INTERNATIONAL JOURNAL OF COMPUTERS
Issue 3, Volume 2, 2008

H/W based Stateful Packet Inspection using a Novel Session Architecture

Seungyong Yoon, Byoungkoo Kim, Jintae Oh, and Jongsoo Jang

***Abstract*—** Stateful Packet Inspection(SPI) remember the previous packet and can thus keep track of the state of the session. SPI was originally developed for Firewall. But recently there are various applications such as VPN, NIDS, Traffic Monitoring, and so on. In this paper we focused on Network Intrusion Detection System(NIDS). Because stateless IDS only look at one packet at a time, a lot of false positive alerts generate during attempt to attack using IDS evasion tool, for example, "stick" or "snot". To prevent this problem, SPI was employed in NIDS and statefulness of NIDS became very important. But most of existing SPI products are software based solutions which have poor performance in current high-speed internet environment. So, in many cases, the SPI module in NIDS remains inactivated. That is against original purpose. Stateful IDS mainly depends on the performance of processing session table and pattern matching. Pattern Matching has been a lot of studied. But, relatively few studies have been devoted to session processing. It is so difficult that we manage a lot of session state information with limited hardware resources and satisfy high-performance. Therefore, our purpose is to design and implement SPI module in FPGA with new session management architecture. And then we prove that can achieve an efficient and fast stateful intrusion detection that supported up to 1 million sessions with high performance.

***Keywords*—**Stateful Packet Inspection, Network Intrusion Detection System

I. INTRODUCTION

One of the major problems and limiting factors with Network Intrusion Detection System(NIDS) is the high false positive alert rate. In order to reduce these false positive alerts, a lot of methods and techniques are proposed. Stateful Packet Inspection(SPI) is one of these solutions. SPI was originally developed for Firewall[1][2], but it became a very important factor in NIDS. Stateless NIDSs generate tremendous false positive alerts while stick or snot attempts to attack[3][4]. Most existing NIDSs have SPI module which is supported statefulness but they don't satisfy high-performance in gigabit internet environment. It is so difficult that we manage a lot of session state information with limited hardware resource and satisfy performance of high-speed internet. In other words, the rapid evolution of recent network technologies to gigabit network environments require existing SPI module to have more improved functions and performance[9][10]. SPI basically requires a session table which stores source and destination IP addresses and port numbers. It is necessary to perform real-time packet inspection by checking, for each input packet, whether or not a corresponding entry is present in the session table. Real-time packet processing at wire speed should not cause any packet delay or loss even when the number of managed sessions is increased to more than one million.

Previously developed software-based solutions cannot meet these requirements. One software-based technique has attempted to use a distributed system[11][13][14]. However, as the number of session increases, this technique requires a higher processing speed, thereby causing performance problems. Thus, software-based solutions cannot perform real-time packet inspection ensuring the wire speed.

To guarantee both performance and functionality with respect to statefulness, we designed and implemented SPI-based intrusion detection module in a FPGA to help alleviating a bottleneck in network intrusion detection systems in this paper. The performance of SPI-based intrusion detection system mainly depends on the performance of processing session table[5] and pattern matching[6][15][16][17][18][19][20]. In this paper, we focused on session state management scheme and omitted pattern matching method. Our work related to pattern matching method is described by Byoungkoo Kim at al. [7] and Dong-Ho Kang at al. [8]in detail.

The rest of this paper is organized as follows. Section II describes our system architecture and SPI-based intrusion detection module. And section III describes a novel session table architecture in detail. Implementation and experimental results are contained in section IV. And conclusion and future work are discussed in section V.

Manuscript received December 12, 2008: This work was supported in part by Korea Ministry of Information and Communication under "Next Generation Security System Development" Project.

Seungyong Yoon is with Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, Korea (e-mail: syyoon@ etri.re.kr).

Byoungkoo Kim is with Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, Korea (e-mail: bkkim05@ etri.re.kr).

Jintae Oh is with Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, Korea (e-mail: showme@ etri.re.kr).

Jongsoo Jang is with Electronics and Telecommunications Research Institute, 161 Gajeong-dong, Yuseong-gu, Daejeon, Korea (e-mail: jsjang@ etri.re.kr).

