Application of Contract Net Protocol to the Design and Simulation of Network Security Model

Kyong-jin Suh[a] and Tae-ho Cho[b]

[a] *School of Information & Communication Engineering, Sungkyunkwan University*
300 Cheoncheon-dong, Jangan-gu, Suwon, Gyeonggi-do 440-746, South Korea
Tel: +82-31-290-7221, Fax: +82-31-290-7211, E-mail:kjsuh@ece.skku.ac.kr

[b] *School of Information & Communication Engineering, Sungkyunkwan University*
300 Cheoncheon-dong, Jangan-gu, Suwon, Gyeonggi-do 440-746, South Korea
Tel: +82-31-290-7132, Fax: +82-31-290-7211, E-mail:taecho@ece.skku.ac.kr

Abstract

With the growing usage of the networks, the world-wide Internet has become the main means to exchange data and carry out transactions. It has also become the main means to attack hosts. To solve the security problems which occur in the network such as Internet, we import software products of network security elements like an IDS (Intrusion Detection System) and a firewall. In this paper, we have designed and constructed the General Simulation Environment of Network Security model composed of multiple IDSes and a firewall which coordinate by CNP (Contract Net Protocol) for the effective detection of the intrusion. The CNP, the methodology for efficient integration of computer systems on heterogeneous environment such as distributed systems, is essentially a collection of agents, which cooperate to resolve a problem. Command console in the CNP is a manager who controls the execution of agents or a contractee, who performs intrusion detection. In the Network Security model, each model of simulation environment is hierarchically designed by DEVS (Discrete EVent system Specification) formalism. The purpose of this simulation is to evaluate the characteristics and performance of CNP architecture with rete pattern matching algorithm and the application of rete pattern matching algorithm for the speeding up the inference cycle phases of the intrusion detection expert system.

Keywords:

IDS, Firewall, General Simulation Environment, Network Security model, CNP, DEVS formalism, rete pattern matching algorithm

Introduction

With the growing usage of the networks, the world-wide Internet has become the main means to exchange data and carry out transactions. It has also become the main means to attack hosts [1]. To solve the security problems which occur in the network such as Internet, we import some kinds of network security elements like an IDS and a firewall. In this paper, we have designed and constructed the General Simulation Environment of Network Security model composed of multiple IDSes and a firewall which coordinate by CNP for the effective detection of the intrusion. IDS monitors system activities to identify unauthorized use, misuse or abuse of computer and network system [2,3]. A firewall that plays a vital role in network security restricts access between the external and the internal network [4,5].

In the beginning of intrusion detection, host-based IDS was installed in each host to protect the internal network. The detection of host-based IDS caused each host to increase system loading as well as to abuse its resources and had a limit in detecting some intrusions from networks. To make up for the weak points in the host-based single IDS, the network-based multi IDS was imported. The network-based multi IDS applying the theory of distributed system reduces the system loading by distributing jobs to agents and the detection time by selecting the best agent to detect an intrusion. It is important that security models are coordinated in the multi IDS for the performance of detection. Most of all, the effective allocation of design agents to distributed and cooperative design tasks becomes a crucial issue to achieve high performance in agent-based cooperative design [6]. Agent, especially multi agent, as a new technology of Distributed Artificial Intelligence (DAI), has become a new hotspot of AI research, and a lot of important production has been made [7, 8]. Agent technology has been extensively applied to many fields, from simple personal E-mail filter to complex systems such as aerial traffic control system and flexible manufacturing system. The CNP, one of DAI, will be applied to the coordination of security system.

To allow effective coordination and control of design tasks, CNP allocates agents to design tasks [9, 10, 11]. In this

approach, agents coordinate their activities through contracts to accomplish a design goal. Contracting involves an exchange of information among agents, an evaluation of the information and a final agreement based on mutual selection. In other words, CNP selects the best agent to serve jobs by bidding and then the selected agent performs the jobs.

At first, expert system is applied to an IDS and the model of an improved IDS is designed by selecting an effective algorithm for its inference cycle. The CNP will be applied to the system environment which distributed IDSes and Firewall are coordinated in. The GSSE (General Security Simulation Environment) will be modeled and constructed by DEVS formalism that is well-formed theory and is proper to model a large scaled system [12].

