HIGH-PERFORMANCE NETWORK SECURITY USING NETWORK INTRUSION DETECTION SYSTEM APPROACH

M.Sathiya
Assistant Professor, Department of Computer Science & Applications
Vivekanandha College of Arts and Sciences for Women (Autonomous), Namakkal, TamilNadu, India

Abstract
Ever increasing demand for good quality communication relies heavily on Network Intrusion Detection System (NIDS). Intrusion detection for network security demands high performance. This paper describes of the available approaches for a network intrusion detection system in both software and hardware implementation. This paper describes 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.
Keywords: Network Intrusion Detection System, Snort, FPGA

Introduction

Network Intrusion detection system is described as the process of identifying and taking necessary actions against malicious activities targeted to network and computing resources. 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.

The network intrusion detection system can be placed at a choke point such as the company's connection to a trunk line, or it should be placed on each of the hosts that are being monitored to protect from intrusion. Intrusion, incident and attack are three terms that we frequently come across while discussing Intrusion Detection System.

A NIDS should have the following desirable features:

- System should be fault tolerant and run the minimal human supervision.
- The NIDS should not be susceptible to attacks from the intruder
- NIDS should not hinder the normal operation of the system.
- NIDS should be portable to different architectures making it is easy to deploy.
- NIDS should be general to detect different types of attacks and should have as less number of false positives as possible.

Shortcomings of Traditional Security

a) Firewall Evasion

b) Tunneling attacks: Tunneling refers to the course of gaining unauthorized access to the network by encapsulating specific messages, which are blocked by the firewall, within

messages of another type. The firewall implements rules of filtering of packets based on the network protocol that is allowed.

c) Attacks due to incorrect firewall configuration: It is a fact that most firewalls are configured and deployed by humans. And human beings are prone to error making. This information is well-known to the intruders who try to take advantage of it. They try to find a security break in the configuration of the firewall and develop it.

d) Attacks by trusted hosts and networks: Like most organizations deploying security mechanisms use encryption for protecting files and external network connections, so the intruder's attention will stretch out on such locations where the encryption/protection of data broadcast is missing or very minimal. This is the case where the data is stored and transmitted to trusted hosts and networks.

e) Attacks by source address spoofing: Address spoofing is a method which is used to hide the real address of the sender of a network packet, particularly the intruder. However, this can also be used to bypass the firewall and gain unauthorized access to a network or computer. Contemporary firewalls have in-built mechanisms to avoid this fraud.

f) Attacking the firewall itself: As the firewall software and hardware are built by humans, they are prone to attack from the intruders. A successful attack on the firewall can lead to very serious consequences as once successfully attacked, intruders can freely access the resources of the protected network without the risk of being detected and traced.

g) Attacks on the firewall authentication system: In this case, the authentication system of the firewall itself being attacked by the intruder. Once the authentication system is attacked, the firewall will not even register a security violation because it will interpret the intruder as an authorized user. Example- CISCO PIX Firewall Vulnerability.

Types of Network Intrusion Detection System

In software-based NIDS approach the IDS are software systems that are particularly designed with the aim of identifying and hence help to stop 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.

- **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 are being unable to identify novel attacks. Snort is a well-defined rule set that uses signature, protocol and anomaly-based detection methods.
- **Anomaly Detection**
  This approach is being identified novel attacks that are yet unknown and hence undetectable by signature-based NIDS. The main disadvantage of anomaly detection method is that it may be 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 cultured 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.

- **Host Based System**

  This type of IDS is present on each host that needs monitoring. These can 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

- **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 inert so that it is easy to apply them to a pre-existing network without causing much disturbance.

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

- **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 to configure the system up to date. The disadvantage of this system is that it detects those malicious activities also that are blocked by firewall.

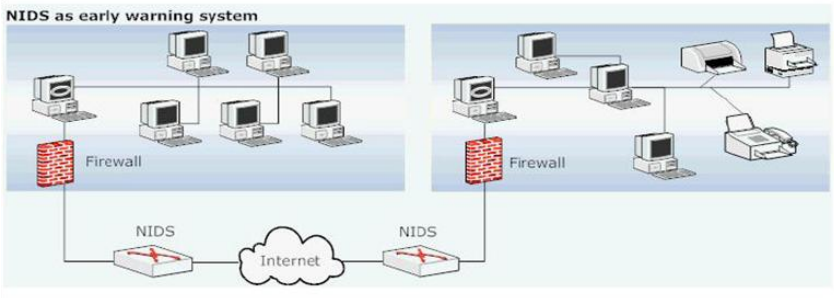


Fig 1 NIDS as an early warning system

- **NIDS as Internal Deployments**

  In this approach the NIDS is deploy 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.

