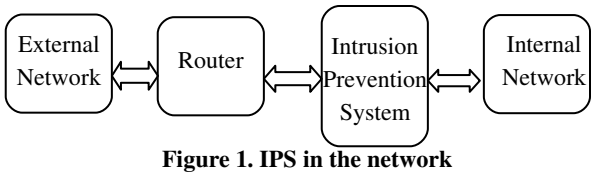

Figure 1. IPS in the network

All the traffic from internal network is passed to external one and vice-versa. And with the addition of malicious activity sensor in between, the data flow will be as shown in Figure 2.

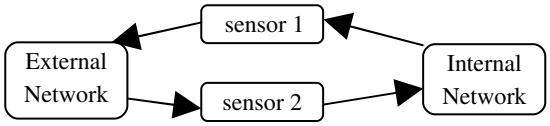

Figure 2. Individual sensor for each interface

This basic design comprises of an individual sensor for each interface by avoiding the servicing of one port at a time thus improving the throughput, i.e. implementation of only one sensor for both interfaces and servicing one interface at a time reduces the throughput to half. Also, by implementing a dedicated sensor for each interface enables us to divide the signature set into two parts, (a) rules for the traffic entering into the internal network (b) rules for the traffic leaving the internal network.

4.SYSTEM ARCHITECTURE

The reference router

The NetFPGA board has four Ethernet ports. The card is inserted in the PCI slot of the host PC. The NetFPGA driver module is inserted in the kernel on the host PC. The four Ethernet ports are identified by four unique device IDs in the kernel. The NetFPGA card has four CPU ports each for four Ethernet ports. There exists a separate queue for Receive and Transmit sections (Figure 3) on each of Ethernet and CPU ports.

Figure 3 is the reference architecture diagram by NetFPGA developer's guide.

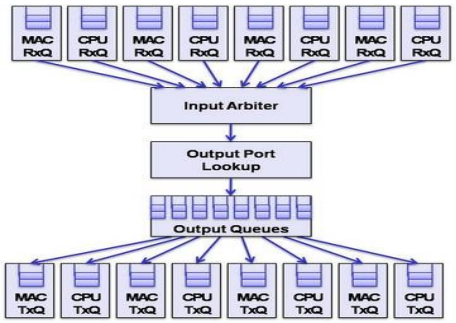

Figure 3. Reference Router

The architecture maintains a module header (Figure 4) for each received Ethernet frame to direct the frame correctly to its destination.

Destination Port	Length in Word	Source Port	Length in Bytes

Figure 4. Internal Module Header

As shown in Figure 3, the input arbiter services each of the eight Rx queues in sequence and passes the data to output port lookup. The output port lookup reads the destination port in module header to put the frame in appropriate output queue out of eight output queues. The output queues are connected to appropriate Tx queues i.e. CPU and Ethernet queues.

The NetFPGA Ethernet ports (PHY) are capable of operating at 1Gbps. Within the card, 64 bit data is operated at the internal clock speed of 125MHz. This produces 8 Gbps operating capability of the NetFPGA, which gets equally distributed over eight Rx queues by the round-robin input arbiter, to produce ultimate 1Gbps of Ethernet speed.

Modification in the data flow

Consider the data flow from ethernet port nf2c0 to ethernet port nf2c1. The incoming data in Rx queue of port 0 has to be forwarded to Tx queue of port 1 and vice versa. Once the frame enters in the Rx queue, the start of any frame resets all the finite state machines to a predefined state. As further data within the frame is received, the packet decoder stores the fields in the ethernet frame and also knows the protocol. According to the protocol, respective FSMs are activated.

A copy of incoming packet is maintained in the block RAM. The frame information is updated in the frame information table. Once the entire frame is done with the FSMs, the decision is written in the information table. The Figure 6 explains the content matching module.

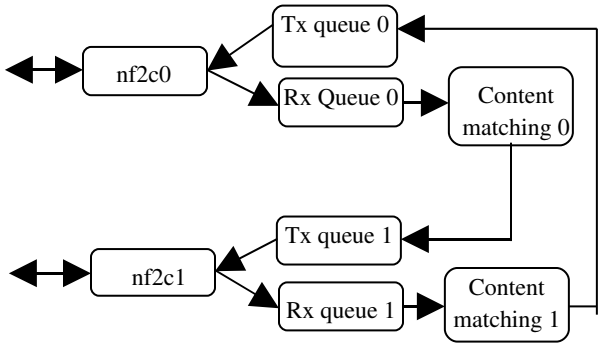

Figure 5. Modified Reference router

With the end of frame and "rdy" signal from all the FSMs, the decision block reads the entry to pass/block the frame. All the traffic to be passed is copied from block RAM to the Tx queue with pipelining the next frame to get processed. If any of the pattern is matched, that particular frame is dropped.

