I.J. Information Technology and Computer Science, **2014, 07, 47-55**
Published Online June 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.07.07

High Performance Network Security Using NIDS Approach

Sutapa Sarkar
Department of Electronics and Communication, MVJ College of Engineering, Bangalore, India
Email: sutapasarkar11@rediffmail.com

Brindha.M
Department of Electronics and Communication,MVJ College of Engineering, Bangalore, India
Email: brindha_mo47@yahoo.com

***Abstract*—** Ever increasing demand of good quality communication relies heavily on Network Intrusion Detection System (NIDS). Intrusion detection for network security demands high performance. This paper gives a description of the available approaches for a network intrusion detection system in both software and hardware implementation. This paper gives a description of the structure of Snort rule set which is a very popular software signature and anomaly based Intrusion Detection and prevention system. This paper also discusses the merit of FPGA devices to be used in network intrusion detection system implementation and the approaches used in hardware implementation of NIDS.

***Index Terms*—** Network Intrusion Detection System, Snort, FPGA

I. INTRODUCTION

Network Intrusion detection system can be described as the process of identifying and taking necessary actions against malicious activities targeted to network and computing resources. A network intrusion detection system should continuously monitor the traffic crossing the network and compare with a previously known set of malicious activities or look for statistical deviation of the system under surveillance from its normal behavior. Aim of network security is to protect the device from unauthorized and potentially harmful activities such as denial of service attacks (forcing the targeted computers to reset or to consume its resources so that it is not able to provide the intended service), port scans or attempt to crack into computers by monitoring network traffic. Network connected devices are very often susceptible to exploitation. The Intrusion detection system (abbreviated as IDS) placed in the network should be able to sense the unusual activity and alert the administrators. A set of well defined rules eg. Snort and Bro are used to identify network events that are other than expected.

The goal of modern network traffic is to provide a high speed good quality communication keeping up with the demand of ever increasing data usage. Deep packet inspection with regular string matching is a very common method of network intrusion detection. Implementation of Signature based network Intrusion Detection System (NIDS) requires to match a predefined string or predefined pattern that is already identified as harmful to the network. As the IDS should inspect the data packets at the rate of data connection, a very high performance is required for the IDS string matching operation. Also the rule set gets regularly updated with the evolution of fresh attacks. Hence the hardware system used to implement NIDS should have the feature of dynamic reprogramming. Both of these features of high network traffic collection ability and dynamic reprogramming is supported by FPGA devices. Hence they are suitable candidates for hardware implementation of NIDS. But the high network traffic collection ability is not matched by the device frequency. Hence like multi core parallelization of microprocessors it is mandatory approach to implement parallelism in FPGA based NIDS traffic analysis.

The network intrusion detection system can be placed at a choke point such as the company's connection to a trunk line [1], or can be placed on each of the hosts that are being monitored to protect from intrusion. Intrusion, incident and attack are three terms that we often come across while discussing Intrusion Detection System. Table I. gives a brief description of these terms[2].

One further challenge in designing these systems is the lack of availability of dataset that is representative of normal traffic. The normal traffic is generally corrupted with different port scans and denial of service activities, hence these patterns also may become a part of normal state system behavior for anomaly detection activities [3].

A NIDS should have the following desirable features [4]:

- System should be fault tolerant and run with the minimal human supervision.
- The NIDS should not be susceptible to attacks from the intruder
- NIDS should not interfere with the normal operation of the system.
- It should be possible to reconfigure the NIDS over time with the changing rules and security policies of the network.
- NIDS should be portable to different architectures making it easy to deploy.
- NIDS should be general to detect different types of attacks and should have as less number of false positives as possible.

This paper serves the purpose of a reference that can be used to understand NIDS and their applications. It also describes the available types of NIDS in both software and hardware implementations. The remaining part of the paper is organized as follows: Section 2 gives a brief description of the different types of Intrusion Detection System, classifying them based on their placement in the network or based on the methodology adopted by them to detect a possible intrusion. Section 3 gives a description of the software based NIDS approach with focus on Snort rule set which is most popular software based NIDS used worldwide. Section 4 gives a description of the hardware based NIDS and the possible approaches to implement them followed by conclusion in Section 5.

II. Types of Intrusion Detection System

In software based NIDS approach the IDS are software systems that are specially designed with the aim of identifying and hence help to prevent the malicious activities and security policy violations. IDS can be classified into two main categories: analysis approach and placement of IDS. Analysis approach consists of misuse detection and anomaly detection.

