International Journal of Engineering Research and General Science Volume 5, Issue 2, March-April, 2017
ISSN 2091-2730

HARDWARE BASED HIGH SPEED NETWORK SECURITY SYSTEM

Augusta Angel. M, Julie Antony Roselin. J, Poorna Lekha. S
Dept of ECE, VV college of Engineering, Tisaiyanvilai

Abstract— In the field of networking, role of network security is immense. It is a very vital tool which provides the security against various external and internal threats in any network. This paper provides solution for multi gigabit network security through packet content scanning mechanism. The most of network firewalls are software based, they runs sequentially. To achieve packet inspection at GigaHz line speed, hardware implementation of parallel Bloom filters are employed. Each Bloom filter is for specified length of hashed signature, the hash function provides compact data base. If the incoming packet signature matches with data base, then the corresponding Bloom filter indicates the presence of harmful content. The Improved Counting Bloom Filter (CBF) architecture is used in this work instead of previous SRAM array. The proposed implementation utilizes an array of up/down linear feedback shift registers (LFSR) and local zero detectors, which have better energy, speed and area constraint. The overall throughput achieved is about 3 Gb/sec. The proposed CBF based security system has been implemented with Xilinx FPGA.

Keywords— CBF, LFSR, FPGA, SRAM, PHP, Hash Function, Network security.

Introduction

The network security is most important concern for any organization; nowadays entire data base is handled by group of computers that are connected through less privacy network. Therefore skilled persons can illegally access the network for valuable information like military secrets, banking details etc; they also destroy it through malicious attack. This unauthorized access is limit by Firewall, which is uses software based packet inspection which execute sequentially.

There is a class of packet processing applications that inspect packets deeper than the protocol headers to analyze content. For instance, if a packet payload carries certain malicious Internet worms or computer viruses, then the network security applications must drop such types of packets . Most payload scanning applications have a common requirement for string matching [9]. For example, the presence of a string of bytes (or a *signature*) can identify the presence of a harmful content. Well-known Internet worms such as Nimda, Code Red and Slammer propagate by sending malicious executable programs identifiable by certain byte sequences in packet payloads. Such applications must be able to detect strings of various lengths starting at arbitrary locations in the packet payload.

Packet inspection applications [15], when deployed at router ports, must operate at line speeds. With networking speeds doubling every year, it is becoming increasingly difficult for software-based packet monitors to keep up with the line rates. These changes have underscored the need for specialized hardware-based solutions that are portable and operate at line speeds.

Proposed design describes a hardware-based technique using Counting Bloom filters. Without degrading network throughput, Counting Bloom filters can detect strings in streaming data . A Bloom filter consists of a set of signatures that represents a data structure . In order to compute multiple hash functions on each member of the set, a bloom filter stores that set of signatures . This technique uses a database of strings to check for the membership of a particular string.

One of the important property of this data structure is that the computation time for performing the membership of a particular string is independent of the number of strings stored in the database.The memory used by the data structure scales linearly with the number of strings stored in it. Furthermore, the amount of storage required by the Bloom filter doesn't depend on the string length.

Related works

SNORT is a type of software used for the purpose of deep packet inspection [9] Measurements on SNORT show that 31% of total processing is due to string matching; the percentage goes up to 80% in the case of web-intensive traffic .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 serve only up to few hundred throughput .therefore, seeking for hardware-based solutions possibly the only way to increase performance for high speeds higher than few hundred Mbps.

Network intrusion detection systems (NIDS) attempt to detect attacks by monitoring incoming traffic for the suspicious contents [4]. They collect data from network, monitor activity across the network, analyze packets, and report any intrusion behavior in an automated fashion. Intrusion detection systems use advanced matching techniques (i.e.Boyer and Moore, Aho and corasic, Fisk and Varghese) on network packets to identify the known attacks.

They use simple rules to identify possible security threats, much like virus detection software, and report offending packets to the administrator for further actions .NIDS should be updated frequently, since new signatures may be added or others may change on a weekly basis. NIDS rules usually refer to the header as well as to the payload of a packet .

Proxy server method breaks the traditional client/server model. Clients are required to forward their requests to a proxy server instead of the real server. After the proxy receives those requests, it will forward them to the real server only if the requests meet a predefined security policy. The real server receives the requests from the proxy, which forces it to believe that the proxy is the real client. This allows the proxy to concentrate all requests and responses from clients and servers. But the worst problems are it is expensive and time consuming to write code for proxy servers.