Content matching module

The malicious activity sensors for each interface will sit in between this design.

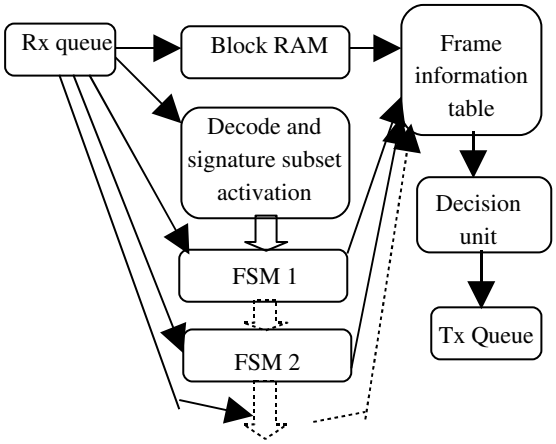

Figure 6. Block diagram for Content matching module

The content matching module serves as the controlling element for the system. It generates the control signals for Block RAM by identifying the ethernet frame. It comprises of submodules as Block RAM controller, packet decoder and subset activator, frame information table and set of signatures described in FSMs.

Flow of data within the interfaces

The malicious activity sensor has to have two Ethernet ports as per the project requirement. It should have two CPU ports to pass the attack information to the host PC. Since this comes to four Rx queues and four Tx queues, the reference design is halved by instantiating each of nf2_mac_grp and cpu_dma_queues twice

instead of four times. Since we need to pass the traffic within two interfaces plainly, we omitted the user_data_path.

This led the elimination of the input arbiter, output port lookup and output queues.

This modification simply connects the two Ethernet ports on the card to each other with the content matching module sitting within them.

Packet Decoder and signature subset activator

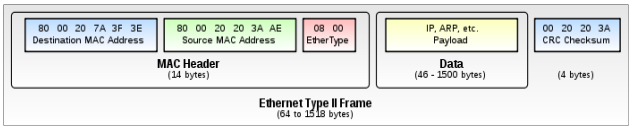

Figure 6. Ethernet Frame Format

The packet decoder is instantiated one for each of the two interfaces. It decodes the ethernet frame 802.3 (Figure 6) and stores the protocol field values in predefined registers. As soon as the protocol of incoming packet is known, the signature_subset_activator activates all the signatures to be matched against, for that particular protocol.

This avoids redundant verification of fields for each signature in the database.

Frame information table

Once the content matching module receives one Ethernet frame, it is sent to the set of signatures. Since the frame can be passed to appropriate Tx queue only when entire frame is matched against all the signatures, the frame has to be copied in a buffer. This is done by maintaining a copy of incoming frame into a bock RAM and also an information table (Table 1) to know the start address, end address in the RAM, the decision bit whether to pass the frame or block it and a valid bit to reuse the table entries. The controlling FSM unit writes the decision in the appropriate entry and raises the valid bit. The decision unit who is waiting on the valid bit, reads the decision to pass/block and makes the valid signal zero. (Decision : 0 for pass, 1 for block).

Table 1. Frame information table

Entry	Start address	End address	Decision	Valid
0	9'b000000100	9'b000011000	0	0
1	9'b000011001	9'b000100010	1	1

Set of Signatures

As discussed before, the rule engine is the crucial part in the system design, which can be time costly operation. To avoid any packet inspection delay to be added in the data flow, fast content matching mechanism has to be implemented.

We have developed a unique deterministic finite state automata for each of the signature pattern. Each FSM (finite state machine) has a valid line which is activated by the packet decoder and signature subset activator. As a new frame begins, each 64 bit data is matched against the signature and a state is maintained according to the match (Figure 7). The number of states within a signature is equal to the number of characters within the content. Figure 8 demonstrates the flow diagram for FSM. The example shows state machine designed for a signature with content "abc".

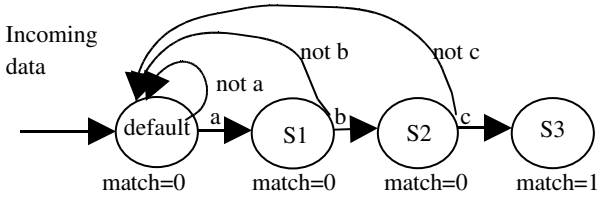

Figure 7. State digram for pattern "abc"

The keywords like depth, within, etc in the SNORT signatures are implemented in form of a counter for incoming bytes and maintaining the current state.