Design of GSSE

To construct general security simulation environment, we should precede not only the modeling of network and security elements but also the design of target network for the simulation.

Design of Target Network

The design of target network can be a criterion to determine whether the result of simulation can influence real system or not. *Figure 1* is the structure of the target network that has three subnets. In the internal network, each subnet has some hosts with web server, mail server, database server, and file server and IDS is installed in four hosts per subnet. There are Router, Gateway, Firewall, and Command Console as a network component in the target network.

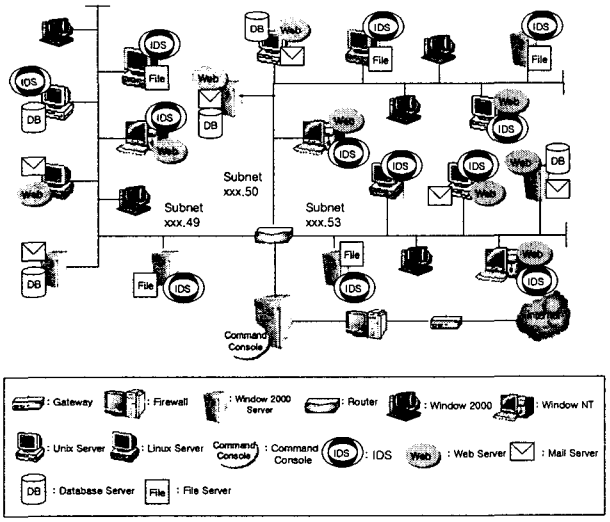

Figure 1 - Structure of Target Network

Design of Network Security Model

Figure 2 is the structure of Network Security model modeling the target network based on DEVS formalism for the simulation.

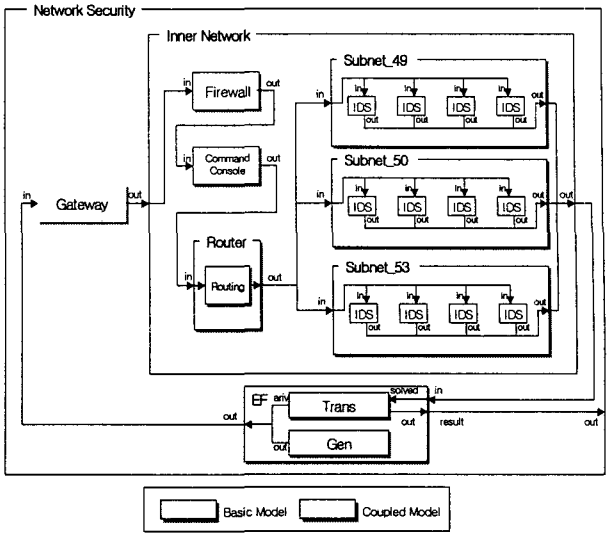

Figure 2 - Structure of Network Security Model

Network Security Model is widely divided into three models; EF model, Gateway model, Inner Network model. EF model is composed of Gen and Trans model. Gen model generates simulation packets based on packet data captured from the real network and sends them to Inner Network model. Trans model controls the simulation and analyzes the simulation results.

System Entity Structure of Target Network

Figure 3 represents a system entity structure for the overall Network Security model. Each model has a decomposition or a specialization relationship hierarchically.

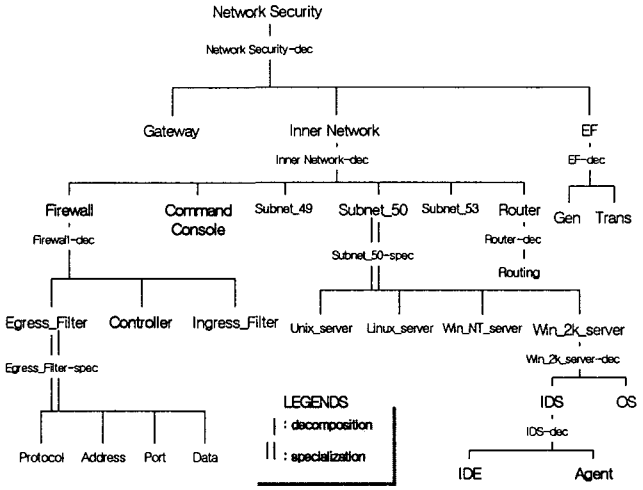

Figure 3 - SES of Target Network

DESIGN OF CNP

CNP

The CNP was originally proposed as a tool for communication and control in a distributed problem solver [9]. Its use was demonstrated in a distributed sensing system in and for a distributed delivery system in [13]. CNP