2.1 Misuse Detection

This approach uses pattern matching algorithm to look for some known misuses. They have very low false positive (IDS generates alarm when no attack has taken place) rate. Since they depend on comparing the incoming traffic with a known set of malicious strings they are unable to identify novel attacks. Hence a high false negative (Failure to detect an actual attack) rate is observed. The number of disallowed patterns now has reached the order of the thousands making the computation a rather difficult task. Signature based Network Intrusion Detection System is a commercial success. Snort is a well defined rule set that uses signature, protocol and anomaly based detection methods. In section 3 we go through a detailed description of Snort rule set as this rule set is directly influencing the design of FPGA based NIDS.

2.2 Anomaly Detection

This approach makes decisions based on normal network or system behavior using statistical techniques [5]. This approach monitors network traffic and compares it against an established baseline of normal traffic profile. The baseline characterizes normal behavior for the network - such as the normal bandwidth usage, the common protocols used. This approach is able to identify novel attacks that are yet unknown and hence undetectable by signature based NIDS. The main disadvantage of anomaly detection method is that it may generate a large number of false positives.

An anomaly detection technique consists of two steps[3]: the first step is called training phase wherein a normal traffic profile is generated; the second phase is called anomaly detection, where the learned profile is applied to the current traffic to look for any deviations. The anomaly detection techniques are as follows: statistical methods, data-mining methods and machine learning based methods. In statistical methods it is assumed that a variation of the traffic in terms of volume of number of packets indicates attack, like bandwidth flooding attack. But if the attacker keeps traffic parameter bellow a certain level this method will not work. Incorrect combinations of port numbers and devices indicate attack. Then NIDS should alert the administrator or user regarding detection of anomalous traffic.

It depends on the situation what can be considered as normal and what can be considered as anomaly. But some possible examples are as follows. Consider the case when a user logs off the system 20 times, although the usual frequency is 1 or 2 times. This increased frequency can be considered as anomaly. Again consider the scenario of an office where if the system gets used at midnight which far beyond the normal business hour, then it can be considered as anomaly. The decision of what is anomaly and what is not is highly subjective.

Based on placement in the network IDS can be classified as host based and network based systems.

2.3 Host Based System

This type of IDS is present on each host that needs monitoring. These are able to determine if an attempted attack is successful and can detect local attacks. It is possible to analyze the traffic and the effect of any attack can be analyzed very accurately. But it's difficult to deploy and manage them if the numbers of hosts that are to be protected are more in number.

Table 1. Useful parameters to understand Network Intrusion Detection

Parameter name	Description
Intrusion	Series of concatenated activities that pose threat to the safety of IT resources from unauthorised access to a computer or address domain
Incident	Violation of the system security, a successful intrusion
Attack	A failed effort to break system security, actual violation not happened

2.4 Network Based System

Monitors the network traffic of the network to which the hosts that are to be protected are connected. In this case the deployment cost is less and it's possible to identify attacks to and from multiple hosts. This type of IDS is passive so that it is easy to apply them to a pre-existing network without causing much disruption. Network based system can be implemented either as a early warning system or can be used in internal deployment mode [6]. Rafsanjani reported summary chart comparing Network-based NIDS and Host-based NIDS as depicted in Table 2.

Table 2. Comparative study of Network-Based and Host-Based network system

Network-Based IDS	Host-Based IDS
Broad in scope	Narrow in scope, monitors specific activities
Near real-time response	Responds after a suspicious entry
Host independent	Host dependent
Bandwidth dependent	Bandwidth independent
Slow down the network that has IDS client installed	Slow down the hosts that have IDS client installed
Detects network attacks	Detects local attacks
Not suitable for encrypted network	Suitable for encrypted network
Better for detecting attack from outside	Better for detecting attack from inside

2.5 NIDS as Early Warning System

NIDS is implemented outside the firewall and it scans all the data that is entering the network. In this case it is possible to detect attacks to and from multiple hosts. This system has a single point of deployment and hence the deployment cost is less and it is easy to update the signatures and configuring the system up to date. The disadvantage of this system is that it detects those malicious activities also that are blocked by firewall**.** Fig. 1[3] gives a schematic of the use of NIDS as an early warning system.