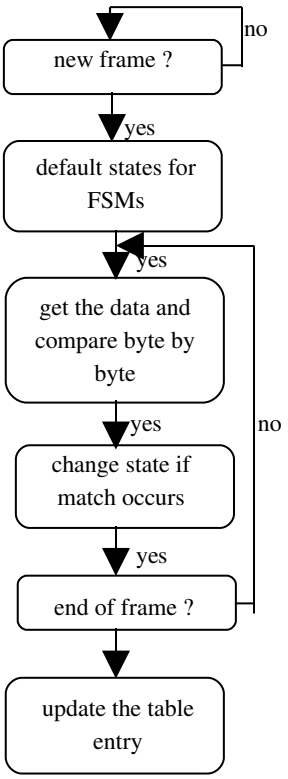

Figure 8. Flow digram for typical FSM

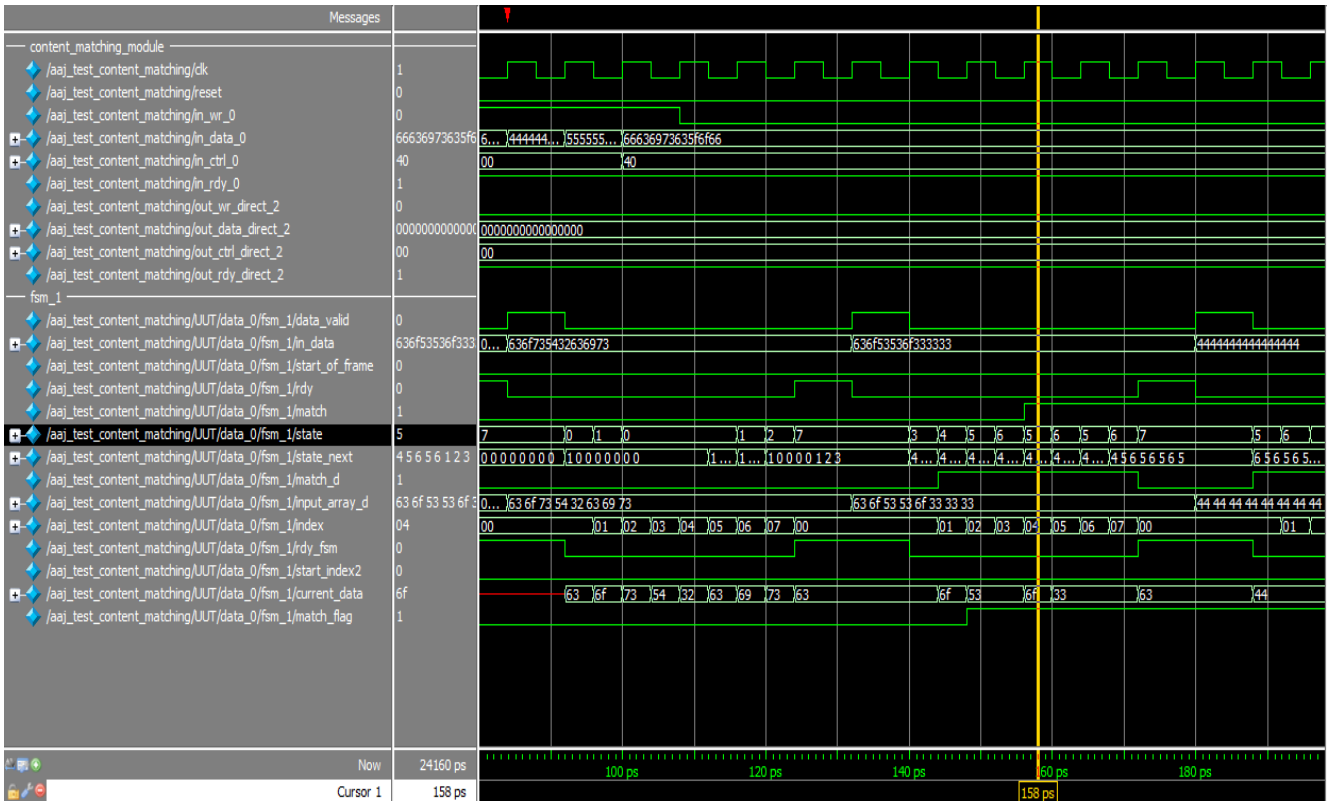

Figure 9. Modelsim waveform sample

We can see in Figure 9, which is Modelsim waveform sample, the progression of index pointing to 8 bytes of incoming frame and the match signal upon successful comparison on a pattern.