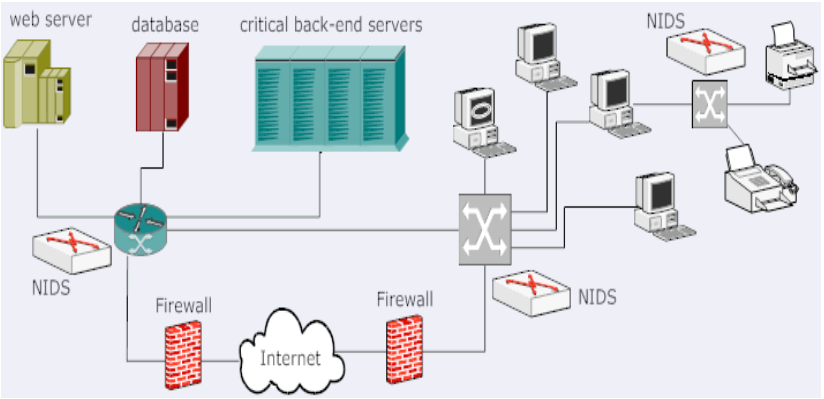


Fig 2 NIDS in Internal Deployment Mode

Software-based NIDS Approach

Snort is a free and open source network intrusion prevention system and network intrusion detection system capable of performing packet logging and real-time traffic analysis on IP networks. Snort was written by Martin Roesch and is now developed by Source fire. Snort performs all the basic functions of a network intrusion detection system which we discussed before. Mainly, Snort performs protocol analysis, content searching/matching and is commonly used to actively block or passively detect a variety of blocks and attacks, some of which are buffer overflows, port scans, web application attacks and operating system fingerprinting attacks.

Snort can also be combined with other free software to give a visual representation of visual data. It is a cross-platform, lightweight intrusion detection system which can deploy on a variety of platforms to monitor TCP/IP networks and detect suspicious activities.

Snort was designed to fulfill the requirements of a prototypical lightweight network intrusion detection system. It has become a small, flexible, and highly capable system that is in use around the world on both large and small networks. It has attained its initial design goals and is a fully capable alternative to commercial intrusion detection systems in places where it is cost inefficient to install full-featured commercial systems. Software-based NIDS relies heavily on Snort Rules. The basic rule set for Internet Traffic Analysis consist 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.

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 deploy they should cause minimal disruption of the operations.

Snort can be configured to operate in three modes: Sniffer mode (reads the packets of the network and displays them in a continuous stream on the console), Packet logger mode (logs packets to the disk), and 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):

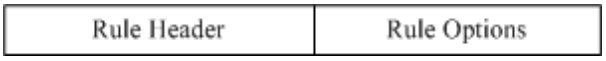


Fig 3 Basic Structure of SNORT Rule

- **Rule Header**
  It consists of information for matching a rule against data packets and information about what action a rule takes.
- **Rule Options**
  It 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).

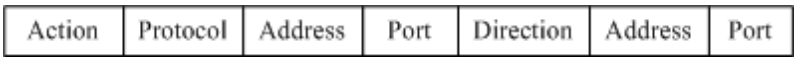


Fig 4 Structure of SNORT Rule Header

Hardware-based NIDS Approach

A Software-based NIDS such as widely employed software implementation of the SNORT rules are not capable of sustaining 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 the 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 constantly. Hardware system used for NIDS implementation should be dynamically reprogrammed (reconfiguration of the FPGA when the system is under operation) and updated by 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 customized 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 influential logic blocks, e.g. Flip-Flops and LUTs.

- **Traffic Aware Design**

The Snort rules can be analyzed and gained 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 be provide 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).

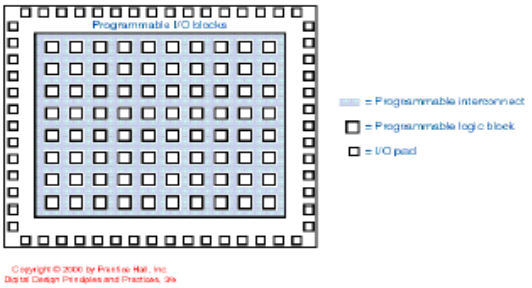


Fig. 5. FPGA Schematic Diagram

- **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. 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 deploying as a SNORT of 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.

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

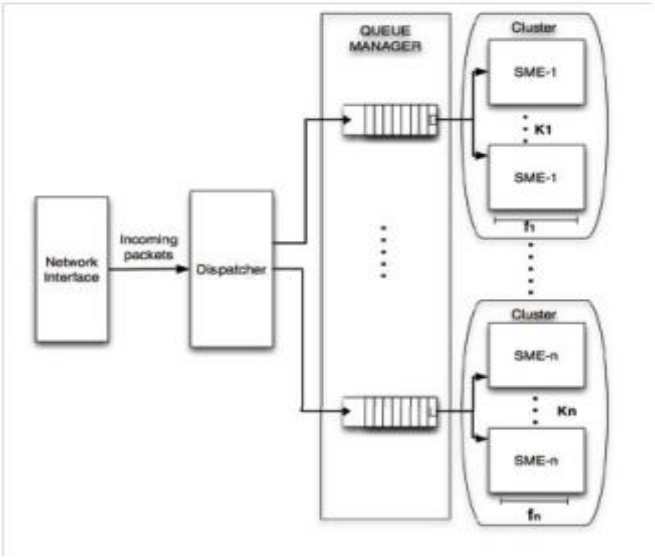
- *Network Interface*

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

- *Dispatcher*

Dispatcher provides a classification of packet based on header.