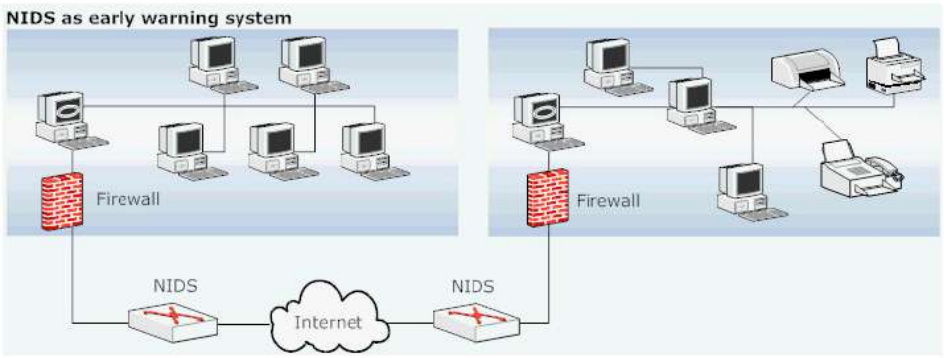

Fig. 1. NIDS as an early warning system

2.6 NIDS as Internal Deployments

In this approach the NIDS is deployed such that it monitors every network link through which the traffic is flowing and provides extra security. In this case the NIDS is placed near the access router near the network boundary. In this case the data that is blocked by the firewall is not scanned by the NIDS. But because of the large number of systems it is difficult to maintain and reconfigure the system with every rule set update. Fig. 2 gives a schematic of the use of NIDS in internal deployment mode. [3]

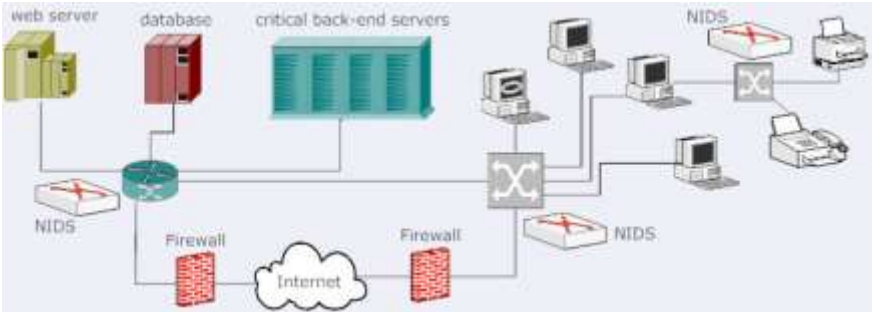

Fig. 2. NIDS in Internal Deployment Mode

III. SOFTWARE BASED NIDS APPROACH

Software based NIDS relies heavily on Snort Rules. Snort is a network intrusion prevention and detection system developed by Sourcefire. Snort is the most popular intrusion detection and prevention technology and has world -wide industry usage. It is a rule based technology that uses signature, protocol and anomaly detection methods. It has the capabilities of sniffer, packet logger and network traffic analysis. The basic rule set for Internet Traffic Analysis consists of 5567 rules.

Snort is a cross-platform, lightweight network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attacks. It can provide administrators with enough data to make informed decisions on the proper course of action in the face of suspicious activity [7].

A lightweight network intrusion detection system can be deployed almost on any node of the network. Lightweight IDS should be small, powerful and flexible so that they can be used as permanent elements of network security infrastructure. When deployed they should cause minimal disruption of the operations.

A signature based design method depends on distinctive marks or characteristics that are present in an exploit. This type of protection only has limited capabilities as the attack has already taken place before a signature can be written.

Snort can be configured to operate in three modes [8]: Sniffer mode (reads the packets off the network and displays them in a continuous stream on the console), Packet logger mode (logs packets to the disk), NIDS mode (performs detection and analysis on network traffic). Snort rules operate on network (IP) layer and transport (TCP/UDP) layer protocols.

The basic structure of Snort rule is as follows (refer Fig. 3):[9]

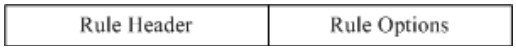

Fig. 3. Basic Structure of SNORT Rule

3.1 Rule Header

Consists of information for matching a rule against data packets and information about what action a rule takes.

3.2 Rule Options

Consists of alert message and information about which part of the packet should be used to generate the alert message.

Structure of the Snort rule header consists of the following parts (refer Fig. 4).

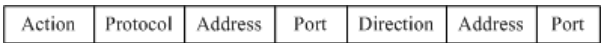

Fig. 4. Structure of SNORT Rule Header

3.3 Action

Action part of the rule determines the type of action taken when criteria are met and a rule is exactly matched against a data packet. E.g. alert or log message or invoking another rule.

3.4 Protocol

The protocol part is used to apply the rule on packets for a particular protocol. Snort recognizes the following protocols: IP, ICMP, TCP, UDP.

3.5 Address

Defines source and destination addresses.

3.6 Port

This part describes the source and destination ports of the packet for TCP or UDP protocols. For network layer protocols like IP and ICMP, port number has no significance.