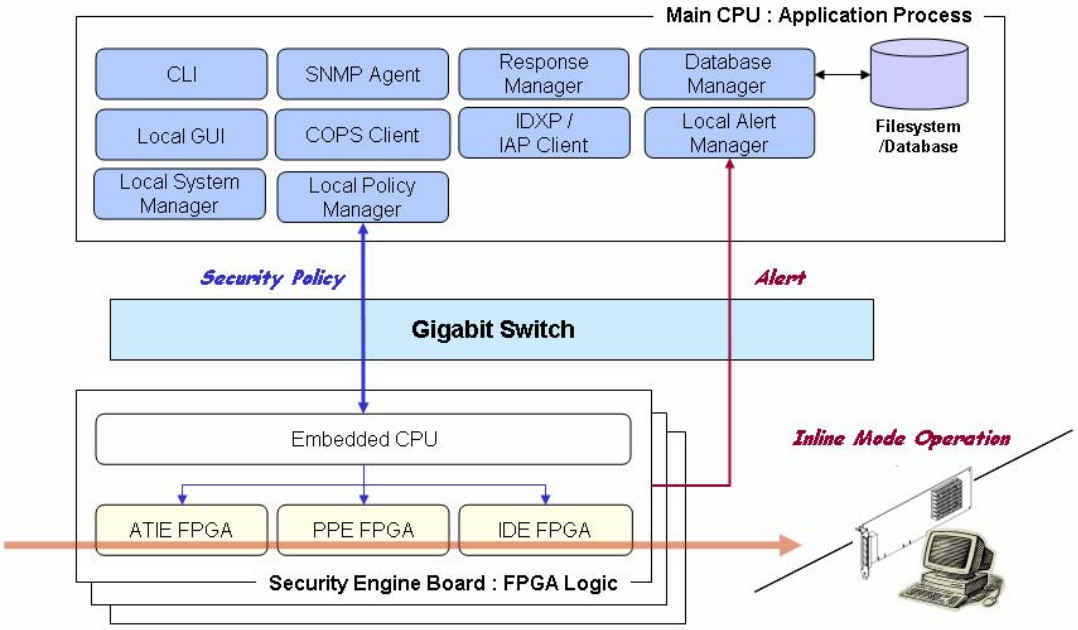


Fig. 2. The Architecture of SGS

II. SECURITY GATEWAY SYSTEM

A. System Architecture

At first, we briefly introduce the architecture of our system, named "Next Generation Security System"(shortly NGSS) designed to detect and response intrusions on high-speed links. NGSS has two main systems: Security Gateway System(SGS) and Security Management System(SMS). SGS is a security node system which is located at ingress point in protected networks. Security policies from SMS are applied and executed on SGS. This basic concept is shown in Fig.1.

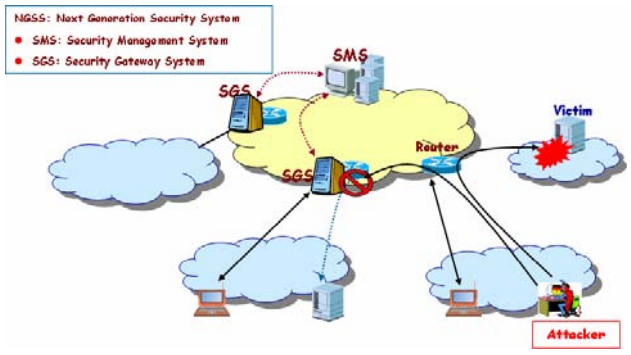


Fig. 1. Security Gateway System(SGS)

Hardware based and high performance SGS provide security functions such as session state management, protocol anomaly detection, pattern matching detection, rate limiting, packet filtering which are implemented on three FPGA (Xilinx Vertex II Pro) chips in each security engine board. Security engine board also has embedded CPU MPC 860 that embedded Linux OS operating in. Total two security engine boards can be installed to SGS. As shown in Fig.2, SGS consists of three parts; Application Process for communication channel with SMS and system management functions, FPGA Logic for wire-speed packet forwarding, packet preprocessing, high-performance intrusion detection and so forth, Gigabit Switch for communication between FPGA Logic and Main CPU. Again, we can divide FPGA Logic into several sub FPGA Logics: Anomaly Traffic Inspection Engine(ATIE), Pre-Processing Engine(PPE), and Intrusion Detection Engine(IDE) FPGA. Embedded CPU on the Security Engine Board manages the ruleset that is required for intrusion detection. Through the interoperability of these components, SGS analyzes data packets as they travel across the network for signs of external or internal attack. Namely, the major functionality of SGS is to perform the real-time traffic analysis and stateful intrusion detection on high-speed links. Therefore, we focus on effective detection strategies applied FPGA Logic.

B. Security Engine Board of SGS

Security Engine Board of SGS is composed of three FPGA Chips and one Embedded CPU aimed at real-time network-based intrusion detection. SGS is capable of managing these several boards according to network environments that it is applied. For detecting network intrusions more efficiently on high-speed links, our FPGA Logic is divided into three FPGA Chips. As shown in Fig.3, one is ATIE FPGA Chip for wire-speed packet forwarding and blocking., another is PPE FPGA Chip for packet preprocessing, and the other is IDE FPGA Chip for high-performance intrusion detection.

First, ATIE FPGA Chip uses the Xillinx FF-1517 FPGA Chip for its own logic. And, it is connected to the PM3386 for incoming packet processing and PM3387 for alert message sending. Also, it uses two external TCAM and two external SRAM for incoming packet scheduling and management. Briefly, the main function of ATIE FPGA Chip is wire-speed packet forwarding and response coordinator such as alert

response and packet filtering. Incoming Packets from PM3386 is sent to PPE FPGA Chip, and if it is determined as attack according to the analysis result from other FPGA Chips, alert message is sent to the main CPU through the PM3387.

Second, PPE FPGA Chip uses the Xillinx FF-1152 FPGA Chip for its own logic. And, it uses two external TCAM and four external SRAM for operating the session management, IP de-fragmentation and TCP reassembly. Briefly, the main function of PPE FPGA Chip is preprocessing the incoming packet. Through these preprocessing functions, it supports the SPI-based intrusion detection function and IDS evasion attack detection function. Also, it checks out the protocol validation about service protocol such as SMTP, HTTP and so forth. If the incoming packet is invalid against service protocol, alert information is sent to ATIE FPGA Chip.

Finally, IDE FPGA Chip uses the Xillix FF-1517 FPGA Chip for is own logic. It uses three mechanisms for high-performance intrusion detection; Flexible Header Combination Lookup Algorithm for packet header pattern matching, Linked Word based Stroreless Running Search Algorithm for packet payload pattern matching, and Traffic Volume based Analysis Algorithm for DoS(Denial of Service) and Port-scan detection. Through these mechanisms, it supports the high-performance intrusion detection function without packet loss.

Fig.3 depicts overall security board composition. Packet filtering, Rate-limiting, and traffic metering are implemented in ATIE chip. Stateful Inspection, IP Defragmentation, and Protocol Anomaly based detection function in PPE chip, and Pattern matching based detection function in IDE Chip. Each security board has two gigabit port interface.