The NetFPGA can operate on 64 bits of data at 125 MHz clock frequency, which reflects the packet processing speed of 8Gbps. Since the PHY operate at 1Gbps, it is necessary for the Tx queue and Rx queue to process packet data at 1Gbps. Thus the content matching module can take 8 clock cycles per incoming 64 bit data to match with the known signature.

5.VERIFICATION AND RESULTS

The functionality test was carried out in two ways, (a) By passing real time traffic samples on NetFPGA IPS system and configuring the Spirent Layer 4-7 equipment to see the results. (b) By running test cases on Modelsim. The aim was to verify the system operation for a particular set of signatures. The results show that the system forwards good traffic to other end and blocks all the malicious traffic.

To perform the throughput test, Spirent Layer 4-7 equipment was used to pump traffic within the system. Various traffic patterns were used for testing. The best case can be described as passing of non-malicious traffic and running all FSMs to end. This was achieved by maintaining the throughput upto 1Gbps.

Verification Environment

The Spirent Layer 4-7 equipment was configured to pump HTTP traffic starting at the rate of 1Mbps (Figure 10). To perform the throughput test, the HTTP traffic rate was increased upto 970 Mbps gradually and the test was conducted for about an hour.

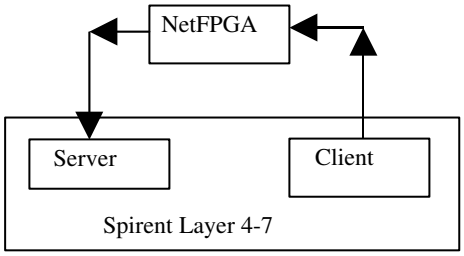

Figure 10. Verification Environment

To perform the accuracy test, various test cases were written and the run on Modelsim. Also, to ensure the working of the system at high incoming rates, the payload in HTTP traffic on Spirent equipment was added with malicious traffic with known percentage like 50% malicious payload. Then the output traffic was measured with respect to the incoming traffic and the accuracy was verified.

6. CONCLUSION

We present the intrusion prevention system architecture implemented on NetFPGA platform. The high speed pattern matching mechanism and pipelined architecture enable the system throughput maintained to that of NetFPGA board. We propose the distribution of SNORT signature set to eliminate the unnecessary comparisons.

The paper provides a mechanism to store and forward the traffic upon decision in an organized way.

We look at IP defragmentation and tcp reassembly as our future work which will enable us to include some more signatures into the design. On the similar lines, stateful packet inspection improves the accuracy by doing the inspection on entire session, which seems to be future experimentation.

7.ACKNOWLEDGMENTS

We thank Mr. Subramanian for giving us an opportunity to implement our ideas on a new platform, the NetFPGA. Special thanks to our team for timely help in the development and testing.

8.REFERENCES

[1] Snort. http://www.snort.org/, 2003.

[2] John W. Lockwood, Nick McKeown, Greg Watson, Glen Gibb, Paul Hartke, Jad Naous, Ramanan Raghuraman, and Jianying Luo. NetFPGA - An Open Platform for Gigabit-rate Network Switching and Routing. IEEE International Conference on Microelectronic Systems Education (MSE'2007), June 3-4, 2007

[3] R. S. Boyer and J. S. Moore. A fast string searching algorithm. Communications of the ACM, 20(10):762–772, 1977.

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

[5] S. Wu and U. Manber. Fast text searching: Allowing errors. Communications of the ACM, 35(10):83–91, 1992.

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

[7] L. Tan and T. Sherwood. Architectures for Bit-Split String Scanning in Intrusion Detection. IEEE Micro: Micro's Top Picks from Computer Architecture Conferences, January-February 2006.

[8] Sarang Dharmapurikar, John Lockwood. Fast and Scalable Pattern Matching for Content Filtering. Proceedings of Symposium on Architectures for Networking and Communication Systems (ANCS) , Oct 2005.

[9] Sarang Dharmapurikar, and John Lockwood. Fast and Scalable Pattern Matching for Network Intrusion Detection Systems. IEEE Journal on Selected Areas in Communications : Oct 2006, Volume : 24, Issue : 10, pp. 1782 - 1792.

[10] Michael Attig and John W. Lockwood. SIFT : Snort Intrusion Filter for TCP. 13th Annual Proceedings of Hot Interconnects, Stanford, CA, August 17-19, 2005.