provides a mechanism for agents to communicate and negotiate to solve a distributed problem via contracts [9]. A contract is a set of tasks to be accomplished. Agents announce tasks that they need performed, make bids to perform tasks announced by other agents, evaluate the bids and award contracts [9]. CNP is essentially a collection of nodes, which cooperate to resolve a problem and a node can be a manager who controls a task's execution or a contractee, who performs it [10]. The application of CNP can improve the efficiency and accuracy of intrusion detection by the complement and coordination among multi IDSes.

Command Console Model

Command Console model, which controls all IDSes model and Firewall model in CNP, comprises three models; Messenger model, Selector model, Commander model. Messenger model manages the communication of messages and is composed of Receiver and Sender model. Receiver model receives messages from IDSes and Sender model sends the messages that made in Selector or Commander model to a certain IDS or many IDSes by means of unicast, multicast, or broadcast. Selector model selects an IDS that detects Inner Network with bids collected from all IDSes. Commander model determines a kind of message that regulate IDSes and Firewall according to the state of Inner Network.

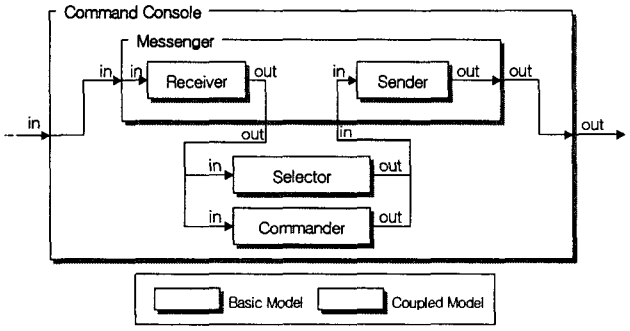

Figure 4 - Structure of Command Console model

IDS Model

Packet data from Command Console model is detected in IDE (Intrusion Detection Engine) and messages related to a control are processed in Agent model. Also agent model treats messages detected or processed in IDE model.

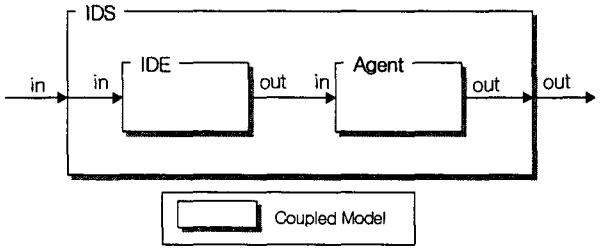

Figure 5 - Coupling Diagram of IDE and Agent

Figure 5 represents the structure of coupling between IDE model and Agent model.

IDE Model

Figure 6 shows the structure of IDE model. IDE model is divided into Detector model, Response_Generator model, and Logger model.

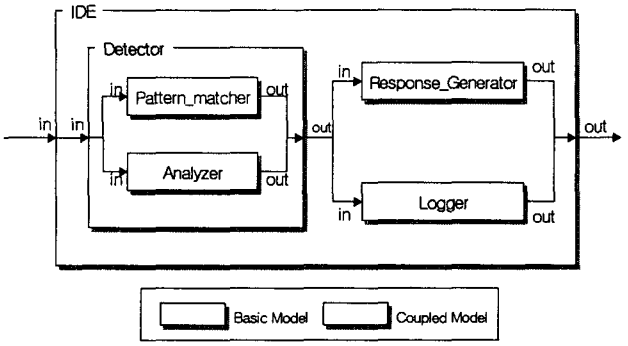

Figure 6 - Structure of IDE model

Detector model is composed of Pattern_matcher and Analyzer model. Pattern_matcher model is a rule-based expert system that detects intrusions through pattern matching procedure between packet data and rules. Most of all, the application of rete algorithm can reduce the processing time of pattern matching phase which takes the most time in inference cycle of expert system. Analyzer model is a statistical detection engine that detects intrusions by analyzing system log and audit. Response_Generator model determines a response according to the detection result of Detector model and sends a message. Logger model records all information of detection procedure in the log file.