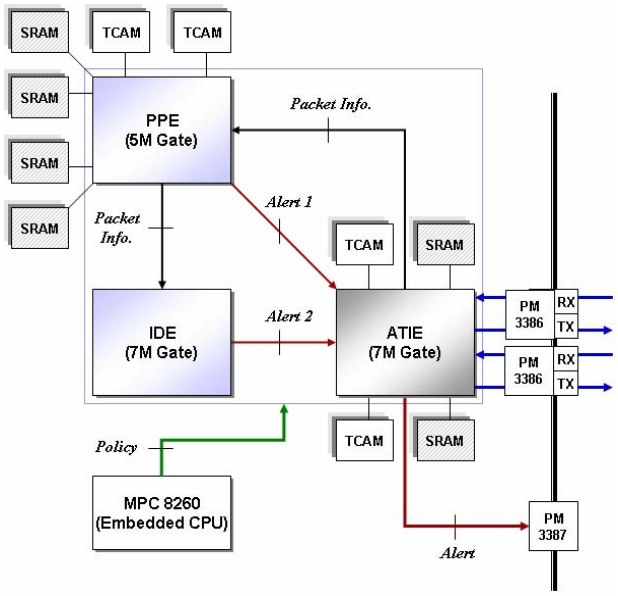


Fig. 3. Security Engine Board

C. SPI-based Intrusion Detection Module

Our SPI-based intrusion detection module was implemented on Security Gateway System(SGS). SGS is a security node system which is located at ingress point in protected network. Strictly speaking, SGS is network intrusion prevention system running in inline mode. Fig.4 shows the SPI-based intrusion detection module of our SGS. Legitimate TCP sessions are established through 3-way handshake and terminated through 4-way handshake. State manager has session table and tracks these session state. If input packet doesn't exist in session entries, this packet will drop or forward to Intrusion Detection Engine(IDE) with additional state information according to security policies.

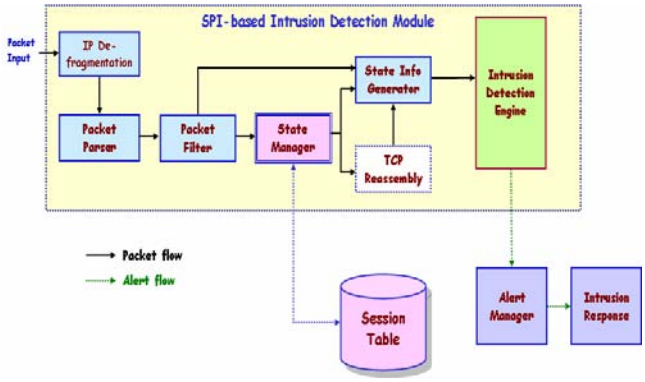


Fig. 4. SPI-based Intrusion Detection Module

At first, if the packet inputted from IP De-fragmentation sub-module, necessary information fields are extracted through packet parsing sub-module. Packet filter transfer to state manager only packet that passed by security filtering policies. These filtering policies can be applied to specific protocols or ports. Each sub-module has functions as follows:

- IP De-fragmentation: combines fragments of packets into packets.
- Packet parser: extracts necessary information through packet parsing process.
- Packet Filter: applies filtering policies and delivers to state manager only packet that passed filter.
- State Manager: manages session table and state information,
- State Info Generator: generates and transmits useful state information for detecting attack and abnormal packets to IDE.
- TCP Reassembly: reassembles the TCP segments in the right order.
- IDE: performs effective pattern matching with state information

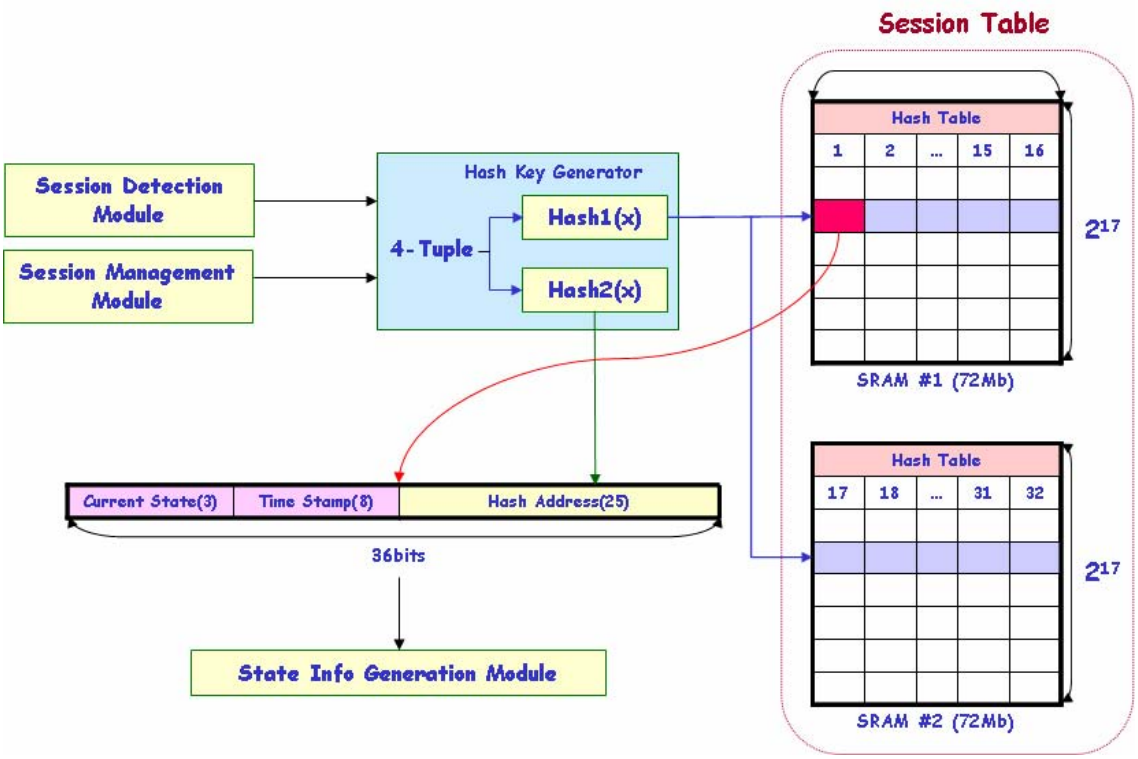


Fig. 5. Basic Architecture of Session State Manager

III. A NOVEL SESSION ARCHITECTURE

A. Basic Architecture

The terms "session", "connection" and "flow" are used interchangeably in this paper. Fig. 5 is a basic architecture of session state manager for stateful packet inspection. As shown in Fig. 5, session state manager includes a hash key generator, a session table, a session detection module, a session management module, and a state info generation module.

