MULTITHREADED SCALABLE MATCHING ALGORITHM FOR INTRUSION DETECTION SYSTEMS

By

ADNAN AHMAD ABDELFATTAH HNAIF

Thesis submitted in fulfillment of the requirements

for the degree of

Doctor of philosophy

May 2010

ACKNOWLEDGMENTS

"All praises and thanks to ALLAH"

A major research like this is never done without the contributions of many different people, in their different ways. Therefore, I would like to extend my appreciation especially to the following.

First of all, the praises and thanks go to almighty ALLAH for giving me the patience, the health as well as giving me the chance to work in such an environment in Malaysia and in USM in particular. Second, I wish to deeply thank my supervisor Prof. Dr. Sureswaran Ramadass for introducing me to the world of research and giving me the golden chance to work in this field. I am also grateful to my co-supervisor Dr. Omar Amer Abouabdalla, for his help and support throughout my period of study.

Special thanks to National Advanced IPv6 Center of Excellence (NAv6) and the USM Fellowship Scheme respectively, for providing a conductive environment and support me during the course of my research.

Last but not the least, I would like to thank those who are close to my heart; my father Mr. Ahmed Hnaif, for his endless, support, and continuous encouragement, to my beloved mother, to my wonderful wife Ahlam and our children Aya, Abdullah, and Izzuddin for their understanding, patience and supporting me throughout the period of research, to my dearest brothers {Khaled, Nader, Amer, Omar, Emad, Ala'a and Hatem} Hnaif for keeping me smiling and motivated, to my father, mother

and brothers in law, and to all my colleagues. Finally I would like to express my thanks to my friend Dr. Mohamed Faizal and his wife Khadija Beevi. I dedicated this work to all of them as without whose support and understanding, this thesis would not have been completed.

Thank you!

Adnan Ahmad Hnaif

Penang, Malaysia. May 2010.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

AC_BM	Aho-Corassick, Boyer-Moore algorithm
BM	Boyer-Moore Algorithm
BMBC	Boyer-Moore Bad Character table
DPHM	Distribute Packet Header Matching
EF	Early Filtering Algorithm / Energy Function
ExB	Exclusion-based String Matching Algorithm
GA	Genetic Algorithm
HIDS	Host Intrusion Detection System
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
LAN	Local Area Network
MPI	Message Passing Interface
NIDS	Network Intrusion Detection System
NNIDS	New Network Intrusion Detection System
OTN	Option Tree Nodes
PF	Packet Filter Algorithm
PME	Pattern Matching Engine
PPS	Packet per second
RHPNIDS	Rule-based High Performance NIDS
RTN	Rule Tree Nodes
SSPP-BM	SHIFT, SUFFIX, PREFIX, and PATTERN_LIST–Boyer Moore
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
VoIP	Voice Over Internet Protocol

LIST OF APPENDICES

ALGORITMA PEMADANAN BERBILANG BENANG YANG BERSKALA UNTUK SISTEM PENGESANAN PENCEROBOHAN RANGKAIAN

ABSTRAK

Peningkatan kelajuan dalam rangkaian komputer memberi kesan secara lansung terhadap prestasi Sistem Pengesanan Penerobosan Rangkaian (Network Intrusion Detection System, NIDS) dari segi kelajuan dalam mengesan ancaman. Oleh itu, prestasi algoritma yang sedia ada perlu diperbaiki dari segi turutan dan keselarian untuk mempertingkatkan kelajuan enjin pengesanan yang digunakan dalam SNORT-NIDS. Tesis ini menghuraikan satu algoritma baru yang dinamakan Padanan Kepala Paket Teragih (Distributed Packet Header Matching, DPHM) dan platform Sistem Baru Pengesanan Penerobosan Rangkaian (New NIDS, NNIDS) dengan menggunakan teknologi hibrid dalam meningkatkan prestasi keseluruhan SNORT-NIDS.

Algoritma DPHM menukarkan set aturan kepala ke dalam bentuk berat dan menyimpannya dalam satu jadual carian (lookup table). Ia kemudiannya dipadankan di antara kepala paket masuk (incoming packet header) dan set aturan kepala (headers rule set). Proses padanan kelajuan SNORT-NIDS dipertingkatkan dengan menggunakan cadangan proses pembelajaran yang terdapat dalam algoritma DPHM.

Platform NNIDS akan mengagihkan paket masuk beban bayar (incoming packets payload) dalam dua senario. Dalam senario yang pertama, paket masuk beban bayar (incoming packets payload) akan diagihkan kepada pemproses yang sedia ada dalam

seni bina memori yang di kongsi dengan menggunakan simpanan Antara Muka Penghantaran Mesej (Message Passing Interface, MPI). Bagi senario kedua, paket masuk beban bayar (incoming packets payload) akan diagihkan di kalangan pemproses yang mempunyai pelbagai teras memproses dengan menggunakan simpanan hibrid MPI dan simpanan OpenMP dalam seni bina memori yang dikongsi.