Detector model detects an intrusion through state transition. *Figure 7* is the state transition diagram of Detector model. When packet data come into the input port, the state transition begins with Passive state that is an initial state and transits to Intrusion state that is a goal state via Vulnerable and Active attack state and then Detector model detects an intrusion. If Detector model fails to detect an intrusion or the loading of a system with IDS is greater than the predefined threshold value, state transits to Failed state and Detector model requests other IDS to detect an intrusion.

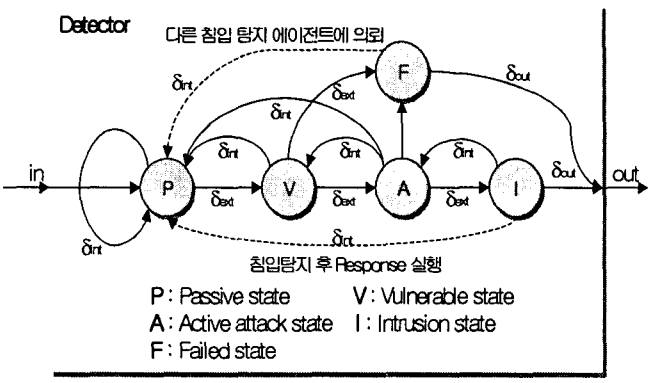

Figure 7 - State Transition Diagram of Detector model

The single event signature detects an intrusion with transiting to Intrusion state if a single event signature

happens in Passive state. The multi event signature detects an intrusion with transiting from Passive state via Vulnerable and Active attack state to Intrusion state by the predefined frequency of occurrence or penetration scenarios.

For example, the detection procedure of trinoo attack that is a kind of DDOS (Distributed Denial Of Service) is that state transits from Passive to Vulnerable if attacker sends gOrave, which is a default password, to master and transits from Vulnerable to Active attack if attacker sends betaalmostdone, which is a default startup password, to master and transits again from Active attack to Intrusion if attacker sends killme, which is a default mdie password, to master and if state is Intrusion state, Detector model detects trinoo attack and inform it to Response_Generator model and return to Passive state.

Figure 8 is the general structure of pattern_matcher model without the application of rete algorithm. Preprocessor model manipulates packet data to use in Rule model. Rule model has several sub-modules of rules that have a similar signature and actually detects an intrusion by pattern matching.

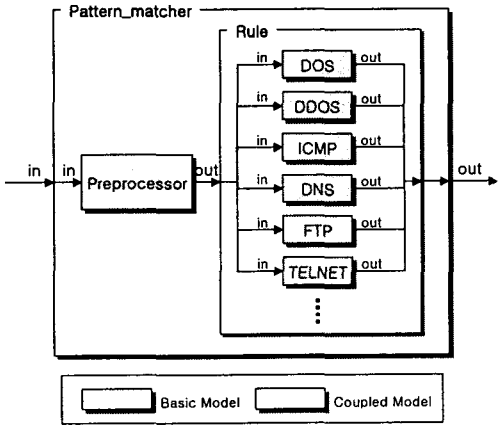

Figure 8 - General structure of Pattern_matcher model

Figure 9 is the structure of Pattern_matcher model applying rete algorithm. Likewise, Preprocessor model does the same function right above model. Rete_Network model constructs rete network with rules generated in Rule_base model and detects an intrusion through pattern matching procedure.

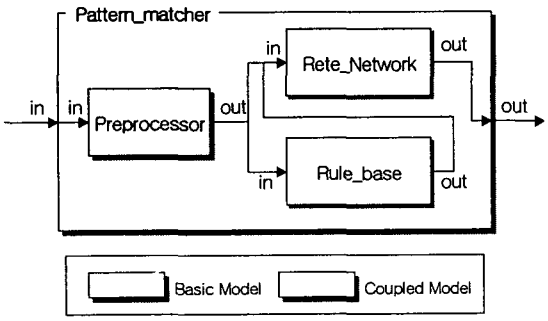

Figure 9 - Pattern_matcher model applying Rete algorithm

Agent Model

Figure 10 is the structure of Agent model that communicates with Command Console model in CNP. Agent model is composed of Messenger, Bidder, and Local_Optimizer model.