3.7 Direction

This part specifies which port is the source port and which one is the destination port.

Rules can be grouped by port or protocol. Table 3 describes a few ports that are used popularly. Then only the group of rules that are applicable to the port/protocol to which the packet belongs is used. This helps to reduce the memory consumption and CPU usage of Snort. Snort rules are usually expressed in the form of (content+modifiers). Contents represent a fixed pattern that is to be searched in the packets payload of a flow. Modifiers are locations where content is searched inside the payload. .

3.8 Example of Snort Rule

Let's try to understand SNORT rule by using the example as provided in [9].This example explains a Snort rule that generates an alert message whenever it detects an ICMP1 ping packet (ICMP ECHO REQUEST) with TTL equal to 100[9].

```
alert icmp any any -> any any (msg: "Ping with TTL=100"; \ttl: 100;)
```

The first part of the rule is called rule header and the later part within parentheses is the options part. The header comprises of the below mentioned parts:

A rule action: Alert is generated when conditions are met. Packets are logged by default when an alert is generated. Depending on the action field, the rule options part may contain additional criteria for the rules.

Protocol: In this rule the protocol is ICMP, which implies that the rule will be applied only on ICMP-type packets. If the protocol of a packet is not ICMP the rest of the rule is not processed further by Snort Detection Engine and will save CPU time.

Source address and source port: In this example source address and source port are set as "any", which means that the rule will be applied on all packets coming from any source. Port numbers have no relevance to ICMP packets unlike to TCP or UDP.

*Direction: D*irection of the rule is indicated by the symbols ->, <- and <> specifying the directions such as left to right or reverse or both ways. In the example it is set from left to right using the -> symbol. This shows that the address and port number on the left hand side of the symbol are source and those on the right hand side are destination.

Destination address and port address:. In this example destination and port address are very generic and both set as "any". This indicates the rule will be applied to all packets irrespective of their destination address.

Table 3. Description of few popularly used ports

Port Number	Description
20	FTP Data
25	SMTP Used for e-mail servers
37	NTP, Used for synchronizing time on network hosts
53	DNS Server
80	HTTP, Used for all web servers
110	POP3, Used for e-mail clients like Microsoft Outlook
161	SNMP
443	HTTPS, or secure HTTP

The direction in this rule does not play any role because the rule is applied to all ICMP packets moving in either direction; due to the use of the keyword "any" in both source and destination address parts.

The options part enclosed in parentheses shows that an alert message will be generated containing the text string "Ping with TTL=100" whenever the condition of TTL=100 is met. TTL or Time To Live is a field in the IP packet header.

IV. Hardware Based NIDS Approach

A Software based NIDS such as widely employed software implementation of the SNORT rules are not capable of supporting very high rates of data (multi Gbits/s traffic rates typical of network backbones). For this reason these are normally applied in small scale networks. Hardware based NIDS can be a possible solution of this problem. But a main concern to be addressed while using hardware based NIDS is that the network intrusion threats and types of attacks are changing regularly. Hence the set of rules to counter them also needs to be updated continuously. Hardware system used for NIDS implementation should be dynamically reprogrammed (reconfiguration of the FPGA when the system is under operation) and updated in accordance with the changed rule set. Field Programmable Gate Array is thus a very attractive choice for NIDS implementation. FPGA support complex hardware architecture and can be dynamically reconfigured i.e. they can be modified when under operation. Reconfiguration of the FPGA requires a complete reprogramming of the chip.

FPGA devices consists of an array of interconnected programmable logic blocks or configurable logic blocks (CLB) surrounded by programmable I/O blocks. Special I/O pads with sequential logic circuitry are used for input and output of the FPGA. Fig.5 represents a schematic of FPGA.

FPGA architecture is of two types:[10] Fine grained Architecture: Consists of a large number of small logic blocks, e.g. transistors and Coarse grained architecture: Consists of larger and more powerful logic blocks, e.g. Flip-Flops and LUTs.

Some of the FPGA architectures provided by different vendors are as follows: Xilinx Virtex Architecture (coarse grained), Lattice ORCA Architecture(coarse grained), Lattice ispXPGA Architecture(coarse grained), Atmel AT40K Architecture(Fine grained), AItera APEmOK Architecture(coarse grained).

Present FPGA devices provide very high speed data collection ability but the device frequency is not enhanced proportionately. Use of FPGA helps in taking the advantage of huge parallelism.