Prestasi algoritma DPHM telah diperbaiki sebanyak 25 peratus berbanding dengan SNORT-NIDS (Algoritma DPHM memerlukan 2.33 saat dan SNORT-NIDS memerlukan 3.22 saat) untuk memproses 3000 kepala paket dengan 0.5MB set aturan kepala di mana bilangan pemproses bersamaan dengan dua. Di samping itu, NNIDS juga telah memperbaiki prestasi SNORT-NIDS yang sedia ada sebanyak 80 peratus (NNIDS memerlukan 0.28 saat dan SNORT-NIDS memerlukan 1.71 saat) untuk memproses 3000 paket dengan 1MB beban bayar set aturan di mana bilangan pemproses bersamaan dengan dua yang setiap satunya mempunyai empat teras. Keseluruhan prestasi SNORT-NIDS dari segi kelajuan telah meningkat sebanyak 50 peratus (dari 3.90 saat kepada 1.71 saat) bergantung kepada beban paket, saiz set aturan dan bilangan pemproses yang digunakan.

MULTITHREADED SCALABLE MATCHING ALGORITHM FOR INTRUSION DETECTION SYSTEMS

ABSTRACT

The increasing speed of today's computer networks directly affects the performance of Network Intrusion Detection Systems (NIDS) in terms of speed of detection of threats. Therefore, the performance of the existing algorithms needs to be improved both in sequential and parallel to enhance the speed of the detection engine used in SNORT-NIDS. Hence, this thesis defines a new algorithm called the Distributed Packet Header Matching algorithm (DPHM), and a New Network Intrusion Detection Systems (NNIDS) platform using hybrid technology in order to increase the overall performance of SNORT-NIDS.

The DPHM algorithm converts the header rule sets into weights and stores them in a lookup table. It then matches the incoming packets header with the headers rule sets. The speed of the SNORT-NIDS matching process is enhanced using the proposed learning process which is contained within the DPHM algorithm.

Furthermore, the NNIDS platform will distribute the incoming packets payload into two scenarios: In the first scenario, the incoming packets payload will distribute among available processor in shared memory architecture using Message Passing Interface (MPI) library. In the second scenario, the incoming packets payloads will be distributed amongst available processors with multiple-cores processors using a hybrid of MPI library and OpenMP library in shared memory architecture.

The performance of the DPHM algorithm has been improved about 25% comparing with SNORT-NIDS (DPHM algorithms need 2.33 seconds and SNORT-NIDS needs 3.22 seconds) to process 3000 packets header with 0.5MB headers rule sets, when the number of processors are equal 2. Whereas, the NNIDS improved the performance of the current SNORT-NIDS about 80% (NNIDS needs 0.28 seconds and SNORT-NIDS needs 1.71 seconds) to process 3000 packets with 1MB payload rule sets when the number of processors are equal 2 with 4 cores each. The overall performance in terms of speed of the SNORT-NIDS has been improved about 50% (from 3.90 seconds to 1.71 seconds) depending on the packets load; rule sets size and the number of processor used.

CHAPTER ONE

INTRODUCTION

With the rapid evolution of the Internet and its applications, the Network Intrusion Detection Systems (NIDS) are becoming inefficient because of the amount of the traffic that needs to be processed daily. Moreover, current SNORT-NIDS implementations are inadequate to process all the traffic in real time. Therefore, the main objective of this thesis is to enhance the speed of engine detection in real time for packets header and packets payload in SNORT-NIDS. For the packets header, we proposed a new algorithm called Distributed Packet Header Matching algorithm (DPHM). This algorithm can be run on a single processor or multiple-cores platform. For the packets payload, this thesis also proposed a new platform called New Network Intrusion Detection Systems (NNIDS). This platform can utilize any exact string matching algorithms.

1.1 BACKGROUND

Network security is responsible for protecting the information passing through any network from the intruders. Moreover, network security refers to all hardware and software functionalities such as: identifying network characteristics and features, operational procedures, measures of accountability, access controls administrative and regulatory policies that are necessary to provide an acceptable level of protection to the network (Alan R. Simon, 1994; Stallings, 2006; and Thomas, 2002).

For example, a firewall is a system that is used to secure the internal network from the external traffic (Muhammad Abedin, 2006). A firewall swaps the information between the Internet and the intranet and provides the first level of defense for the networks. This will also stop unauthorized people from accessing the network.

However, the traditional firewalls are insufficient to ensure network security because (Li W., 2004; Dressler, 2004/2005).

- A firewall usually cannot detect any threats from the internal network, such as the trojans and botnets. A firewall is meant to protect the network boundaries.
- A firewall filters all unwanted network traffic, but allows some of the services (i.e. VoIP traffic data) to pass (Chen, 2008). Intruders can use this limitation to break into the network.

Firewalls can be grouped into three main categories (Innella, 2000):

- Packet-filtering router firewall: this determines which traffic is allowed to pass through the router to the local area network (LAN).
- Application-level gateway firewall: it is used to identify and validate the network applications access privileges based on the application usage level.
- Circuit-level gateway firewall: this is used to identify and authenticate the users' network access.

Because of the weaknesses of the firewalls, the intruders can still find different methods to penetrate the network every time. Intrusion Detection Systems (IDS) can be used to detect these intrusions that could affect the network. This makes the IDS

being regarded as a complimentary solution within most organizations (M. M. Pillai, 2004; Giovanni Vigan, 2004; and Konstantions Xinidis, 2006) because IDS will monitor the internal activities against the intrusions and warn the network adminestrator about them.