The session table stores session entries that are indexed and managed by the hash key generator. 4-tuple information including a source IP address, a destination IP address, a source port, and a destination port is input, as information used to hash a newly received packet, to the hash key generator. Once the packet is inputted, a packet parser extracts this information from the packet. The hash key generator indexes and manages a session entry corresponding to the received packet based on the input 4-tuple information. Hash key generator has a dual hash structure with two different hash functions Hash1(x) and Hash2(x). The hash functions Hash1(x) and Hash2(x) are well-known functions that are used to hash packets. For example, XOR or CRC functions can be used as these hash functions. One hash function "Hash1(x)" is used to generate indices that point to hash sets permitting hash collisions in order to achieve faster session table search. The other hash function "Hash2(x)" is used to generate hash addresses that are used to identify session entries in a hash set pointed by the hash function "Hash1(x)". Session table may be designed and implemented using two or more SRAM devices, if necessary. In this paper, the session table is constructed using two SRAMs (SRAM#1 and SRAM#2), which can be accessed simultaneously or in parallel using a hash set index that is generated by the Hash1(x) to achieve faster session table search.

The session table stores session data of packets inputted from an external network. For efficient session table management, the session table has an N-way set associative session table structure in which each hash set in the session table can include N session entries. The session table shown in Fig. 5 is a 32-way set associative session table that is constructed using two 72-Megabit SRAMs with each session entry having a length of 36 bits.

Each session entry stored in the session table includes current state, time stamp, and hash address parts. The current state part includes current connection state information of a corresponding session, the time stamp part is used to determine which session entry is to be deleted when the session table is full, and the hash address part is used to identify each session entry in the same hash set. The time stamp is updated by an internal timer each time a corresponding session is accessed. If any hash sets of the session table are full so that new session cannot be allocated to the hash sets, current time of internal timer is compared with the time stamp of each session entry to replace the oldest session with a new session. For example, Least Recently Used (LRU) algorithm is applied to this process.

Session state information is separately managed in embryonic state and established state for timeout mechanism. Embryonic state includes sessions that TCP 3-way handshaking does not finish, on the other hand established state includes completed sessions. It is necessary to manage

separately states, because embryonic session needs to have shorter timeout value than that of established session. The SPI devices and computers have vulnerabilities to SYN flooding attack in nature[21][22]. This mechanism helps to prevent against denial-of-service attack such as SYN flooding.

Although a Transmission Control Protocol (TCP) session is terminated without sending an RST or FIN packet, a corresponding session entry is immediately removed if a time stamp in the session entry exceeds a timeout threshold predetermined by the administrator. Accordingly, a session which has been terminated without sending an RST or FIN packet is positively removed from the session table.

The session detection module searches the session table according to the received packet. Specifically, the session detection module obtains a hash set pointer from Hash1(x) calculated by the hash key generator and then searches the session table for a session entry corresponding to the hash value from Hash2(x). The session management module performs a process for adding, deleting, and changing sessions of the session table in order to maintain the session table. The state info generation module generates state information regarding the direction of the packet and session establishment information and then transmits this information to intrusion detection engine with packet data.

Fig. 6 schematically describes a method for processing direction information included in each packet, which indicates the direction of the packet in a corresponding session. Each packet transmitted over the network includes information regarding the direction of the packet in a corresponding session, which indicates whether the direction of the packet is from the client to the server or from the server to the client. This information is very useful in a network intrusion detection or prevention system. However, the direction information may cause a significant confusion in searching for a corresponding session in the session table since the hash address of each packet belonging to the same session may vary depending on the direction. To prevent the hash address from varying depending on the direction, the hash key generator compares the value of a source IP address with the value of a destination IP address and modifies a corresponding 4-tuple value so that one of the source and destination addresses, which has the lower value, always precedes the other with the higher value. A specific flag is defined to indicate whether or not such a position change has been made. For example, a flag "Position_change_flag" is defined, which is assigned "1" when the position change has been made and "0" when no position change has been made. The "Position_change_flag" information is very efficiently used in generating state information together with current state information.

B. *Session State Information*

Session state information is stored in a current state part in each session entry. In consideration of hardware resource, we designed current state part to have 3 bits in a 36-bit session entry. The first bit of the current state part contains session establishment information. For example, when a session has been established between the client and the server, the first bit is set to "1" and, when no session has been established between the client and the server, the first bit is set to "0". The second bit of the current state part contains information indicating whether or not the source and the destination were reversed when the session was registered in the session table. This information is different from the information contained in the flag "Position_change_flag" shown in Fig. 6. The difference between the information contained in the second bit and the information contained in the flag "Position_change_flag" is described below in detail. The third bit of the current state part contains information indicating whether or not the connection is in a half-closed state. Each session is terminated only when the second FIN packet is received when the connection of the session is in a half-closed state. Namely, when the connection is in a half-closed state, the third bit is set to "1" and, when the connection is not in a half-closed state, the third bit is set to "0". The TCP connection establishment process is performed through 3-way handshake. When the client sends a SYN packet to the server to request it to establish a new connection, the server responds with a SYN/ACK packet and then the client sends an ACK packet to the server in response to the SYN/ACK packet, thereby completing the establishment of the connection. The TCP connection termination process is normally performed through an RST packet or an FIN packet. The FIN packet is transmitted through 4-way handshake. If one of the client and the server sends an FIN packet, then the other sends an ACK packet in response to the FIN packet. This state in which the first packet has been received is referred to as a "half-closed state". If the client transmits a second FIN packet in the half-closed state, then the server transmits a second ACK packet in response to the second FIN packet, thereby terminating the TCP session.