Fig.6 Implementation of the overall string matching system



- *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.

- *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 deploys in each cluster, per-engine optimized hardware design and traffic-load based system dimensioning.

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 to 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.

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

a. Receiving packets
b. Processing packets
c. Outputting packets.

Each of these operations is controlled independent to 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

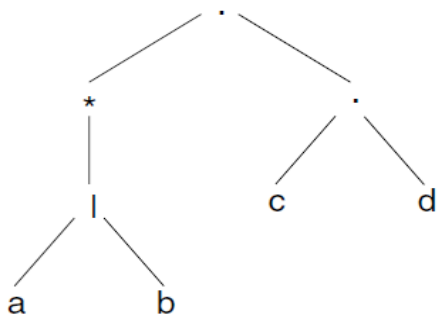


Fig 7 Syntax Tree for ((a\b)*)(cd)

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.

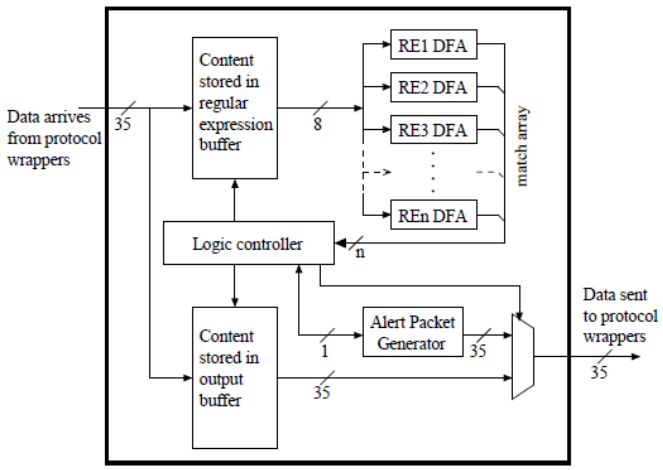


Fig 8 Content Scanner Block Diagram

Use of Non Deterministic Finite Automata for Implementation of Pattern Matching

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.

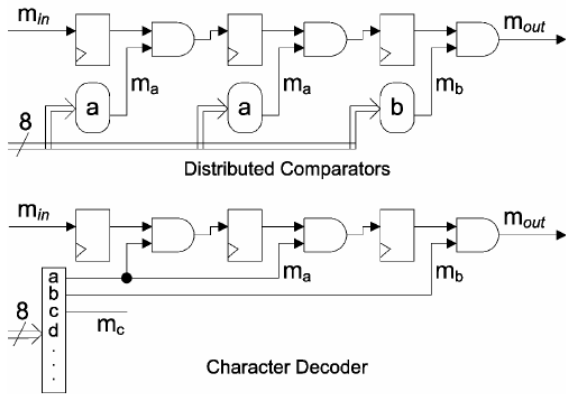


Fig 9 Distributed Comparator and Character Decoder circuits

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.

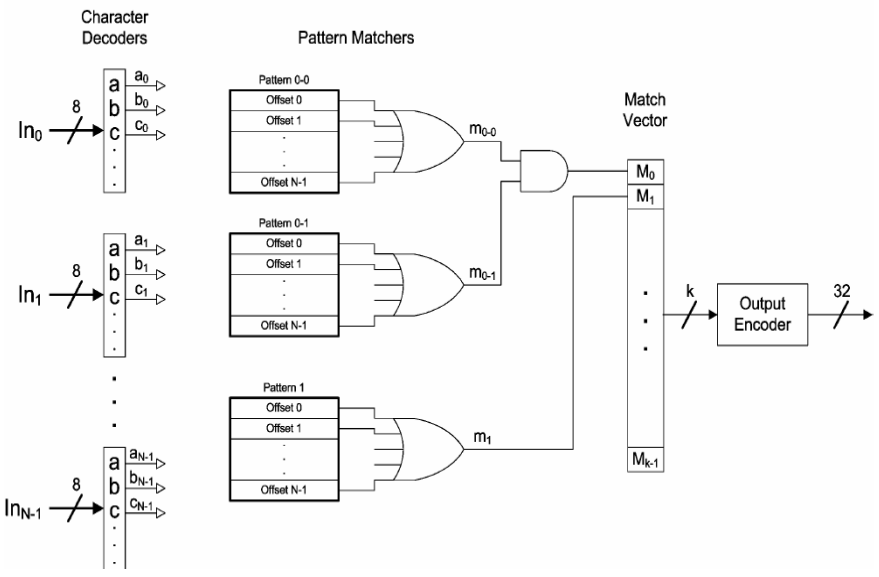


Fig. 10. Block Diagram of N-character Decoder NFA module

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 to 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. PrzemyslawKazienko&PiotrDorosz. 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. SrilathaChebrolu, AjithAbrahama,,*, Johnson P. Thomas, Feature deduction and ensemble design of intrusion detection systems, Elsevier Ltd. doi:10.1016/j.cose.2004.09.008
5. UweAickelin, 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. SumanthDonthi 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