1.2 Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) is a system that works after the firewall to prevent unauthorized people from accessing the network. In addition, Intrusion Detection Systems is a system to detect intrusions that are trying to steal information and reporting these intrusions' existence to the network administrator (Li W. , 2004).

IDS work a mechanism for protecting confidentiality, availability, and integrity to avoid bypassing the boundary security mechanisms. Other researchers like (S. Antonatos K. G., 2004) defined IDS as a process of determining whether the attack was an attack attempts or the attack is taking place. Where (Eduardo Mosqueira-Rey, 2007) and (Alan R. Simon, 1994) defined IDS as the detection of the activities that are contrary to the normal behavior of the network.

The real time IDS overview is depicted in Figure 1.1 that consists of four engines:

1. Capture engine: this engine is used to capture all the incoming packets from the network.
2. Preprocessor engine: used to prepare the captured packets for the detection engine.

3. Detection engine: used to check all the preprocessed packets against any possible intrusions.
4. Alert, log, and pass engine[1]: used to generate a suitable level of alert to the network administrator

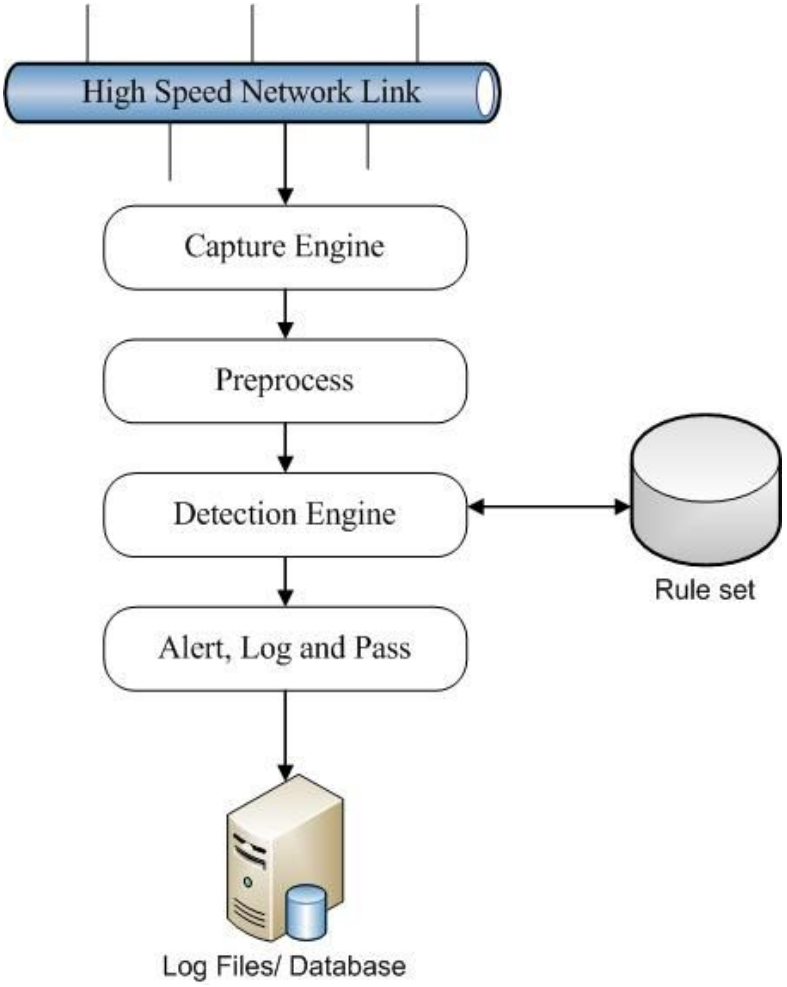

Figure 1.1: Intrusion Detection Systems Architecture.

IDS can be classified based on their techniques into two categories: (Zhou Chunyue, 2006; Varghese, 2004; and Mahadeo, 2008):

- Misuse Detection IDS.
- Anomaly Detection IDS.

[1] Three cases discussed in chapter two, section 2.3.1

Misuse Detection IDS is based on known patterns/signatures. The IDS is using this technique to identify the existence of a pattern in the monitored network traffic. Anomaly Detection IDS is based on network behaviors. The IDS used it to differentiate between normal and abnormal network behavior (Dr. Fengmin Gong, 2002).

Furthermore, IDS can be divided into three groups based on the intrusion behavior (Pels, 2005; Sinn, 2007):

1. Network Intrusion Detection Systems (NIDS) (Network-based IDS).
2. Host Intrusion Detection Systems (HIDS) (Host-based IDS).
3. Stack-Based IDS: works closely with the TCP/IP stack to allow network traffic to be examined/checked against any known intrusion.

The NIDS is used to identify intrusions by monitoring the network traffic. The HIDS is used to monitor the hosts of the local files and process the activities. Moreover, the HIDS listens or links to the network traffic to identify known attacks against the host (Rehman, 2003; Laing, 2000; Bezroukov, 2003; Ericsson, 2006; Michael Gregg, 2007.

NIDS offers the following additional functionalities that HIDS does not provide.

- Packet Analysis: NIDS examines all incoming packets header to detect any malicious activity. In addition, NIDS can also check the packet payload for special commands or syntax.