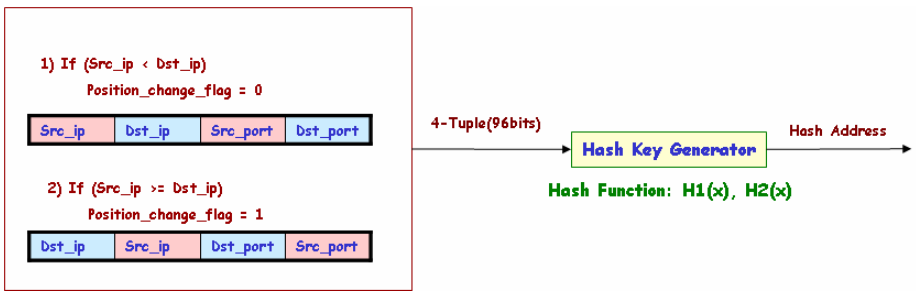


Fig. 6. Input of Hash Key Generator

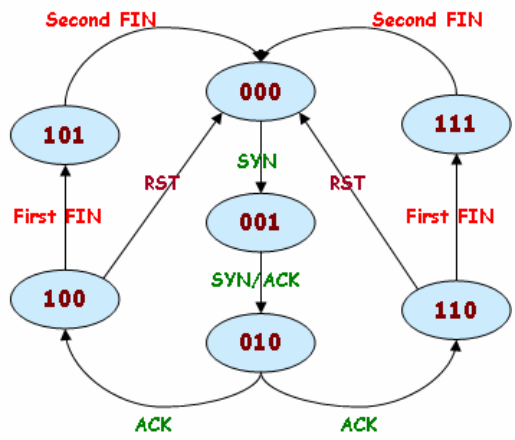


Fig. 7. Session State Transition Diagram

Fig. 7 is a state transition diagram showing the relationship between the 3-bit values stored in the current state part of session entry and input packet. The current state value is "000" in an initial state where no session has been established between the client and the server. If the client transmits a SYN packet to the server to establish a TCP session, the current state value transits to "001". Thereafter, if a SYN/ACK packet is transmitted, the current state value transits to "010". If the last ACK packet is transmitted in the state of "010" in the 3-way TCP handshake process for establishing a TCP connection, the value of the source is compared with the value of the destination. The current state value transits to "110" if the position change has been made. However, the current state value transits to "100" if no position change has been made. If the first FIN packet for terminating the TCP connection is transmitted in the "110" state, the current state value transits to "111". Thereafter, if the second FIN packet is transmitted in the "111" state, the current state value transits to the initial state value "000". If the first FIN packet is transmitted in the "100" state, the current state value transits to "101". Thereafter, if the second FIN packet is transmitted in the "101" state, the current state value transits to the initial state value "000". If an RST packet for terminating the TCP connection is transmitted in any one of the "110", "100", "101", and "111" states, the current state value transits to the initial state value "000".

Table 1 shows state information generated from a "Position_change_flag" value and a current state value stored in the current state part. The state information is generated basically using the current state value and the direction of each packet is determined from a combination of the current state value and the "Position_change_flag" value. For example, if the current state value is "100" or "101" while the Position_change_flag" value is "0", the direction of the current packet is from the client to the server since the source and destination of the current packet have not been reversed and the source and destination had not been reversed (i.e., the direction was from the client to the server) when the corresponding session was registered. On the other hand, if the current state value is "100" or "101" while the Position_change_flag" value is "1", the direction of the current packet is from the server to the client since the source and destination of the current packet have been reversed and the source and destination had not been reversed (i.e., the direction was from the client to the server) when the corresponding session was registered.

Table 1. State Information Generation

Position_change_flag	Current State	State Information
Any Value	000	Not Established
Any Value	001	SYN_RCVD(3-way Handshaking)
Any Value	010	SYNACK_RCVD(3-way Handshaking)
Any Value	011	Reserved(unused)
0	100, 101	Established, Direction: Client → Server
1	100, 101	Established, Direction: Server → Client
1	110, 111	Established, Direction: Client → Server
0	110, 111	Established, Direction: Server → Client

SPI module has a following procedure for processing a packet according to our proposed scheme. First, when a packet is inputted, a hash key value is generated using 4-tuple information extracted from the packet and a session table is searched for a corresponding session using the generated hash key value. If the corresponding session is found in the session table, its session entry information is updated. If the corresponding session is not found in the session table, a new session is generated only when the current packet is a SYN packet. If the session table is full, the oldest session entry is selected using the LRU algorithm and then replaced with the new session. If the session table is not full, a new session is generated for the session table. Once the session table for the received packet is constructed as described above, state information of the packet is generated. It is preferable that the method described in Table 1 be used to generate the state information of the packet. Then, inspection of the packet is performed based on the generated state information.

C. Performance Simulation

Our session state management scheme in SPI-based intrusion detection system is affected by two major factors, hash collision rate and miss rate. There is every probability of hash collision occurrence because hash function for faster session table search is used. The wrong state information is generated if the hash collision is occurred. Therefore, the SPI-based intrusion detection module generates the false positive alert. The hash collision rate is determined by the Hash

1(x) and Hash 2 (x). Theoretically, the probability of hash collision is $1/2^{42}$(Hash 1(x): 17 Bits + Hash 2(x): 25 Bits = 42 Bits)

As the number of session entries increase gradually, the session table is filled with new session. Also, there is every probability of miss occurrence because the size of hash set has limitation(32-way set). When the session table is full, the probability that each session is missed is very important in a session table management scheme because wrong session state information is generated if any existing session, which has not yet been terminated, is replaced with a new session. In this case, since the SPI-based intrusion detection module generates the false negative alert, the miss rate can be said to be the factor which is important than the hash collision rate. In order to ensure that miss rate is reasonable in our design, we made a simulation for distribution of the number of sessions allocated to each hash set in the session table when one million sessions are established. We used a separate set of traffic data collected from various network environments for this simulation. Fig. 8 shows the result of this simulation.