Another technology used for performing network security was packet-filtering firewalls. It was implemented by using access control lists (ACL) embedded in routers. Access control was one of the primary concerns of the early age of commercial use of the Internet. Because routers are the connection point between internal and external networks, they are used as access control devices . Simple packet filters analyze each of the packets passing through a firewall which matches a small part of their contents against previously defined groups of access control rules. The basic limitations were as follows:

• Because they analyze individual packets, they could not identify security violations that can only be visualized by screening more of the traffic flow.

• Very little information from the packets was analyzed, avoiding the identification of several problems that could only be seen in the application layer.

• The rules were static, creating many security problems for screening protocols that negotiate part of the communication options, like ports and connections, on the fly (the FTP service is a classic example).

• In general, router ACLs, implemented through command-line parameters, are harder to manage than rules created in easy-to-use graphical user interfaces.

The hardware-based technique using Bloom filters, [1] which can detect strings in streaming data without degrading network throughput. A Bloom filter is a data structure that stores a set of signatures compactly by computing multiple hash functions on each member of the set. This technique queries a database of strings to check for the membership of a particular string.

The design takes multiport SRAM as memory [7] for hash table data base maintenance, which use hashed address for adding, removing and query of elements. The SRAM access path can be broken down into two components: the decoder, which is the portion from the address input to the word line, and the output multiplexer, which is the portion from the cells to the output.

The read access as it determines the critical timing for the SRAM. For the read access, the address input is decoded to activate a specific word line. The decoder typically employs the divided word line structure, where part of the address is decoded to activate the horizontal global word line and the remaining address bits activate the vertical block select line. Energy dissipation in an SRAM are dynamic energy to switch the capacitance in the decoders, bit lines, data lines and other control signals within the array, the energy of the sense amplifiers and the energy loss due to the leakage currents.

Counting bloom filters

The updating of signature database by inserting and deletion of stings is difficult task in Bloom filter; in order to overcome this, Counting Bloom Filter (CBF) is adopted, shown in fig 1.

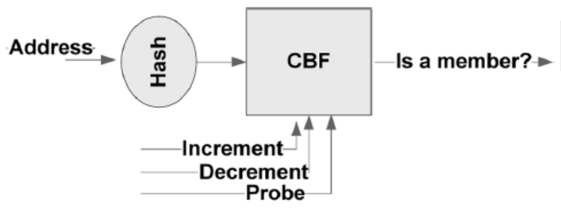

Fig.1 Counting Bloom

The significance of hardware *counting bloom filters* (CBFs) is that they improve the power, delay and performance of various complex multiprocessor or multi-core systems. CBFs have been also utilized to improve the scalability of load/store scheduling queues and to reduce instruction replays by assisting in early miss determination at the L1 data cache.

In the existing *SRAM-Based CBF Implementation*, both delay and energy suffer as updates require two SRAM accesses per operation. In order to avoid accesses over long bitlines ,an array of up/down counters with local zero detectors are built. In this way, Low power CBF operations would be localized and there would be no need to read/write values over long bitlines.

For the CBF, the actual count values are not important and we only care whether a count is “zero” or “nonzero.” Hence, any counter that provides a deterministic up/down sequence can be a choice of counter for the CBF. L-CBF consists of an array of up/down LFSRs with embedded zero detectors.

Instead of synchronous up/down counters with the same count sequence length , L-CBF uses up/down LFSRs that offer a better delay, power, and complexity tradeoff . Compared to S-CBF , L-CBF significantly reduces energy and delay but results in more area. Though the increase in area is a minor concern in modern processor designs, compared to most other processor structures, the CBF provides the abundance of on-chip resources and the very small area compared to most other processor structures. (e.g., caches and branch predictors).

A maximum-length-bit LFSR sequences through 2^{n-1} states. It goes through all possible code permutations except one. The LFSR [5] consists of a shift register and a few embedded XNOR gates fed by a feedback loop. Each LFSR has the following defining parameters:

•Width, or size, of the LFSR (it is equal to the number of bits in the shift register);

• Number and positions of *taps* (taps are special locations in the LFSR that have a connection with the feedback loop);

• Initial state of the LFSR which can be any value except one (all ones for XNOR feedback).

State transitions proceed as follows. The non-tapped bits are shifted from the previous position. The tapped bits are XNORed with the feedback loop before being shifted to the next position. The combination of the taps and their locations can be represented by a polynomial. Fig 2 shows an 8-bit maximum-length Galois LFSR, its taps, and polynomial.