- Real-Time detection and response: In order to save the information from being damaged or stolen, NIDS can detect network attacks and then reports them to the network administrator in real time.

On the other hand, HIDS has an advantage over NIDS in determining if the attack has been successful. This is due to the fact that NIDS suffers from high false positives alerts. However, it is argued that the IDS must include both NIDS and HIDS for more efficiency and detection accuracy (Benjamin Morin, 2007).

1.3 Problem Statement

NIDS works by matching the string patterns or signatures against packets header and packets payload using exact string matching algorithm such as (ESMAJ, 1997).

1- Boyer-Moore Algorithm (BM). BM is widely used because of its efficiency in using a single pattern matching to problems. BM uses bad character shift and good suffix shift to find the pattern in the text.

2- AC_BM (Aho-Corassick, Boyer-Moore) Algorithm. This algorithm examines the text from right to left and uses a common prefix approach instead of a common suffix approach. The keyword tree moves from the right end of the packet payload to the left end. Meanwhile, the character comparisons are performed from the left to the right.

3- Horspool Algorithm. This algorithm works in any order and the average number of comparisons for one text character is between $1/\sigma$ and $2/(\sigma+1)$. Horspool algorithm is faster than BM algorithm and AC_BM algorithm for the medium size patterns.

These exact string matching algorithms work in the linear mode and all of them were depended upon to create the Boyer-Moore Bad Character (BMBC) table with some modifications (Rafiq, 2004). Some algorithms are a little bit faster than the others, especially in a short text (Rong-Tai Liu, 2004).

In addition, Intel's Communication Technology laboratory parallelized a SNORT-NIDS Intrusion Detection Systems on four execution cores (Verplanke, 2007). They used the POSIX threading library (Pthreads for win32, 2006) to implement their technique. Each thread will be executing the same loop, reading incoming packets and processing the packets independently (Verplanke, 2007).

The (Bart Haagdorens1, 2004) presented five designs for the multi-threaded NIDS sensors. The main idea from their work is to save incoming packets into a queue. After the packet goes to the preprocessor phase, all the threads regardless of their number will be processing one single packet at a time until the queue is empty. However, this technique has two problems:

1. In the case of one packet in the queue, all the threads will rush to process this single packet. In this case, one thread could be faster than multiple threads because of the synchronization between the threads.
2. Each thread will process its own data in one single packet. The borderline (the area between one part and another in the same packet) data will not be processed until all the threads are finished. Thus, the system will stop for a while, to process the borderline data.

As a result, the open source Network Intrusion Detection Systems (SNORT, 2010) is one of the famous tools known for intrusion detection in this field. In linear processing, SNORT-NIDS typically consumes 31% of the total processing time due to string matching and consumes 80% of the total processing time in the case of http traffic (S. Antonatos K. G., 2004). This means that, the matching process is the most expensive process. Since SNORT-NIDS is a real time IDS that required a lot of time to process the whole incoming packets (YU Jianming, October 2007). NIDS needs to increase its performance in high speed networks links.

The (Lambert Schaelicke, 2003; W. Lee, 2002; and Haoyu Song, 2005) proved in their experiments that on high speed networks, the software alone is not enough to process all the traffic. The exact string matching algorithms are used to detect intrusions, but these algorithms are insufficient to process all the network traffic in a linear phase. This is because; nowadays the speed link can reach up to 10 Gbps (Xiang, 2006; and Deri, September, 2007). The need for Network based Intrusion Detection Systems (NIDS) as complimentary software to the firewall is required to detect all kinds of intrusions whether the intrusion is of internal or external threads.

1.4 Research Goals and Objectives

The main goal of this thesis is to propose a new algorithm to enhance the speed of the intrusion detection engine based on the packets header called Distributed Packet Header Matching algorithm (DPHM). This thesis also proposed a new platform to enhance the speed of the detection engine for packets payload called New Network Intrusion Detection Systems platform (NNIDS). Furthermore, the NNIDS platform

can be used along with any exact string matching algorithms. Therefore, the objectives of this thesis are:

- To propose a new exact matching algorithm for enhancing the speed of the detection engine based on packets header in both sequential and parallel modes using multi threading technology.
- To enhance the speed of the existing SNORT-NIDS detection engine based on packets payload in real time by using multi-processors technology coupled with multiple-cores platform (Hybrid).

1.5 Contribution of this thesis

NIDS suffers from a slow speed of its detection engine in linear and in parallel modes, and a lot of overhead costs in some of the parallelized technology. Therefore, this thesis contribution can be summarized as follow:

1. Matching algorithm: A new algorithm called Distributed Packet Header Matching algorithm (DPHM) is introduced to enhance the speed of the detection engine for the packets header in a linear and multiple-cores platform.
2. Load Balancing: A new platform called the New Network Intrusion Detection Systems platform (NNIDS) is introduced to increase the speed of the detection engine for the packets payload in real time. By utilizing both:
    a. Multi-processing technology.
    b. Multi-processors technology with Multi-threading techniques (Hybrid).

The MPI stands for "Message Passing Interface". It is a library of functions in C that can be used to perform data communication between processes. The OpenMP is another example of a multiple-cores platform and it can be used to direct multi-threaded, shared memory programs on shared memory system. By using the OpenMP, the incoming packets payload will distribute among available cores that will reduce the synchronization between threads.