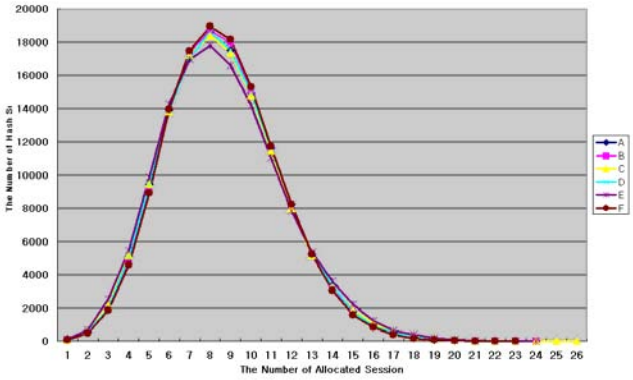


Fig. 8. Simulation Results

Distribution of the number of sessions allocated to each hash set in the session table follows a normal distribution as expressed by Probability density function(Equation (1)). This is standardized using Equations (2) and (3) and then the push-out probability of each session in the 32-way set associative session table is calculated to obtain P{X>32}= P{Z>8.3}. This indicates Z-score of 8.3 which is nearly 0%. (Z-score of 6 corresponding probability is 0.0003%.)

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \quad (1)$$

$$Z = \frac{X-\mu}{\sigma} \quad (2)$$

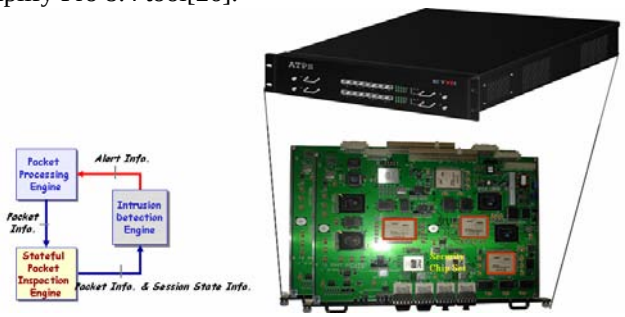
$$P(a < X < b) = P(\frac{a-\mu}{\sigma} < Z < \frac{b-\mu}{\sigma}) \quad (3)$$

According to the result of simulation, it is proved that our design for session state management is very reasonable with respect to hash collision rate and miss rate.

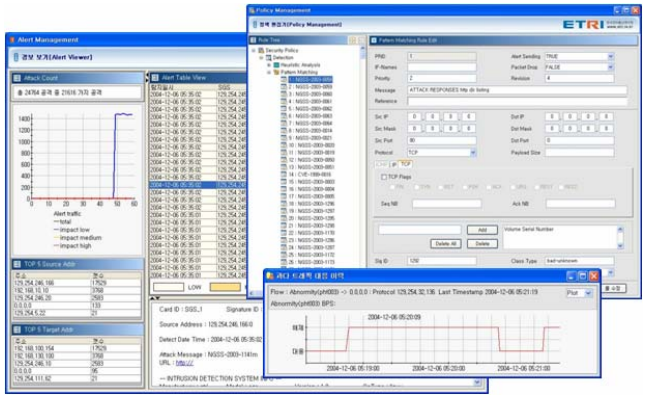
IV. IMPLEMENTATION AND EXPERIMENTS

A. Implementation

Our SPI-based intrusion detection module was implemented on SGS prototype. Session State Manager of SGS is implemented on a Xilinx Vertex-II Pro XC2VP50 FPGA(5M Gate)[23] and Cypress CY7C1470V33 SRAM(72Mbit)[24] using verilog HDL(Hardware Description Language) that is best suited for high-speed packet processing. The simulation of all functions were conducted by the ModelSim PE 6.1 simulator[25]. And all logics have been synthesized by Synplify Pro 8.4 tool[26].



(a) The Prototype of Security Gateway System



(b)A Screen Shot of the Policy and Alert Window

Fig. 9. Exampels of Implementation

In our prototype, main logics were implemented on three FPGA chipsets, Packet Processing Engine, Stateful Packet Inspection Engine, and Intrusion Detection Engine. Especially, the prototype we have developed focus on FPGA logic for real-time traffic analysis and SPI-based intrusion detection on high-speed links. Also, we employed inline mode capable of effective response by using four Gigabit Ethernet links as shown in Fig.9. The minimum clock period for data from input to output is 8ns which corresponds to a throughput of 2Gbps. That is, our system is capable of processing until a maximum throughput of full-duplex 2Gbps about incoming packets in FPGA Logic.

B. Experiments

If the SPI device is tracking TCP session state, then it has the potential to introduce denial of service when the session table becomes full(too many connections) or if it can't keep up with the creation of new sessions(too many connections per second). That is, Max Concurrent Sessions(MCS) and Connections Per Second(CPS) are very important factors for performance evaluation.

We made use of the test bed shown in Fig. 10 for performance evaluation of our prototype system. The test bed consists of IXIA Traffic Generator[27], Gigabit Switch, Spirent Avalanche/ Reflector[29], IDS Informer Attack Tool[28], and Nessus Vulnerability Scanner[30] for experiments.

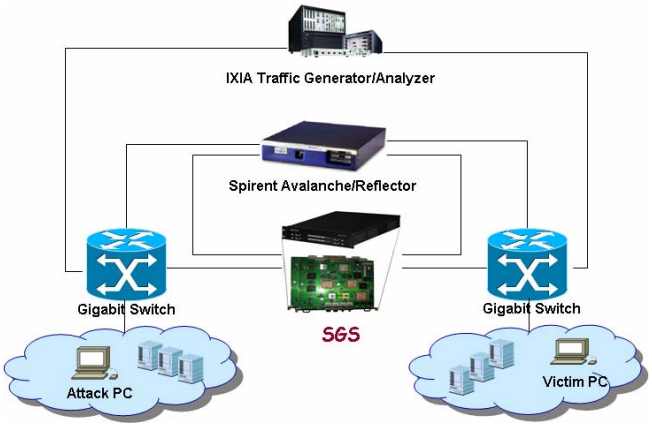


Fig. 10. Test bed for experiments

In the results of measurement using Spirent Avalanche/Reflector, our system supported up to 40,000 Connections Per Second.

