# Exploiting Stateful Inspection of Network Security in Reconfigurable Hardware

Shaomeng Li, Jim Tørresen, Oddvar Søråsen

Department of Informatics
University of Oslo
N-0316 Oslo, Norway
{shaomenl, jimtoer, oddvar}@ifi.uio.no

**Abstract.** *One of the most important areas of a network intrusion detection system (NIDS), stateful inspection, is described in this paper. We present a novel reconfigurable hardware architecture implementing TCP stateful inspection used in NIDS. This is to achieve a more efficient and faster network intrusion detection system as todays' NIDSs show inefficiency and even fail to perform while encountering the faster Internet. The performance of the NIDS described is expected to obtain a throughput of 3.0 Gbps.*

## 1 Introduction

"Stateful inspection" is applied in Network Intrusion Detection Systems (NIDS) and is a more advanced network security tool than firewalls. It is used for checking the handshakes in a communication session by using detailed knowledge of the rules of the communication protocol. This is to make sure that it is completed in an expected and timely fashion. By checking a connection (packet by packet) – not just one single packet, and knowing what has just happened and what should happen next, stateful inspection detects incorrect or suspicious activity and alerts flags to the system administrator [1].

### 1.1 TCP connection stateful inspection

TCP (Transmission Control Protocol) [2] is an important Internet protocol. It provides a full duplex reliable stream connection between two end points in the TCP/IP network. The approach of using stateful inspection will be one of the best ways (maybe the only way) to monitor a TCP connection.

### 1.2 Snort – one Network Intrusion Detection System

In NIDS Snort, a software based STREAM4 preprocessor with 3000 lines software code is designed to conduct the TCP stateful inspection performing two functions: Stateful inspection sessions (monitoring handshakes) and TCP stream reassembly (collecting together packets belonging to one TCP connection). Testing Snort on various networks has shown that the STREAM4 preprocessor leads to a bottleneck in Snort for some network traffic environments (details can be found in [3]). Thus, to improve the performance of Snort, we would like to explore how reconfigurable hardware might be used to replace the STREAM4 TCP stateful inspection.

## 2 Exploiting New Implementation Methods for TCP Stateful Inspection

The new approach would be to process the stateful inspection in hardware rather than software as usual. Implementation in Field Programmable Gate Arrays (FPGAs) is appropriate to make such explorations. The new hardware architecture is proposed in Fig. 1. This unit will be an add-on unit for the computer running the Snort software. Incoming packet data (32 bit width) is input to the

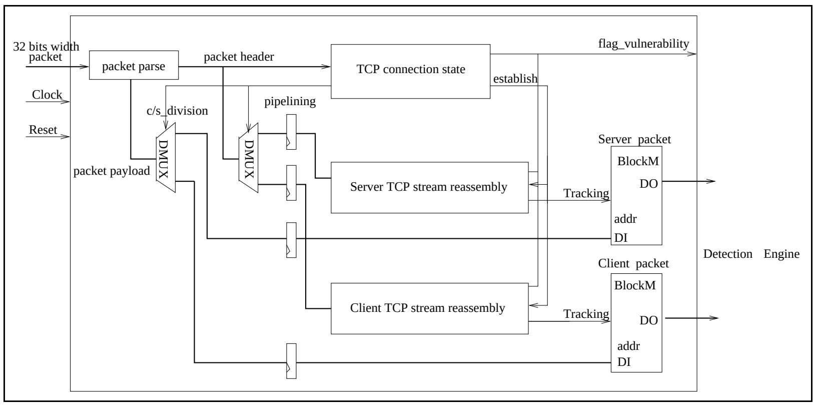

**Fig. 1.** Block diagram of reconfigurable hardware on TCP connection.

reconfigurable hardware unit which processes the TCP three way handshake and the Server and Client TCP stream reassembly. The information of the packet header will be stored in some registers based on the libpcap library which is used in Snort to get a packet off from the wires.[1] The basic packet header information most frequently referenced are the sequence number, acknowledge number, window size and TCP flags such as the SYN and ACK bit.

The TCP connection state unit is implemented as a state machine to check the three way handshake of the TCP connection. After establishing the proper connection (by TCP three way handshake), the data over a TCP connection can be exchanged between the Client and the Server. The processing of data flowing to the Server side and the Client side can be performed separately and in parallel, even if the Server and the Client TCP stream reassembly units conduct the same function. This means that packets sent to the Server and the Client side are reconstructed individually in independent hardware units. By doing this, the processing of TCP stream reassembly units in a NIDS is accelerated, of course, at a cost of extra FPGA resources.

Two 32 bit DMUXs (one for header and one for payload) are added to separate incoming packets into the Server and the Client packets. The reason for doing this is to feed incoming packets into the Server TCP stream reassembly

[1] Registers for the packet header are not shown in Fig.1.

unit and the Client TCP stream reassembly unit, respectively. The TCP stream reassembly units are running in parallel and determine which packets need to be stored in the "Client packet"or the "Server packet" memory. This avoids the need for large TCP stream reassembly buffers.

Two 32 bit comparators and one 32 bit adder are needed to implement one TCP stream reassembly unit. If the sequence number of an incoming packet is outside of the band size (band size is decided by the initial sequence number (ISN) and window number), the packet will be dropped. The payload of the packet is otherwise stored into the "Server packet" memory or "Client packet" memory respectively, to reconstruct the data for a succeeding detection engine. By pipelining the TCP stream reassembly, the "Server packet" memory and the "Client packet" memory unit, the total performance can be enhanced.