4.1 Traffic Aware Design

The Snort rules can be analyzed and organized into disjoint subsets by suitable combinations of packet header files. Checking a protocol field can reject a large number of rules. The number of rejected rules varies significantly with the protocol field [11]. The rule set that used to counter the exploits against http servers (protocol=TCP, destination port = 80) differs from the ones employed for FTP or SMTP protocols. This subset of rules also differs from the ones used against exploits for web clients (protocol=TCP, source port = 80). Analysis of the traffic provided by the internet service providers can help to determine the expected worst case per-class throughput. Variations in the traffic mix occur during the operating lifetime of the NIDS. This can be of the order of several weeks. But we have to rerun the synthesis of rule content matching engine at every rule set update (order of once per week). Hence variation of the traffic mix can be accounted for while rerunning the synthesis for rule set update.

String Matching Algorithms based on Deterministic finite Automata (DFA) and Non-deterministic Finite Automata (NFA) has been proposed and mapped on an FPGA. These solutions pattern matching and matching of regular expressions. One more useful architecture for NIDS is the compare and shift architecture. Prefix sharing rules can save the FPGA area. The basic unit of a pattern-matching circuit designed using any of the approaches of DFA, NFA or compare and shift architecture is a character match unit. Approaches based on Hashing do not comply with the implementation of Snort, while the other approaches easily support it.

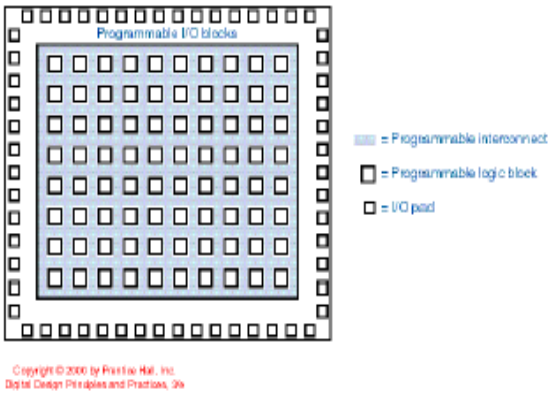

Fig. 5. FPGA Schematic Diagram

4.2 Compare and Shift Approach of Traffic Aware Design

The main input of the circuit is an 8 bit signal. This signal transports the payload under inspection one character each clock cycle [12]. The only output of the circuit is the "Match" signal. Match signal goes to high when a string is matched. The input is fed into an 8 bits register chain. The outputs of the register chain are provided as input to a combinatorial network that detects which are the characters are stored. The "Match" signal indicates that a rule has been matched without specifying which rule. This system can be deployed as a snort off loader that is devised to forward the malicious packets to a software IDS implementation driving simple pass/drop packet logic. The deployment of a full-fledged hardware IDS requires supplementary features (e.g. alert generation, packet logging and so on), that can be better performed in software.

Practical approaches implement a large number of content matching rules (of the order of thousands) and the strings to be matched can be hundred characters long. In such situation parallel search for different strings increases the fan in and fan out of the circuit components, also the length of the register chain increases linearly with length of the string to be matched. One solution of this problem is the use of a data bus.

The main architecture of the string matching system consists of the following components [12]:

4.2.1 Network Interface

Network interface is responsible for collecting packets from network link under monitoring.

4.2.2 Dispatcher

Dispatcher provides a classification of packet based on header.

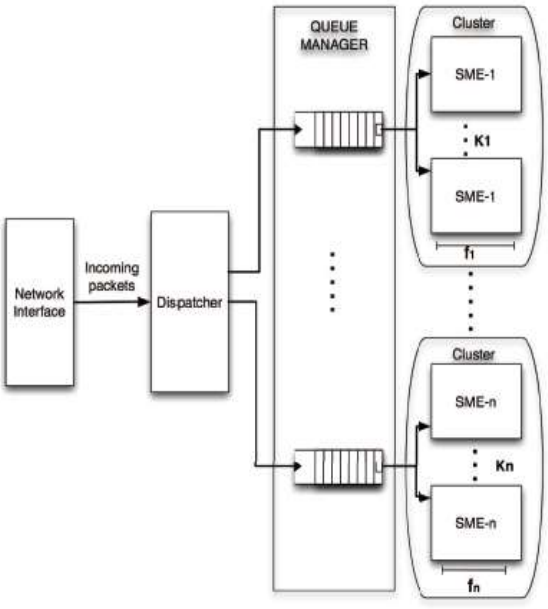

Fig. 6. General Implementation of overall String Matching System

4.2.3 String Matching Engines

String Matching Engines perform the string matching operation. The designs of different clusters used in the implementation are identical. But the content searching rules synthesized in string matching engines belonging to different clusters differ and specifically depend on the type of traffic routed to the considered cluster.