These two contributions will improve the performance of NIDS detection engine, and reduces the overhead problem resulting from the synchronization between the threads (see chapter 5).

1.6 Thesis Outline

This thesis is organized into six chapters. This chapter (Chapter 1) presents the objectives of this thesis. It starts by presenting a background discussion for the Network Intrusion Detection Systems (NIDS) along with our research objectives and contributions.

In Chapter 2, we discuss the most current and related works in NIDS. The researcher will also discuss the most important exact string matching algorithms used in NIDS. The reasons why we choose the methodologies for our system are discussed

Chapter 3 covers the methodology discussion on how the proposed solution was designed. The new algorithm for the packets headers detection is introduced in this chapter. The new platform for the packets payload is also described in this chapter.

The implementation details and issues are discussed in Chapter 4. The illustration of the experimental direction and the implementation of a real time detection engine are also mentioned.

The results obtained from the experiments in Chapter 4 are the primary content of Chapter 5. This chapter is divided into two parts. The first part reports the results of the detection engine for the packets header in a linear and on a multiple-cores platform. The second part reports the result of the detection engine for the packets payload on multi-processors with multiple-cores platform.

Finally, in Chapter 6, the conclusion, recommendation and the possible future work for this study are presented in details.

CHAPTER TWO

RELATED WORKS

2.1 Introduction

This chapter will explore the related works of the detection engine in Network Intrusion Detection Systems (NIDS).The detection engines categorized into two types of detection engine namely: the detection engine for packet header and the detection engine for packet payload. Each type is discussed with the corresponding related works. Later, the proposed solution is also described in this chapter, while the design and the methodology of the proposed solution are discussed in detail in the Chapter Three.

2.2 Related Works

2.2.1 Intrusion Detection Systems Based on Packets Header

Any pattern based NIDS must contain a detection engine as part of its components. This detection engine is responsible to detect any intrusion that exists in a packet. Accordingly, the detection engine is the most important part in NIDS and therefore, many of the researchers previously tried to enhance this part to increase the performance of NIDS through modifying the above exact string matching algorithm, or by proposing a new algorithm altogether. However, in this section, the related

works of the detection engine for packets header are divided into two groups based on their techniques:

- Group 1: Matching Algorithms.
- Group 2: Genetic Algorithm.

2.2.1.1 Matching Algorithms:

1- Boyer-Moore Algorithm

Boyer-Moore algorithm is considered as one of the most famous pattern matching algorithms, one that is considered very fast in practice, and it was designed for the exact string matching of many strings against a single keyword (Lecroq, 2004). The first heuristic phrase used is "*bad character shift*". Bad character shift starts a comparison from the right to the left and if a character is seen that does not exist in the text to search for, then the search algorithm can be shift forward to an "M" character where "M" is the length of the pattern. The second heuristic phrase used in the Boyer-Moore algorithm is "good suffix shift". Good suffix shift starts a comparison from the right to the left and if it is matches, then the algorithm check the next character in the text with the next character in the pattern, until matching all the strings. In the case of mismatching, the Boyer-Moore algorithm is looking for the next occurrence of a substring that was matched before.

Boyer-Moore algorithm suffers from two issues: (Zhou Chunyue, 2006) (RONG-TAI, August 2004)

3. Table skips function: This function is complicated and is used only in the case of a short text. When BMBC (Boyer-Moore Bad Character) shift fails

to find a character in a short text, then Boyer-Moore returns to the values in the table skip function to determine the number for further shifting.

4. Boyer-Moore algorithm depends on the text information and rarely refers to pattern information. Usually, all the exact strings matching algorithms will determine the number of shifting according to the characters in the pattern because the pattern is usually shorter than the text.

Therefore, (Rafiq, 2004) modified Boyer-Moore algorithm as follows:

1. The mismatched character of pattern is searched in the text instead of mismatched character of text being searched in the pattern. This technique makes the searching process working faster.
2. There is no need to calculate the distance between the two sub strings as Boyer-Moore algorithm does, because it is costly to compute.
3. He developed an algorithm to find all the occurrences of a pattern in the text.

2- AC_BM algorithm (Aho-Corassick, Boyer-Moore)

AC_BM algorithm examines the text from the right to the left using a common prefix approach instead of a common suffix approach. The keyword tree moves from the right end of the packet payload to the left end, while the character comparisons are performed from the left to the right.

The AC_BM algorithm can be used only by one of two ways: Firstly, the AC_BM algorithm used "*bad character shift*", which is similar to the heuristic of Boyer-Moore algorithm. Meaning, if a mismatch occurs, then AC_BM algorithm will

automatically shift to the next occurrence in some other keyword in the pattern tree. If it is a mismatch, the AC_BM algorithm will shift according to the length of the smallest pattern in the tree. Secondly, the AC_BM algorithm used the “good prefix shift”, which is similar to the “bad character shift”, but here the AC_BM algorithm will shift to the next occurrence according to the smallest pattern length (C. Jason Coit, 2001).