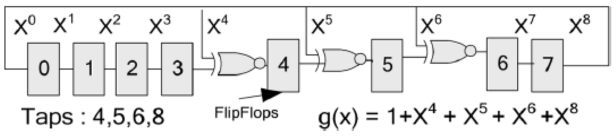
X^0 X^1 X^2 X^3 X^4 X^5 X^6 X^7 X^8
0
1
2
3
4
5
6
7
Taps : 4,5,6,8
FlipFlops
$g(x) = 1+X^4 + X^5 + X^6 + X^8$

Fig.2 LFSR Structure

By appropriately selecting the tap locations it is always possible to build a maximum-length LFSR of any width with either two or four taps. Additionally, ignoring wire length delays and the fan-out of the feedback path, the delays of the maximum- length LFSR is independent of its width (size).

System Overview

This system relies on a predefined set of signatures grouped by length and stored in a set of parallel Bloom filters in hardware [8]. Each Bloom filter contains signatures of a particular length. The system uses these Bloom filters to monitor network traffic and operate on strings of the corresponding length from line data. The high-level organization of L-CBF is shown in Fig 3 L-CBF includes a hierarchical decoder and a hierarchical output multiplexer.

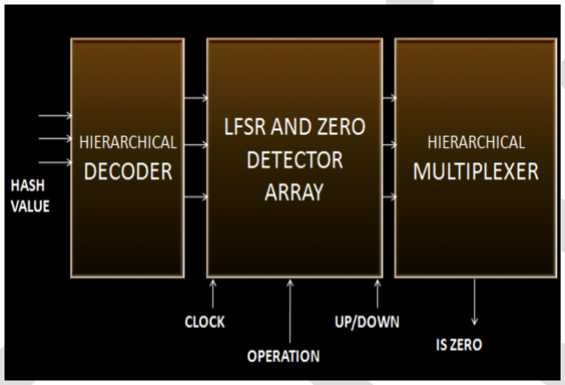

Fig.3 Architecture of L-CBF

The core of the design is an array of up/down Linear Feedback Shift Registers and zero detectors. The L-CBF design is divided into several partitions in which each row of a partition consists of an up/down LFSR and a zero detector.

L-CBF accepts three inputs and produces a single-bit output that is noted as *is-zero*. One of the input *operation select* identifies the type of operation: INC, DEC, PROBE, and IDLE. The input *n* bit *address* specifies the address in question and the input *reset* is used to initialize all LFSRs to the *zero* state. The LFSRs utilize two phase clocks generated internally from an external clock.

The core of the design is an array of up/down Linear Feedback Shift Registers and zero detectors. The L-CBF design is divided into several partitions in which each row of a partition consists of an up/down LFSR and a zero detector.

L-CBF accepts three inputs and produces a single-bit output that is noted as *is-zero*. One of the input *operation select* identifies the type of operation: INC, DEC, PROBE, and IDLE. The input *n* bit *address* specifies the address in question and the input *reset* is used to initialize all LFSRs to the *zero* state. The LFSRs utilize two non-overlapping phase clocks generated internally from an external clock.

To minimize the energy-delay product, the decoder is used for decoding [6] the address . The decoder consists of a predecoding stage, a global decoder to select the appropriate partition, and one local decoder per partition. Each partition has a shared local *is-zero* output. A hierarchical multiplexer collects the local *is-zero* signals and provides the single-bit output *is-zero* .

The system tests each string for membership [11] in the Bloom filters. If it identifies a string to be a member of any Bloom filter, the system then declares the string as a possible matching signature. Such strings receive further probing by an analyzer, that determines if the string is indeed a member of the set or a false positive.

The window length will vary with signature length; here six byte length window is used with four Bloom filters. The signatures are grouped based on their length and they are allocated to unique Bloom filter, the length of filters are three, four, five and six. The incoming serial bits are continuously inspected by parallel Bloom filters; control signal from PHP(*Packet Header Processor)* enables the bloom filters whenever the payload arrive the window.

The packet length is calculated by Packet Header Processor (PHP) through reading total length field at IP header. There is 16 bit representation of total length that gives length of IP header, TCP header and Payload .

The length of payload is extracted which is used to enables the control signal to parallel Bloom filter. Therefore the inputs are applied to parallel Bloom filter only at payload part of each TCP/IP packet flows through streaming data window .

The counting sequences are used in PHP for tracking the fixed header length and variable payload length. There are three counting, first one count up to Total Length field at IP header then exact payload length is calculated. Second count is up to TCP header termination and third count is equal to payload length that is calculated previously. The hierarchical decoder is used for decoding [6] the address x. The decoder consists of a predecoding stage, a global decoder to select the appropriate partition, and a set of local decoders, one per partition. Each partition has a shared local *is-zero* output. A hierarchical multiplexer collects the local *is-zero* signals and provides the single-bit output *is-zero* .