4.2.4 Queue Manager

This block provides a queue for each SME cluster. This is used to maintain sudden burst of packets.

The general implementation of overall string matching system is depicted in Fig.6.

The implementation of the above concept requires attention to the following parameters:

Dispatcher classification policy, String matching rules loaded over each cluster of engines, operating frequency of each cluster, number of string matching engines deployed in each cluster, per-engine optimized hardware design, and traffic-load based system dimensioning.

4.3 Use of Deterministic Finite Automata for Implementation of Content Scanning Module

Intrusion detection system can provide protection to the Local Area Network (LAN) by implementing Access Control Policies (ACP) for both incoming and outgoing traffic. With regular expressions the efficiency of ACP can considerably be improved. Additionally the use of regular expressions in ACPs give them the ability to enforce rules on mutable contents that are found in many Denial of Service(DOS) Attack and services.[13]. DFA has one active state. This provides the advantage of compact state encoding which in turn supports efficient context switches useful for certain applications. The disadvantage of single active state is that it might need complex state transition logic or a state machine with a large number of states.[14]

A regular expression has individual characters as the basic building blocks, eg. "a", "b", "c". They individually can be considered as simple regular expressions. Characters can be combined with meta characters (*, |,?) to form more complex regular expressions. Table. 4 describe a few examples [12]. Fig.7 represents a syntax tree for regular expressions.[15]

The design of the content scanning engine consists of three parts:

i) Receiving packets
ii) Processing packets
iii) Outputting packets.

Each of these operations are controlled independent of the other two and can run in parallel.

Data enters the receiver in 32 bit chunks. Three control signals are used to indicate the start of packet, end of packet and a valid signal to indicate the presence of a valid 32 bit data in the bus. Every valid data word along with the three control signals are written into input memory buffers.

Table 4. Description of Regular Expression

Regular expression	Comment
A	Singleton set {“a”}
a*	Matches any string composed of zero or more occurrences of ”a”, Denotes the infinite set {“”, “a”, “aa”, “aaa”,…}
a?	Matches any string composed of zeo or one occurrence of “a”
a|b	Matches any string composed of a or b, Denotes the set {“a”, “b”}
Ab	Matches any string composed of a concatenated with b
E	Regular expression that matches the empty string

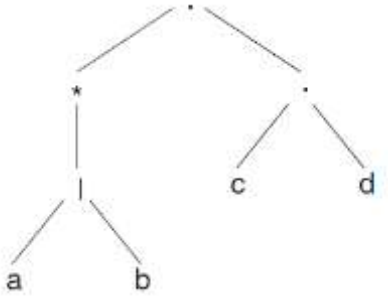

Fig. 7. Syntax Tree for ((a\b)*)(cd)

On each clock tick, one character (8- bits) is read from the memory bus and sent to each of the regular expression DFAs. One counter is used to address the memory devices. All of the DFAs search in parallel. Each DFA maintains a 1-bit match signal which is asserted high when a match is found within the packet that is being processed. When the counter reaches the end of the packet, if the match signals from all of the DFAs indicate no match was found, or if any of the match signals indicate a match was found but do not require dropping the packet, then a pointer to the packet is inserted into a queue for output. Otherwise a pointer to the packet is not inserted into a queue for output, hence the packet is dropped. To generate an alert signal a special pointer is used. A packet is output from the content scanner whenever there is an available pointer in the output queue. Each pointer can be either for a regular packet or for an alert packet. Fig.8 represents a content scanner block diagram.

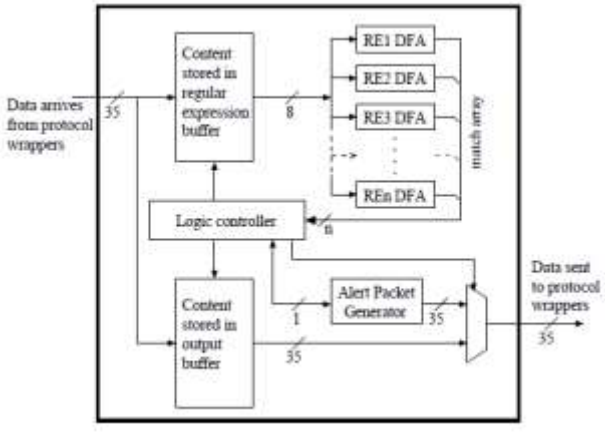

Fig. 8. Content Scanner Block Diagram

4.4 Use of Non Deterministic Finite Automata for Implementation of Pattern Matching

The non-deterministic finite automata (NFA) approach has lesser transition logic complexity as compared to deterministic finite automata. NFA has multiple active states and can support regular expressions. Informally an NFA is a directed graph that has each node as a state and edges labelled with a single chracter or ε. One state is the initial state and some states are accepting or final states. DFA has no edge labelled ε and no state has more than one edge labelled with the same character.[15]