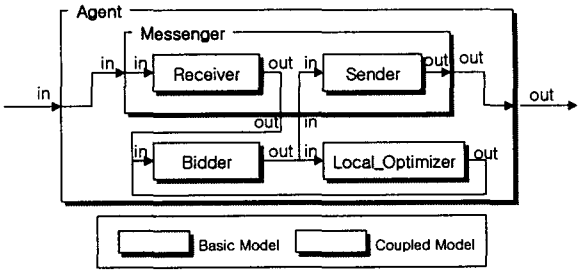

Figure 10 - Structure of Agent model

Messenger model has Receiver that receives messages from Command Console model and Sender that sends messages made in Bidder model. Bidder model makes bid with data from Local_Optimizer model and controls IDE model according to the command of Command Console model. Local_Optimizer model manages system information for bid and optimizes rules used in IDE model. it especially manages state information for expertise, experience, and loading. Expertise represents the number of rules that an IDS has and experience represents the number of times that an IDS has detected an intrusion before. If IDS detects an intrusion, experience is increased to one degree. Loading represents system loading of CPU and memory by means of three-level scheme; under-loaded, normal over-loaded.

Firewall Model

Firewall model basically carries out the filtering of ingress and egress packets. If IDS detects an intrusion, Firewall model blocks the packets related to an intrusion or perform the responses after mapping them to the policy.

This type of CNP structure can cause single point of failure because Command Console model controls and manages all agents. If Command Console becomes invalid, the function of whole system can be stopped. Therefore, Firewall model should filter off packets to access Command Console from internal and external network except messages from Agent model.

Figure 11 is the structure of Firewall model that has Controller, Ingress_Filter, and Egress_Filter model. Controller model controls Ingress_Filter and Egress_Filter model and manages all messages from Command Console model. Ingress_Filter model filters packets which comes from external networks to internal network through four model; Protocol, Address, Port, and Data. Protocol model filters protocol information extracted from packet data and Address model filters IP address and Port model filters port information. Data model carries out the filtering of data part. Egress_Filter model filters packets that comes from internal network to external networks and has the same sub-model as Ingress_Filter model.

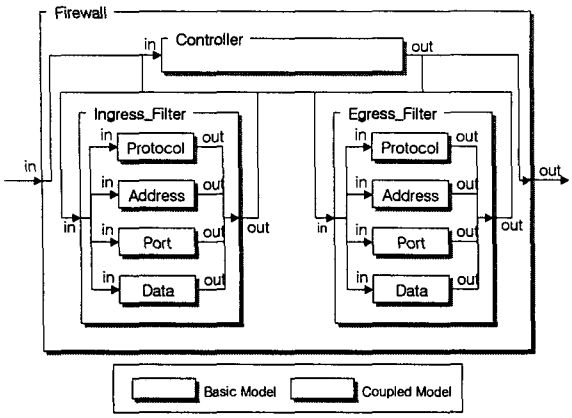

Figure 11 - Structure of Firewall model

Controller Model

Figure 12 shows the structure of Controller model inside Firewall model. Controller model is composed of Messenger and Commander model. Messenger model manages all messages which comes from external or internal network, Command Console, Ingress_Filter, and Egress_Filter. Commander model actually decides and controls all activities that Firewall model performs.

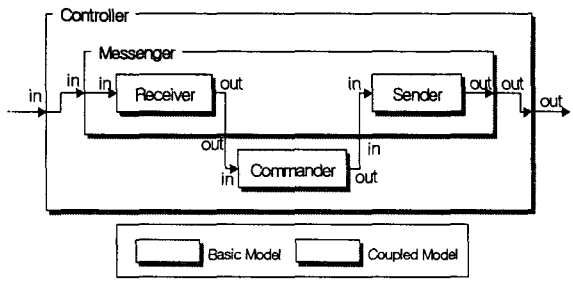

Figure 12 - Structure of Controller model

Coordination of CNP

Coordination Cycle of CNP

If simulation starts up and packets come into the internal network, Command Console sends bid message to all IDS agent. After receiving bid message, each agent makes bid and sends bid_data message. Command Console selects IDS agent to detect an intrusion by selection algorithm with bid_data and sends award message to the selected IDS agent. After receiving award message, the selected agent prepares for the detection of intrusion and waits for packet_data message. Command Console copies packet data to packet_data message and sends it. The selected IDS agent detects an intrusion from packet_data.

