Chapter 7

Conclusions

In this dissertation a stream processing engine in a Network Intrusion Detection Systems has been proposed. First, we proposed a FSS filter which can mitigate SYN Flood attacks efficiently. In our test, all the attacking tools failed to exhaust the victim server's TCP state so that the availability of its web service is assured. We cannot find even one attacking tool against the weakness of FSS filter, but if it does exist, this mechanism can still reduce at least 50% attacking rate so that it can work in conjunction with other defense mechanisms such like SYN-Cache, SYN-Proxy, and so on. The extra memory space for FSS filter is much relatively small than the others, and this design requires less computing power as well.

Then we proposed a TCP scrubbing engine which erases the ambiguities of TCP evasion packets in several ways: header field sanity check, state transition check, out-of-order handling, and overlapping resolution. Our design can pass the evasion test of both NSS and OSEC which are the most reputable and professional NIDS test. This design hardly copies data and its memory consumption is relatively lower than the others since we don't need a stream buffer for each connection but a low-cost tree structure. Moreover, it can be easily integrated into existing NIDS implementations since it fulfills the general layer-4 functions comprehensively.

The first proposed pattern search algorithm, *FNP*, is a Network Processor-based algorithm that utilizes the hash engine of the Network Processor to achieve high performance. Network Processors usually lack of cache memory so that accessing main memory are quite expensive in this environment. Unfortunately, pattern matching algorithms usually need to access memory quite frequently. For example,

the Aho-Corasick algorithm needs to access main memory for every single byte in packet payload. The proposed *FNP* algorithm outperforms other alternatives in this way so that its performance is quite good in our test.

The second proposed algorithm is the FNP^2 algorithm. This algorithm is not for Network Processor platform specifically but a general software-based solution. The FNP^2 algorithm is modified from Wu-Manber algorithm (*MWM*) and needs less memory access than *MWM* does, especially when the size of shortest pattern is small. According to current snort ruleset, there're a lot of patterns whose size is less then three, therefore FNP^2 is quite suitable to be implemented in a software-based NIDS. We also implemented FNP^2 algorithm in a Network Processor platform, and its performance is better than other competitors according to our experiments.

Besides, this dissertation also presented a novel and fast hardware-based pattern matching engine, *FTSE*, which can process in up to 6Gbps to 8Gbps. *FTSE* comprises a TCAM to store signature prefix, a DDR SDRAM to store the whole signature database, and an ASIC/FPGA where the main algorithm running. There are two stages in *FTSE*, and the first stage is a pre-filter which filter out the strings impossible to match while the second stage performs exact match between a 9-byte matched candidate and the signature. These two processes work in parallel. In our simulation, the probability of passing strings to the second stage is only 0.2% to 2%, and the first stage runs seven-time faster than brute-force algorithm. On the other hand, the accuracy of the pre-filter in the first stage is over 90% in our test.

The future work of our research topic is to develop a more scalable pattern-matching mechanism which can support more than 32K patterns and regular expression syntax so that not only NIDS but Anti-Virus application could be satisfied.