The system tests each string for membership [11] in the Bloom filters. If it identifies a string to be a member of any Bloom filter, the system then declares the string as a possible matching signature. Such strings receive further probing by an analyzer, which determines if the string is indeed a member of the set or a false positive.

The window length will vary with signature length; here six byte length window is used with four Bloom filters. The signatures are grouped based on their length and they are allocated to unique Bloom filter, the length of filters are three, four, five and six. The incoming serial bits are continuously inspected by parallel Bloom filters; control signal from PHP(*Packet Header Processor)* enables the bloom filters whenever the payload arrive the window.

The packet length is calculated by Packet Header Processor (PHP) through reading total length field at IP header. There is 16 bit representation of total length that gives length of IP header, TCP header and Payload .

The length of payload is extracted which is used to enables the control signal to parallel Bloom filter. Therefore the inputs are applied to parallel Bloom filter only at payload part of each TCP/IP packet flows through streaming data window .

The counting sequences are used in PHP for tracking the fixed header length and variable payload length. There are three counting, first one count up to Total Length field at IP header then exact payload length is calculated. Second count is up to TCP header termination and third count is equal to payload length that is calculated previously.

Results and discussions

The parallel Bloom filter is allowed to inspect the in coming packet at desire time only, the payload streaming time is calculated by PHP then control signal is used to enable the Bloom filters. The zero detector produce valid output only when operation is set to low. During insertion and deletion signal operation is set to high, the up or down signal select whether insertion or deletion to be takes place. The LFSR is enabled by during this process. The figure 5 shows parallel Bloom filter outputs 'v1, v2, v3, v4' are zero detectors output signal, which indicates presence and absence of signatures. The serial data is applied to streaming window from packet switching line.

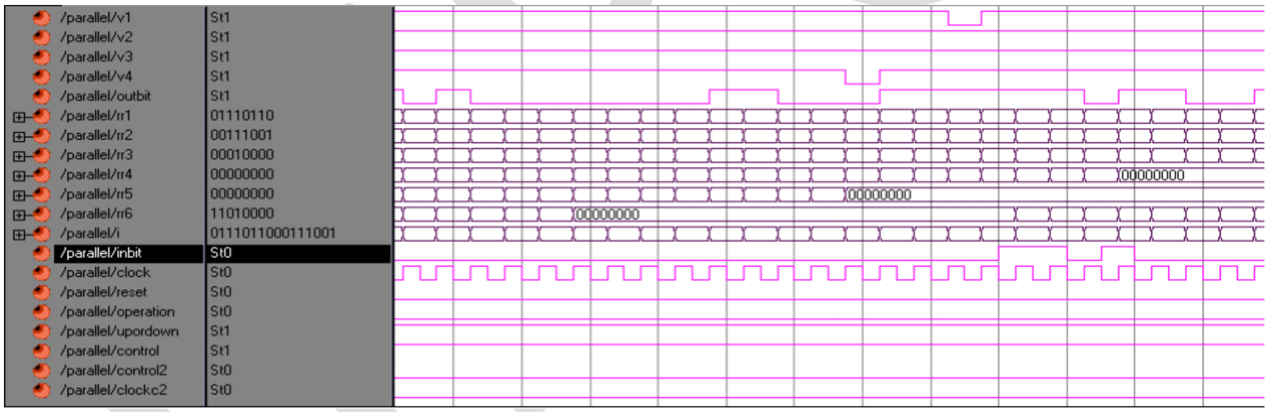

Fig.5 Parallel Bloom filter waveform

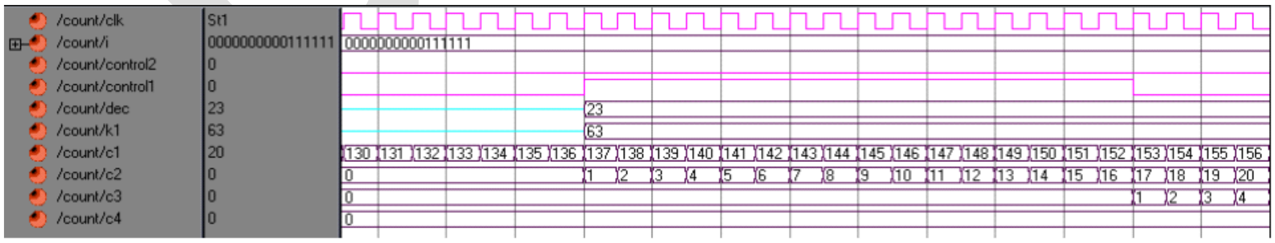

Fig. 6 Packet Header Processor waveform

The 'rr1, rr2, rr3, rr4, rr5, rr6' are byte of data at window, which are applied to Bloom filters with clock. There is control over data flow through window during Bloom filter Programming.