The size of the packet memories are 5x32 bit (16 bit data bus). 5x32 bit is required as the signature pattern can be matched at a maximum of 5x32 bits in the succeeding detection engine [4]. However, a dual port (write/read) memory is required for the Server/Client packet units with minimum size of 5x32 bits. Using a dual port RAM for the packet memory is important to be able to receive new data when matching (reading) is concurrently undertaken.

Virtex XCV1000-6 FPGA to be used in this work contains RAM blocks called SelectRAMs. Each has a capacity of full synchronous dual ported 4096-bit memory and is ideal to implement the Server/Client packet unit. One such block SelectRAM can be configured as a memory with different data widths and depths. However, since dual port RAM is required, the maximum data width is limited to 16 bits. The library primitives, the RAMB4-S16-S16 is dual ported where each port has a width of 16 bits and a depth of 256 bits which is available in the XCV1000-6. By considering the size of the RAMB4-S16-S16 and the packet which has 32 bit data width, two such block SelectRAMs are therefore needed to implement *one* 32 bit data bus packet memory – see Fig.2. Since there are two packet memories (the Client and the Server), a total of four block SelectRAMs are therefore needed to implement the "Server packet" and the "Client packet" memory units. Processing the data flow on the Server

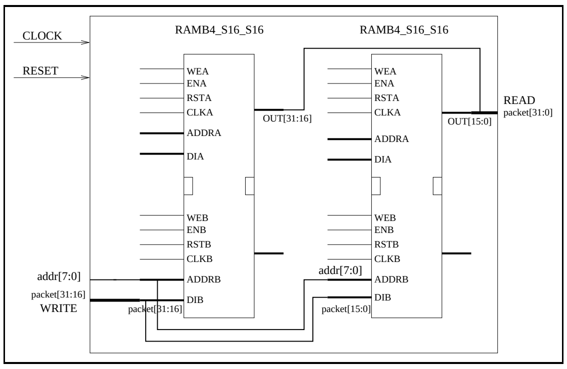

**Fig. 2.** The Server (or the Client) packet memory.

side and Client side in parallel and eliminating the need for a large reassembly buffer are our main contributions to improve the process of TCP connection in a NIDS. This makes it different from the approach in [5]. Data path processing in parallel is the main feature used when implementing the Server and the Client TCP stream reassembly in FPGA. Thereby the performance of NIDS facilitated by the method could be enhanced.

## 3 Experiments

Our implementation of this study is analyzed by using the ISE FPGA tool from Xilinx [6]. Designs are to be mapped onto a Virtex XCV1000-6 FPGA.

All individual modules such as TCP connection state, TCP stream reassembly unit and DMUX are implemented in VHDL. The simulation of those functions were conducted by the Modelsim XE II v6.5a simulator [6].

Except for the packet parsing, the whole system has been placed and routed into a XCV1000-6 FPGA. The minimum clock period for data from input to output is 10.467 ns which corresponds to a throughput of 3.06 Gbps.

However in IP/TCP networks, the Server often needs to be able to handle multiple connections simultaneously. Hence, multiple TCP connections have to be considered in this study. The process which consumes most SLICEs in the FPGA is the module which does doing the TCP three way handshake. Although there are 12288 SLICEs in one XCV1000-6 FPGA, the possibility of having multiple TCP connections is limited to the capability of implementing units of the “Server packet” memory and the “Client packet” memory in one such FPGA. The reason for this is that the height of the CLB array in one FPGA decides the number of block SelectRAMs, consequently determining the size of the packet units. One XCV1000-6 FPGA with the amount of 32 block SelectRAMs can therefore implement only 8 TCP connections. Although the size of the SLICES of such an FPGA should be checked to see if it is enough to implement remaining modules of 8 TCP connections simultaneously.

By using a Virtex XCV812E FPGA which has 280 block SelectRAMs as used in [5], 70 multiple TCP connections can be expected to be implemented in one FPGA.

## 4 Conclusions

Stateful inspection over a TCP connection is studied and implemented in FPGA based hardware to remove the bottleneck of TCP connection in a network traffic environment. A novel approach using reconfigurable hardware is introduced. Experiments show that the performance could be improved by this implementation to a throughput of 3.0 Gbps.

## References

1. Michael Clarkin. *"Comparison of CyberwallPLUS Intrusion Prevention and Current IDS technology"*. NETWORK-1, Security Solutions, Inc., White Paper.

2. J. Postel. *"Request For Comment 793, Transmission Control Protocol"*. 1981.
3. Sergei et al. *"SNORTRAN: An Optimizing Compiler for Snort Rules"*. Fidelis Security Systems, Inc., 2002.
4. Shaomeng Li et al. *"Exploiting Reconfigurable Hardware for Network Security"*. in Proc. of 11th Annual IEEE Symposium on Field-Progammable Custom Computing Machines(FCCM'03), 2003.
5. Marc Necker et al. *"TCP-Stream Reassembly and State Tracking in Hardware"*. in Proc. of 10th Annual IEEE Symposium on Field-Progammable Custom Computing Machines(FCCM'02), School od Electrical and computer Engineering, Georgia Institute of Technology, Atlanta, GA, 2002.
6. *http://www.xilinx.com.*