The AC_BM algorithm is faster than the Boyer-Moore algorithm from between 1.02 to 3.32 times. On the other hand, the AC_BM algorithm does not support any priority search in its searching phase. In non case-sensitive patterns as well, the AC_BM algorithm needs an additional structure to solve this problem. (Rong-Tai Liu, 2004). On the other hand, the AC_BM algorithm has a problem which is the maximum number of shifting depends on the length of the shortest pattern size

3- Quick Search Algorithm

The Quick Search algorithm is more simplified version of Boyer-Moore algorithm, but the Quick Search algorithm used only the “*bad character shift*”. The Quick Search algorithm work like a Horspool algorithm as well by working on one of two shifts of pattern. The Quick Search algorithm is easy to implement and is very fast in practice for short and large patterns (Lecroq, 2004).

4- The Horspool Algorithm

The Horspool algorithm looks like the Quick Search algorithm and Boyer-Moore algorithm but in a slightly different way. The Horspool algorithm works in any order, and the average number of comparisons for one text character is between $1/\sigma$ and $2/(\sigma+1)$ (Lecroq, 2004).The Horspool algorithm has a teething problem like the AC_BM algorithm which is the maximum number of shifting depends on the length of the shortest pattern size (Rong-Tai Liu, 2004).

Table 2.1 summarizes the complexity, preprocessing phase and searching phase for the mentioned algorithms.

Table 2.1: Complexity of some exact string matching algorithms (Rafiq, 2004)

Algorithm Name	Complexity		
	Preprocessing Phase		Searching Phase
	Space	Time	
Boyer-Moore	*O(m + |∑|)*	*O(m + |∑|)*	A: *O(mn)*
Horspool	*O(|∑|)*	*O(m + |∑|)*	A: *O(mn)*
Quick Search	*O(|∑|)*	*O(m + |∑|)*	A: *O(mn)*

Where A is the average case, m is the pattern length, and n is the string length

2.2.1.1.1 Rule Tree Nodes (RTN) and Option Tree Nodes (OTN) Techniques

(C. Jason Coit, 2001; and S. Antonatos K. G., 2004) find out that SNORT-NIDS relies on pattern matching to determine the intruders, and that the total number of rules is growing from time to time. Therefore, SNORT-NIDS divided its rule sets into two dimensional link lists as follows:

- The Rule Tree Nodes (RTNs)
- The Option Tree Nodes (OTNs)

The Rule Tree Nodes (RTNs) hold the main information of each rule, such as: source/destination address, source/destination port and protocols type (i.e. TCP, ICMP, UDP). While The Option Tree Nodes (OTNs) hold the information for the various options that can be added to each rule such as: TCP flags, ICMP codes and types, packet payload size and packet payloads. These two structures are organized into chains where the RTNs are strung from the left to the right as the chain headers and the OTNs are hanging down from the RTNs'. Figure 2.1 depicts the structure of SNORT-NIDS rule sets.

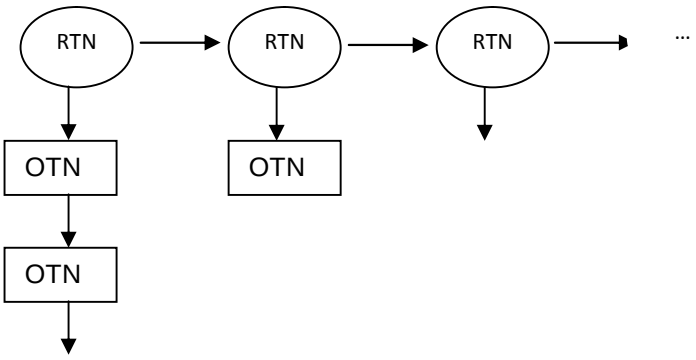

Figure 2.1: Structure of SNORT-NIDS rule sets (C. Jason Coit, 2001)

Through analyzing the SNORT-NIDS rule sets, there are over 3100 rules in SNORT-NIDS 2.3.3 (Li X. W., 2006), 200 rules are header only without payload. Therefore, the first procedure is to examine the RTNs' from the left to the right, until the packet matches the RTN. Then, the SNORT-NIDS detection engine checks the packet against each OTN in the chain, until a match is found. To increase the performance of SNORT-NIDS, it is better to leave the payload match to the last phase, because checking the payloads required more time due to its various sizes. Therefore,

SNORT-NIDS checks all the other options before a payload match. If there is a need to check a payload, SNORT-NIDS will utilizes the Boyer-Moore pattern matching algorithm to check the payload string held in the OTN against the entire packet payload and if there is no match, SNORT-NIDS will then check the next OTN in the list.

Algorithm: Boyer-Moore algorithm cannot determine the rule that it may match with the incoming packet header directly, and must search from the beginning to the end of the header rule set. Also, Boyer-Moore algorithm has to create the Boyer-Moore Bad Character (BMBC) table for each incoming packet and good suffix shift for each header rule set (wasted time).

Load Balancing: SNORT-NIDS structure does not Load Balancing the incoming packet header into available processors/cores. It will process the first header rule set with the incoming packet header, if matching then it will process the packet payload, else it will go to the next packet header sequentially and so on.