Max Concurrent Sessions was measured as following procedure. First, a legitimate TCP session is opened through 3-way handshaking and then Spirent Avalanche opens various numbers of TCP sessions from 500,000 to 1,500,000 with the Reflector. Exploit is transmitted which is required to trigger an alert in the initial TCP session. If the Session State Manager is still maintaining state on the first session established, the exploit will be detected. If the state table has been exhausted, the exploit string will be seen as a non-stateful attack, and will thus be ignored. Table 2 shows the results of alert generation in this experiment. As a results of experiment, we can see that our system supported up to 1, 300,000 Max Concurrent Sessions.

Table 2. The Results of Max Concurrent Sessions

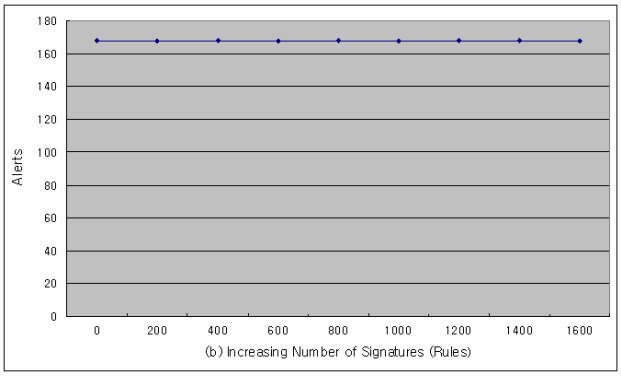
	First	Second	Third
500,000	Yes	Yes	Yes
600,000	Yes	Yes	Yes
700,000	Yes	Yes	Yes
800,000	Yes	Yes	Yes
900,000	Yes	Yes	Yes
1,000,000	Yes	Yes	Yes
1,100,000	Yes	Yes	Yes
1,200,000	Yes	Yes	Yes
1,300,000	Yes	Yes	Yes
1,400,000	Yes	No	Yes
1,500,000	Yes	No	No

Another experiment was made on detection rate. Detection rate is most important factor in SPI-based intrusion detection system. Generally, both traffic rate and the number of signatures have an effect on detection rate.

For performance evaluation of our prototype system, we applied Snort ruleset to SGS. And we also used Fig.10 test bed for experiments.

At first, we generate and transmit packet to the test bed using IXIA Traffic Generator. And we tried to attack by Nessus and IDS Informer. As background traffic generated by IXIA increase gradually, we observed the rate of alert generation. That is, we measured the decrease in effectiveness of the detection when the traffic rate increases. The ruleset used included 200 rules. Fig. 11(a) shows the results of this experiment. Increasing traffic rate hasn't an effect on detection rate of SGS.

(a) Alerts according to the Increasing Traffic



(b)Alerts according to the Increasing Rules

Fig. 11. The Results of Detection Rate

The second experiment was to run SGS with a constant traffic rate of 100Mbps and an increasing number of signatures. The experiment starts with only the 200 signatures that are needed to achieve maximum detection for the given attacks. Fig. 11(b) shows the results of this experiment. Also, increasing number of signatures hasn't an effect on detection rate of SGS. The previous two experiments using Snort sensors are performed by Kruegel et al.[12]. Compared with Snort sensor, our prototype

system showed a consistent performance in traffic level and had nothing to do with increasing number of signatures used.

V. CONCLUSION AND FUTURE WORK

One of important requirements of SPI-based intrusion detection system is high performance. Even though SPI technology in network security system is developed to reduce false positive alerts, if not satisfied with performance, it may not be used.

Due to the increasing link speed, the number of attack patterns, and signatures to be maintained, it is a challenging issue to provide a seamless protection for secure network service. In this paper, we presented the architecture of our system that performs the real-time traffic analysis and intrusion detection on high-speed links, and proposed the novel detection mechanisms in FPGA-based reconfiguring hardware that supports more efficient intrusion detection. We have implemented the prototype of our system for the analysis of the traffic carried by a Gigabit link. Most of all, our system focus on reducing a degradation of performance caused by high-speed traffic analysis to the minimum level. Therefore, it is run by the FPGA logic proposed for improvement in performance. Also, it has the advantage that is capable of supporting the effective response by using inline mode monitoring technique on four Gigabit links.

The performance of SPI-based intrusion detection system mainly depends on the performance of processing session table.. In this paper, we also proposed session state management scheme which can perform stateful packet inspection in real time by performing session table processing that allows more efficient generation of state information. And we designed and implemented SPI-based intrusion detection module in a FPGA to help alleviating a bottleneck in network intrusion detection systems.

However, the current prototype needs some improvement and a thorough evaluation to be deployed in a real-world environment. In order to resolve the problem derived from the verification of implemented system, it is necessary to upgrade system performance and availability, and to perform faults-tolerance test with prototype. Also, we need to keep up much effort for improvement in performance of detection mechanism on high-speed links. We hope to implement and expand our designed system and give more effort to demonstrate effectiveness of our system.

