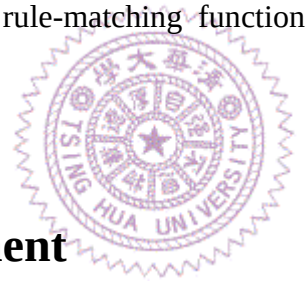
Chapter 1
Introduction

Until recently, the more complex network environment becomes, the more kinds of rules in Intrusion Detection System (IDS) or firewall we have to handle. However, the bottleneck of performance in those network inspection devices is still their rule matching algorithm [1] [2]. According to the evolution of Snort rules, there are more rules which focus on content checking, especially the relationships between some keywords in content. The facts show us in trend the rules in IDS or firewall will become more detailed to detect various intrusions. Therefore we need a powerful and efficient matching engine of rule-matching function to solve the problem in this modern world.



1.1 Problem Statement

In traditional cases, for example, supposed that there are *N* rules with length *L*, it has to take *O (N*L)* to check these rules linearly when a packet enters. Thus with the increase of rule numbers, traditional IDS or firewall may waste lots of time to check rules. If there is a faster rule-matching engine, it could save our time when comparing all the rules.

Focus on the relationships between every keyword in content, IDS or firewall usually handles them after picking those matched objects, like post processor does. It is indeed a solution but not a good one when we have lots of complex rules to handle. In facts, there are more and more rules about pattern-relationships and PCRE (Perl Compatible Regular Expression) pattern-matching, such as newly Snort rules. It is necessary to create a fast matching system with wide flexibility to process those rules

which may be written in Regular Expression [3] [4].

Third, the size of data structure in rule-matching algorithms is also an important considering factor, especially for implementation of hardware. If the size is small enough, it is easier to store the whole data structure in smaller but faster memory technology to upgrade the performance. Lower space requirements will bring us better efficiency.

From those reasons mentioned above, it is desired to build a pattern-matching system with the following characteristics [5].

- Faster Search Speed
- Low Storage Requirements
- Ability to handle complex matching, especially pattern-relationships and Regular Expressions

To achieve these goals, first we survey several papers about pattern matching algorithms and there are some introductions and comparisons in Chapter 2. After determining which algorithm could be used to implement our automata system, the tree data structure is used to construct our custom automata for processing Snort rules and make some changes in data structure from original AC algorithm to match our needs. Finally the proposed custom automata system is implemented for Snort rules in software as simulation and make some experiments about comparisons between custom automata and Snort system in Chapter 3. With the attractive characteristics of the proposed custom automata system, a hardware architecture is also designed for implementing it and raising the performance of our system. In order to develop a real multi-pattern matching system, there are many special designs in our hardware architecture and make custom automata suitable to implement in hardware. In Chapter

4, we discuss how to process Regular Expression efficiently. The most common solution is to use EGREP and PCRE as single-pattern matching algorithm. However, there are more and more complex Regular Expression rules now and single-pattern matching algorithm may not satisfy our needs. The AC algorithm and EGREP are employed to construct Regular Expression automata as multi-pattern matching algorithm in Chapter 4. In the comparisons between our Regular Expression automata and EGREP, we find that Regular Expression automata are faster than EGREP. It could handle multi-pattern matching of Regular Expressions easily and solve the bottleneck of performance. Thus according to our design, custom automata hardware system has good performance and powerful functionality. It is one of the best solutions as being real-time IDS and our ideas could make everything of writing intrusion rules or designing automata IDS easier. With low memory requirements and nice performance, the custom automata hardware system is a good choice of designing next generation IDS.