The figure 6 shows the packet length calculation performed by PHP, here total length is 63 bytes in which header is 40 bytes remaining 23 bytes are payload. The control2 signal is set to high for payload length.

Conclusion

The proposed Counting Bloom Filter based Network security system has been developed using Verilog and the functionality is verified using modelsim simulator. The system is implemented with Xilinx Spartan 3E FPGA. The system ensures that local network security against virus attacks, based on signature matching by packet content inspection through parallel Bloom filter. The performance is improved by LFSR based Counting Bloom Filter (CBF) design in terms of delay, power and area. The signature database updating is simplified to up or down count of corresponding LFSR. The system throughput is improved to line speed through parallelism. This design can efficiently implement in FPGA, in order to achieve real time virus detection that inspect all internet protocol packets. The existing system maximum throughput is 150Mb/sec which is improved by hardware implementation up to 3 Gb/sec.

REFERENCES:

[1] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd S. Spruill ,John W.Lockwood "*Deep Packet Inspection Using Parallel Bloom Filters*" IEEE Computer Society pp- 52-61, January - February 2004

[2] Elham Safi, *Andreas Moshovos,and* Andreas Veneris*"L-CBF: A Low-Power, Fast Counting Bloom Filter Architecture" IEEE Transactions on Very Large Scale Integration* (VLSI) systems, vol 16, no. 6, June 2008

[3] Ahmadi M. and Wong S.,*"Hashing Functions Performance In Packet Classification"*, Proceedings of International Conference on the Latest Advances in Networks (ICLAN-2007), pp 127-132, 2007.

[4] B. L. Hutchings and R. Franklin and D.Carver "*Assisting Network Intrusion Detection Reconfigurable Hardware"* Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02) 2002

[5] Mircea R. Stan, Member, IEEE, Alexandre F. Tenca, and Milos D. Ercegovac *"Long And Fast Up/Down Counters"* IEEE Transactions on computers, vol. 47, no. 7, July 1998

[6] Bharadwaj S. Amrutur and Mark A. Horowitz, Fellow, IEEE *"Fast Low-Power Decoders For RAMs"* IEEE Journal Of Solid-state Circuits, Vol. 36, No. 10, October 2001

[7] Bharadwaj S. Amrutur and Mark A. Horowitz "*Speed And Power Scaling Of SRAM'S"* IEEE Transactions On Solid-state Circuits, Vol. 35, No. 2, February 2000

[8] Arun Kumar S P *"High-Speed Signature Matching In Network Interface Device Using Bloom Filters"* International Journal of Recent Trends in Engineering, (Academy Publisher) Vol 1, No. 1, May 2009

[9] Alok Tongaonkar, Sreenaath Vasudevan, and R. Sekar*, "Fast Packet Classification for Snort by Native Compilation of Rules"* published at 22nd Large Installation System Administration Conference (LISA '08) ,2008

[10] Harwayne Gidansky J., Stefan D and Dalal I., *"FPGA-Based Soc for Real-Time Network Intrusion Detection Using Counting Bloom Filters"* IEEE Southeast Conference, Atlanta,2009.

[11] Sarang Dharmapurikar, Praveen Krishnamurthy, David E. Taylor "*Longest Prefix Matching Using Bloom Filters"* SIGCOMM'03, August 25–29, 2003.

[12] Ioannis Sourdis, Dionisios N. Pnevmatikatos and Stamatis Vassiliadis "*Scalable Multigigabit Pattern Matching for Packet Inspection"* IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 16.No 2,Pp 156-166, February 2008

[13] Taskin Kocak and Ilhan Kaya "*Low-Power Bloom Filter Architecture for Deep Packet Inspection"* IEEE Communications Letters, Vol. 10, No. 3,Pp 210-212 , March 2006

[14] Gianni Antichi, Domenico Ficara, Stefano Giordano, Gregorio Procissi, and Fabio Vitucci **"***Counting Bloom Filters for Pattern Matching and Anti-Evasion at the Wire Speed"* IEEE Network pp 30-35 January/February 2009

[15] Sarang Dharmapurikar and John Lockwood "*Fast and Scalable Pattern Matching for Network Intrusion Detection Systems*" IEEE Journal on Communications, Volume-24, Issue-10, pp-1781to1792, Oct 2006.