2.2.1.1.2 Packet Filter (PF) Algorithm

The Packet Filter (PF) algorithm classified each packet into two fields: Header and Payload (Randy Smith, 2006). The Header contains the main information of the packet such as: source/destination address, source/destination port, protocol …etc. They defined a packet filter as a rule sets since these rule sets determine which packets are allowed to pass and which are not allow. When the incoming packets arrived, its header information will be examined to check; if the packet header matches with one of the rule sets (Yang, 2003).

In some cases, there are more than one rule set that can be matched with any of incoming packets. In this case, the PF algorithm will take the highest priority. Therefore (Yang, 2003) divided the packets filters into two policies. The first policy is allowing all the packets to pass except for the specific type of packets. The second policy is considered contrary to the first policy in that it is rejecting all packets except some specific types of packets that can be allowed according to the policy. Furthermore, the authors divided their algorithm into two parts: build algorithm and searching algorithm. The build algorithm is used to build the decision tree, while the searching algorithm is used to find the matched rule in the decision tree for an input packet.

Algorithm: The PF algorithm is also used exact string matching algorithm to find the pattern in the string. Hence, PF algorithm has to create the BMBC table for each incoming packet and good suffix shift for each header rule set.

Load Balancing: The PF algorithm does not use any Load Balancing for the incoming packet header to spread the process into available processors/cores. It builds the decision tree and search on it sequentially.

2.2.1.1.3 Early Filtering (EF)

Some researchers focus on developing a new architecture to split the packet headerinto chunks of packets (I. Charitakis, 2003). The author's categorizes the NIDS rulesets into two categories: the header rule set without any payload, and the header ruleset with a payload. The former category is called the Early Filtering (EF) rule set. When the incoming packet reaches to the EF rule set without payload. It will

check if it matches with one of the EF rule sets. If it matches, then the packet will be discarded but if there is no rule matches even though the packet has a payload, then the packet will be sent to one of the sensors to evaluate the packet again. The researchers have also developed the locality buffer technique to improve the performances of NIDS sensors. The locality buffer technique is based on the fact that each specific rule sets is for a specific type of traffic. Here it means that each sensor will process the corresponding rules of that traffic. Figure 2.2 depicts the EF splitter architecture

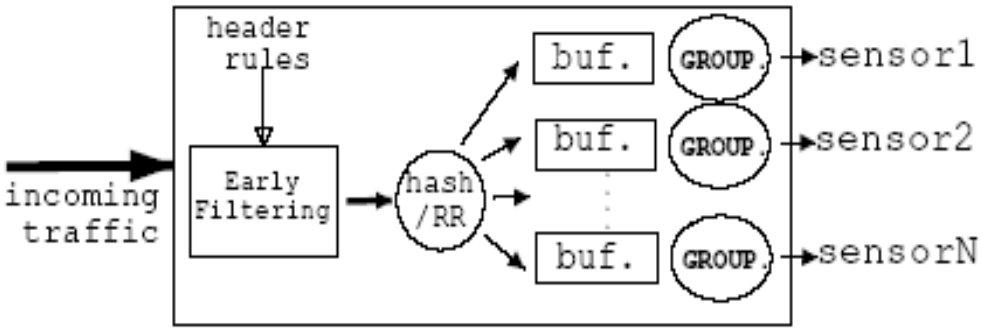

Figure 2.2: NIDS splitter architecture (I. Charitakis, 2003)

Algorithm: The EF splitter architecture split the incoming packets header into buffers according to the specific type of traffic. In this approach, the buffer that is involved to save the http traffic will be always full because 80% of the traffic is http traffic. Therefore, the speed of processing time will not enhance because this will be similar to the same scenario if we used one buffer.

Load Balancing: The EF splitter architecture does not distribute the incoming packets header into available processors/cores. It is working in sequential mode only.

2.2.1.1.4 Packet Filtering

Some other researchers considered the packet filtering process as two sequence procedures (Yoshiyuki Yamashita, 2007) as follows:

- Classify each incoming packet into one of three possibilities: To accept, to drop, and to forward.
- Action that will be taken according to the classification procedure.

Thus, to make the loop optimization, (Yoshiyuki Yamashita, 2007) divided the loop into two loops. For the first loop that is consists of logical and arithmetic operation, it must process multiple consecutive packets together with software in a pipelined procedure, that is considered a highly sophisticated aggressive instruction scheduling technique for loops (Yoshiyuki Yamashita, 2007).

Algorithm: The Packet Filtering software tool is suffering from a high complexity (O( )) because it divided the loop into two loops where each loop process different function.

Load Balancing: The Packet Filtering software tool working on sequential mode where the system will finish from the first loop and then process the second loop. Notice here, loop 2 is fully depending on loop 1. Therefore, this software tool is impossible to work in parallel mode.

2.2.1.2 Genetic Algorithm (GA)

Genetic algorithm is an algorithm that can be used to find exact or approximate solutions that achieve the optimized and search problems (Goldberg, 2005). Therefore, some researchers used a Genetic Algorithm (GA) in the Intrusion Detection Systems to create the rule sets from the network traffic. These rules are stored in the rule base and take the following form:

IF {Condition}

Then {act}.

Where *condition* refers to the matching case between the current network connection and the rules in IDS (e.g. source/destination address, source/destination ports, protocol ...etc), while *act* refers to an action that is defined by the security policies within the organization (e.g. reporting an alert, stopping the connection, etc...) (Chris Sinclair, 1999).