REFERENCES

[1] http://www.checkpoint.com, Firewall-1 Product
[2] Lance Spitzner, Understanding the FW-1 State Table, http://www.spitzner.net/fwtable.html
[3] Brian Caswell, Jay Beale, James C. Foster, Jeremy Faircloth, Snort 2.0 Intrusion Detection(Syngress Publishing, February 2003)
[4] http://www.snort.org, Snort Preprocessor Stream4
[5] Xin Li, Zheng-Zhou Ji, and Ming-Zeng Hu, "Stateful Inspection Firewall Session Table Processing", Proc. Of the International Conference on Information Technology: Coding and Computing(ITCC'05), Volume 2, April 2005, Pages:615-620
[6] Sergei et al., "SNORTRAN: An Optimizing Compiler for Snort Rules", Fidelis Security Systems, Inc., 2002
[7] Byoungkoo Kim, Youngjun Heo, and Jintae Oh, "High-Performance Intrusion Detection in FPGA-based Reconfiguring Hardware", in Proceeding of APNOMS, 2005
[8] Dong-Ho Kang, Byoung-Koo Kim, and Jin-Tae Oh, "Protocol Anomaly and Pattern Matching based Intrusion Detection System", WSEAS Transaction on Communications, Issue 10, Vol. 4, October 2005, pp.994-1001
[9] Slobodan Bojanic, Vladimir Milovanovic, Zorana Bankovic, Carlos Carerras, and Octavio Nieto-Taladriz, "Intrusion Detection Using New FPGA Architecture", WSEAS Transaction on Communications, Issue 10, Vol. 4, October 2005, pp.1077-1085
[10] M. Mehde, M. Bensebti, A. Anou, and M. Djebari, "Real Time Solution for Computer Network Intrusion Detection", WSEAS Transaction on Computers, Issue 1, Vol. 5, January 2006, pp.216-222
[11] Myung-Sup Kim, Young J. Won, and James Won-Ki Hong, "Application-Level Traffic Monitoring and an Analysis on IP Networks", ETRI Journal, Vol.27, No.1, February. 2005, pp.22-42.
[12] C. kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful Intrusion Detection for High-Speed Networks", in Proceedings of the IEEE Symposium on Research on Security and Privacy, Oakland, CA, IEEE Press, May 2002
[13] I. Charitakis, K. Anagnostakis, and E. Markatos, "An Active Traffic Splitter Architecture for Intrusion Detection", Proc. Of 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems(MASCOTS 2003), Orlando, October 2003, pp. 238-241
[14] Tarek Abbes, Alakesh Haloi, and Michael Rusinowitch, "High Performance Intrusion Detection using Traffic Classification", AISTA 2004 in Cooperation with the IEEE Computer Society Proceedings, Nov. 15-18,. 2004,
[15] Sarang Dharmapurikar, Praveen Krishnamurthy, T.S. Sproll and J.W. Lockwood, "Deep packet inspection using parallel bloom filters", IEEE Micro, Volume 24, Issue 1, Pages:52-61, Jan.-Feb.2004
[16] D.V. Schuehler, J. Moscola and J.W. Lockwood, "Architecture for a hardware-based, TCP/IP content-processing system", IEEE Micro, Volume 24, Issue 1, Pages:62-69, Jan.-Feb. 2004
[17] M. Aldwairi, T. Conte, and P. Franzon, "Configurable string matching hardware for speedup intrusion detection", In Workshop on Architecture Support for Security and Anti-virus(WASSA) Held in Cooperation with ASPLOS XI, Oct. 2004
[18] Z.K. Baker and V.K. Prasanna, "Time and area efficient pattern matching on FPGAs", In proceeding of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, pages 223-232, ACM Press, 2004
[19] C.R. Clark and D.E. Schimmel, "Efficient reconfigurable logic circuits for matching complex network intrusion detection patterns", In 13th International Conference on Field Programmable Logic and Applications, Sept. 2003
[20] R. Franklin, D. Carver, B.L. Hutchings, "Assisting Network Intrusion Detection with Reconfigurable Hardware", Proc. of the IEEE Symposium on FPGA's for Custom Computing Machine, April 2002
[21] Shaomeng Li et al. "Exploiting Stateful Inspection of Network Security in Reconfigurable Hardware", Proc. Of FPL2003, Lisbon, Portugal, September, 2003

[22] Hyogon Kim, Jin-ho Kim, Inhye Kang, and Saewoong Bahk, "Preventing Session Table Explosion in Packet Inspection Computers", IEEE Transaction on Computers, Vol. 54, No. 2, February 2005
[23] http://www.xilinx.com
[24] http://www.cypress.com
[25] http://www.model.com
[26] http://www.synplicity.com
[27] http://www.ixiacom.com
[28] http://www.bladesoftware.net
[29] http://www.spirentcom.com
[30] http://www.nessus.org

Seung-Yong Yoon received the B.S. and M.S. degrees in Computer Engineering from Chungnam National University in 1999 and 2001, respectively. Since 2001, he has stayed in Security Gateway System Team, Electronics and Telecommunications Research Institute(ETRI) of Korea to study Network Security related Topics.

Byoung-Koo Kim received the B.S. and M.S. degrees in Information and Communication Engineering from Sungkyunkwan University in 1999 and 2001, respectively. Since 2001, he has stayed in Security Gateway System Team, Electronics and Telecommunications Research Institute(ETRI) of Korea to study Network Security related Topics.

Jin-Tae Oh received the B.S and M.S degrees in Electronics Engineering from Kyungpook National University in 1990 and 1992, respectively. He worked at ETRI (Electronics and Telecommunications Research Institute) from 1992 to 1998. During 1998-1999, he stayed in MinMax Tech , USA, as a Research staff. He served as a Director in Engedi Networks, USA, during 1999-2001. He was both Co-founder and CTO Vice President in Winnow Tech. USA during 2001-2003. From 2003, he works with the Security Gateway Team, ETRI, Daejeon, Korea.

Jong-Soo Jang received the B.S and M.S degrees in Electronics Engineering from Kyungpook National University in 1984 and 1986, respectively. He received his Ph. D degree in Computer Engineering from Chungbuk National University in 2000. Since 1989, he has been working with ETRI, Daejeon, Korea and now is the Director of Applied Security Group. Since January 2008, he has been a Vice-President of KIISC(Korea Institute of Information Security and Cryptology)