Pattern matching can be done using an 8-bit comparison of input and the pattern character. Fig. 6 gives a diagram of distributed comparators.[14] Instead of using a distributed comparator a character decoder can be used. In this case all the processing are performed in a single central location and only the necessary matched information is passed to the required unit. For 8-bit characters, this can be achieved by using a shared 8-to-256 decoder and connecting the appropriate one-bit output of the decoder to each unit. Compared to the distributed comparator design, the character decoder approach doubles the maximum pattern capacity of a given reconfigurable logic device. Using this method it is possible to fit one character matching unit into one logic element whereas tho logic elements are required for the distributed system. One logic unit is defined as one 4-input LUT and a flip-flop. Fig. 9 gives the diagrams of distributed comparators and the character decoders.

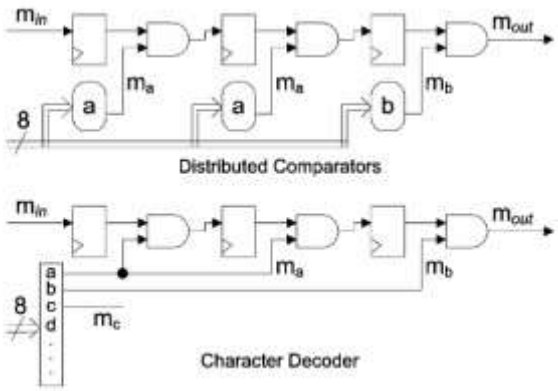

Fig. 9. Distributed Comparator and Character Decoder circuits

Character decoder technique can be used to build circuits that can process more than one character at a time.

This helps to increase the throughput without increasing frequency. A pattern matching circuit uses N character decoders simultaneously decoding a different input character to process N characters per clock cycle. All patterns should be searched at N possible offsets by implementing N parallel state machines to track matches at all offsets. Fig. 10 represents block diagram of N-character decoder NFA module. A wire label of the form ci represents the match signal output of i-th input character decoder for the character code c.

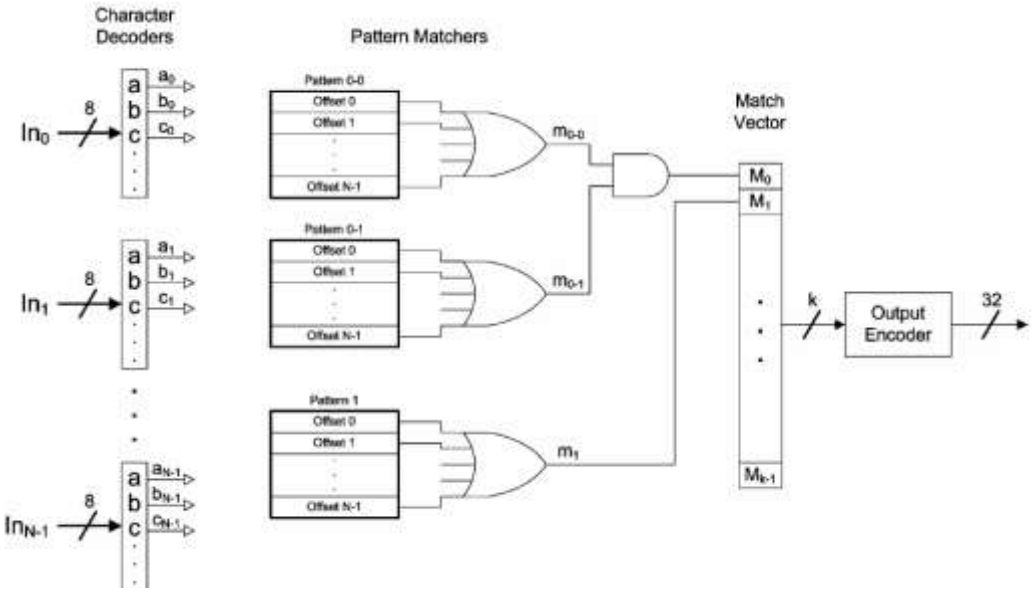

Fig. 10. Block Diagram of N-character Decoder NFA module

V. Conclusion

The demand for a secure network is ever increasing. One central challenge with computer and network security is the determination of the difference between normal and potentially harmful activity. The core component of popular IDSs, like Snort [2], is a deep packet inspection engine that checks incoming packets against a database of known signatures (also called rules).The dominant factor in determining the performance of this signature matching engine, both in software or hardware implementation is the number and complexity of the signatures that must be tested against incoming packets. Exploitation of traffic classification and load statistics may bring significant savings in the design of Hardware Network Intrusion Detection Systems (NIDS). The ultimate design goal for an intrusion detection system is the development of automated and adaptive design tool for network security.