If the state of Detector model becomes Failed state, agent sends announcement message to Command Console and after receiving it Command Console repeats the coordination cycle.

If Detector model detects an intrusion, agent sends intrusion message to Command Console and sends directly intrusion_data message. Command Console receiving them sends intrusion and intrusion_data message to Firewall in turns and sends broadcast and broadcast_data message, which includes intrusion data, to all agent in turns. Command Console and IDS agent repeat this cycle. *Figure 13* sequentially represents the coordination cycle among IDS, Command Console, and Firewall model.

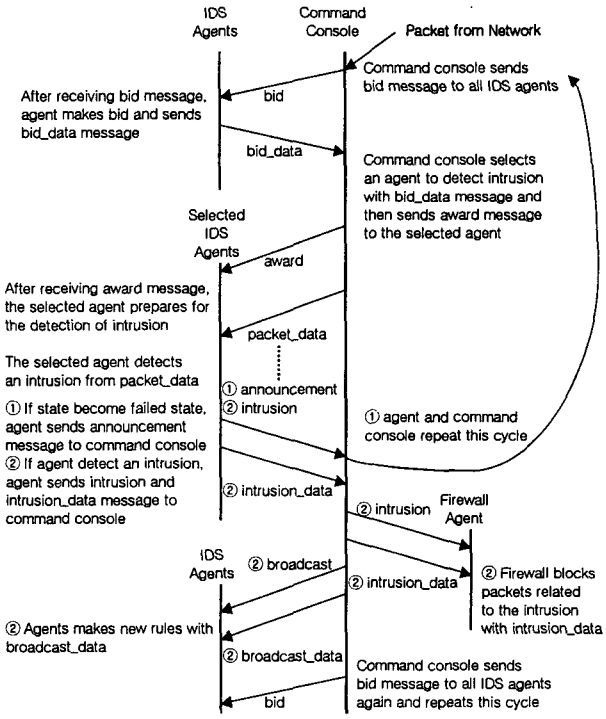

Figure 13 - Coordination Cycle of CNP

Figure 14 shows the coordination structure of CNP and the flow of messages. Command Console is a centralized entity that controls the coordination between IDSes and Firewall in CNP. The coordination is actually performed by exchanging messages among Command Console, Agent model of IDS, and Controller model of Firewall.

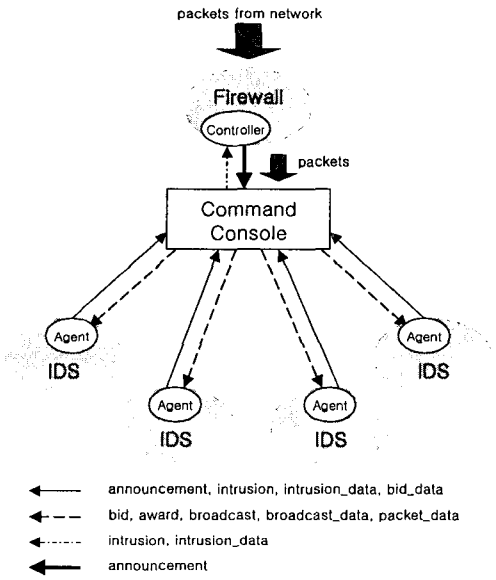

Figure 14 - Coordination Structure of CNP

Message

Command Console, IDS, and Firewall model have Messenger model that plays a role to exchange messages. Message is largely divided into control message and data message and is distinguished by msg_type field. *Table 1* represents the structure and types of messages.