The goal from applying Genetic Algorithm (GA) into IDS is to generate rules to match anomalous traffic. These rules are used to filter new traffic to catch all suspicious packets in the network. In addition, these rules are tested based on historical traffic. These data sets are gathered using any popular network traffic filters such as SNORT-NIDS, TCPDUMP (www.tcpdump.org) and Ethereal (www.ethereal.com).

Other researchers used GA on a small random rule set (Li W. , 2004). GA can easily generate a large amount of data set that contains the rule sets for IDS. These rule sets are enough for filtering any new network traffic. They find out that there are many

parameters that can be considered for the application of GA, and the evaluation function is considered one of the most important parameters in GA. Figure 2.3 depicts their proposed algorithm.

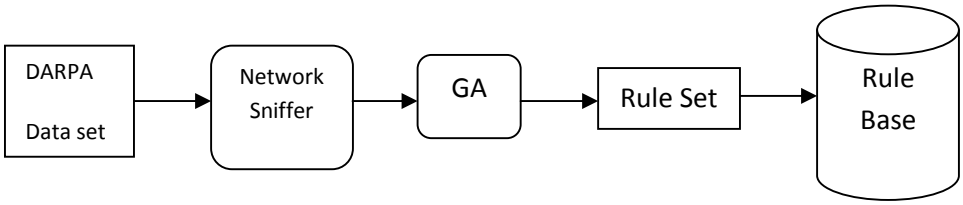

Figure 2.3: Architecture of applying GA into intrusion detection (Li W. , 2004)

Algorithm: Genetic Algorithm creates rule sets from the suspicious traffic. Therefore, the rule set will be increased continuously, which will lead to increase the searching/computational time.

Load Balancing: The two mentioned previous works that are used Genetic Algorithm in their proposed solution don't not load balance the incoming packets header into processors/cores.

2.2.2 Intrusion Detection Systems Based on Packets payload

In fact, some researchers have used dedicated hardware to enhance the speed of the detection engine (S. Ioannidis, 2002), (Sarang Dharmapurikar, 2003), and (Lockwood, 2005). Software based solution has a slow speed; it only performs lightweight processing on low network links. On the other hand, hardware based solution is faster and perform intensive processing on network traffic and supports a much higher network speed (Lambert Schaelicke, 2003).

However, the detection engine for packets payload should be faster than the detection engine for packets header. Because usually, the size of the packet payload is larger than the size of packets header that is always fixed size.

In this section, the researcher mentioned the related works of the detection engine for packets payload into two groups:

- Sequential processing.
- Parallel processing.

2.2.2.1 Sequential Processing

In this section, we are going to briefly describe the related works of the detection engine for packets payload using Exact Matching Algorithm.

The Misuse Detection Filter is considered as the simplest type of filters and it works by looking for a specific signature in the packet over the network traffic where this signature is called rule. Meanwhile, the Anomaly Detection is used to monitor and detect special types of events in a system. This means that a system will generate an alert when there are some changes that happen to the normal system behavior. Therefore, the researchers find out that there is a subsystem from anomaly detection called the Protocol Anomaly Filter, which is looking for specific types of protocols that are misused (Defcom, 2001). The Protocol Anomaly Filter can be used to detect all kinds of attacks that are trying to use the protocols outside normal usage.

The Misuse Detection filter depends on signatures to detect intrusions. These signatures are already registered in a database to detect the attacks. However, the database will grow up exponentially causing performance degradation to the misuse detection engine.

2.2.2.1.1 Developing a new architecture for the Misuse Detection Engine

However, the (Eduardo Mosqueira-Rey, 2007) proposed an architecture for The Misuse Detection as depicted in Figure 2.4.

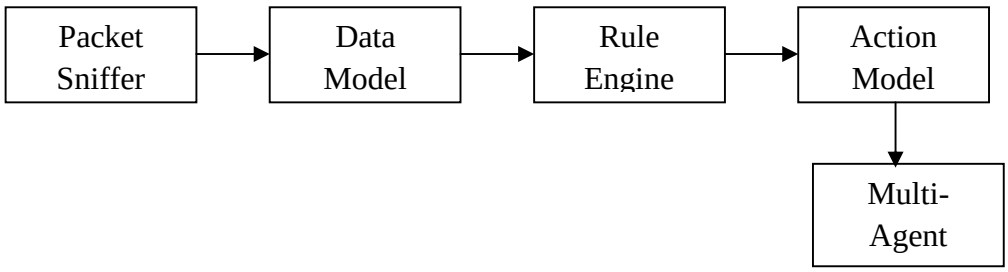

Figure 2.4: Eduardo's Misuse Detection architecture (Eduardo Mosqueira-Rey, 2007)

From Figure 2.4, the authors have designed a new architecture that is similar to SNORT-NIDS but in the JAVA environment. They also use Rete algorithm for the pattern matching process. Specifically, an implementation in JAVA language (drools-JBoss Rules) was used, and a parser was implemented that converts SNORT-NIDS rule sets to Drools rules.

Finally, they grouped the advantages and disadvantages of the Misuse Detection and Anomaly Detection as depicted in Table 2.2.