References

[1] Zachary K. Baker, Student Member, IEEE, and Viktor K. Prasanna, Fellow, IEEE. Automatic Synthesis of Efficient Intrusion Detection Systems on FPGAs. IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 4, October-December 2006.

[2] Przemyslaw Kazienko & Piotr Dorosz. Intrusion Detection Systems (IDS) Part I - (network intrusions; attack symptoms; IDS tasks; and IDS architecture). www.windowsecurity.com › Articles & Tutorials

[3] Sailesh Kumar, "Survey of Current Network Intrusion Detection Techniques", available at http://www.cse.wustl.edu/~jain/cse571-07/ftp/ids.pdf.

[4] Srilatha Chebrolu, Ajith Abrahama,,*, Johnson P. Thomas, Feature deduction and ensemble design of intrusion detection systems, Elsevier Ltd. doi:10.1016/j.cose.2004.09.008

[5] Uwe Aickelin, Julie Greensmith, Jamie Twycross . Immune System Approaches to Intrusion Detection - A Review.http://eprints.nottingham.ac.uk/619/1/04icaris_ids_review.pdf

[6] http://www.intechopen.com/download/get/type/pdfs/id/8695

[7] Martin Roesch , "Snort – Lightweight Intrusion Detection for Networks", © 1999 by The USENIX Association

[8] The Snort Project, Snort User Manual 2.9.5,May 29, 2013, Copyright 1998-2003Martin Roesch, Copyright 2001-2003 Chris Green, Copyright 2003-2013 Sourcefire, Inc.

[9] Chapter 3, Working With Snort Rules, Pearson Education Inc.

[10] Sumanth Donthi Roger L. Haggard . A Survey of Dynamically Reconfigurable FPGA Devices. 0-7803-7697-8/03/2003 IEEE.

[11] S. Sinha, F. Jahanian, J. Patel, "Wind: Workload-aware intrusion detection",Recent Advances in Intrusion Detection, Springer, pp. 290–310,2006.

[12] Salvatore Pontarelli, Giuseppe Bianchi, Simone Teofili. Traffic-aware Design of a High Speed FPGA Network Intrusion Detection System. Digital Object Indentifier 10.1109/TC.2012.105, IEEE TRANSACTIONS ON COMPUTERS

[13] J. Moscola, J. Lockwood, R.P. Loui, and M. Pachos, "Implementation of a Content-Scanning Module for an Internet Firewall," Proc. of 11th IEEE Symp. on Field-Programmable Custom Computing Machines, FCCM 2003, pp. 31-38.

[14] C. R. Clark and D. E. Schimmel, "Scalable parallel pattern-matching on high-speed networks," in Proc. of IEEE Symposium on Field- Programmable Custom Computing Machines, FCCM 2004, pp. 249-257.
[15] R. Sidhu and V.K. Prasanna, "Fast Regular Expression Matching Using FPGAs," in Proc. of the 9th IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2001, pp. 227 - 238.

Authors' Profiles

Sutapa Sarkar: Sutapa Sarkar is pursuing her Master in Technology in Digital Electronics and Communication Engineering from M.V.J. College of Engineering, Bangalore, India. She completed Bachelor of Technology in Optics & Opto-Electronics in 2006 from University of Calcutta, India. Before starting her post-graduation, she worked with HCL Technologies Ltd for three and half years in field of software testing in avionics domain(2007-2010). Her research interests cover wireless sensor networks, safe and secure communication, FPGA implementation.
Email:sutapasarkar11@rediffmail.com.

M. Brindha: completed Bachelor in Engineering in Electronics and Communication in 2004, Master of Engineering in Electrical Drives and Embedded Control from Anna University, Chennai (2007). She is currently an Associate Professor in the Department of ECE, M.V.J. College of Engineering, Bangalore, India. She has published many journals and attended many Conferences in National and International Level. Her research areas are Embedded Systems, Control system, Networking, FPGA Implementation and Algorithms.
Email: Brindha_mo47@yahoo.com.

How to cite this paper: Sutapa Sarkar, Brindha.M,"High Performance Network Security Using NIDS Approach", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.7, pp.47-55, 2014. DOI: 10.5815/ijitcs.2014.07.07