Table 1 - Structure and Type of Message

msg_type	msg_content

	msg_type	메시지 종류
Control Message	0	Broadcast
	1	Announcement
	2	Bid
	3	Award
	4	Intrusion
Data Message	5	broadcast_data
	6	bid_data
	7	packet_data
	8	intrusion_data

Figure 15 shows the structure of msg_content. msg_content field is composed of agent_name field used to search an address of agent and data field including actual information. Because Command Console has a mapping table that maps agent_name to the address of agent, it can receive or send messages to all agents.

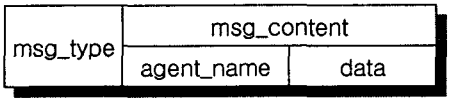

Figure 15 - Structure of msg_content

Control Message

Control message is used for Command Console to control IDSes and Firewall. Bid message announces that all IDS agents should send bid to Command Console. Award message informs that an agent is selected as the IDS agent for intrusion detection. Intrusion message informs Command Console that an intrusion is detected and also makes Firewall response to the intrusion. Announcement message informs Command Console that all agents repeat coordination cycle when state transits to Failed state. Broadcast message informs all IDS agents that Command Console broadcasts detection information.

Data Message

There are four types of data messages; bid_data, intrusion_data, packet_data, and broadcast_data message. Bid_data message has information requested for agent selection. Packet_data message contains packet data extracted from the real packets and broadcast_data message includes the detection information.

The data field of bid_data message used for agent selection during bidding procedure is composed of expertise, experience, and loading like Figure 16. Expertise represents rules that an IDS has by means of numerical value and experience represents the number of times that an IDS has detected an intrusion before by means of numerical value. Loading represents system loading of CPU and memory by means of three-level scheme. This three value is managed in Local_Optimizer model.

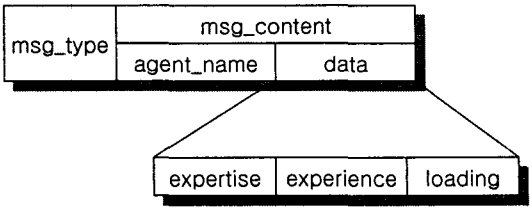

Figure 16 - Structure of bid_data

IDS Agent Selection Algorithm

Each agent sends bid_data message to Command Console if the value of loading field is not greater than the threshold value of system loading. Selector inside Command Console makes a list of bids and puts bids from agents into the list. At first, Selector sorts it by expertise in descending order and selects an agent that has the greatest value of expertise. If the number of bid including the greatest value of expertise is more than 2, Selector deletes bids from the list except bids including the greatest value of expertise and sorts the list by experience in descending order. Also Selector chooses an agent that has the greatest value of experience. If the number of bid including the greatest value of experience is more than 2, Selector deletes bids from the list except bids including the greatest value of experience and sorts the list by loading in ascending order. Finally Selector selects the best agent, the first element of list, and the selected agent detects an intrusion. *Figure 17* is the selection algorithm that represents the routines of selection.

```
Let bidi be bids
Set bid_list = empty set
Let bid_list = ( bid1, bid2,..., bid n ) be a list of bids
for i = 1 to n
   if loading of bidi >= threshold value
      Delete bidi from bid_list
Sort bid_list by expertise in descending order
if the number of bid including the greatest value of expertise >= 2 then
{
   Delete bids from bid_list except bids including the greatest value of expertise
   Sort bid_list including bids of the same expertise by experience in descending order
   if the number of bid including the greatest value of experience >= 2 then
   {
      Delete bids from bid_list except bids including the greatest value of experience
      Sort bid_list including bids of the same experience by loading in ascending order
   }
}
Select Agent from bid_list(the first element)
```

Figure 17 - Selection Algorithm

Application of rete algorithm

Intrusion Detection Expert System (IDES)

Forward-chaining systems are composed of a set of inference rules that contain the knowledge of the expert system, and the working memory (WM), containing the current state of the system in the form of assertions called working memory elements (WMEs) [14]. The left-hand side (LHS) of each rule consists of a conjunction of condition elements (patterns) that corresponds to the if-part of the rule. The right-hand side (RHS) of each rule is a set of actions, that corresponds to the then-part of the rule. When the expert system interpreter determines the set of rules whose LHSs satisfy the WMEs currently present in the WM, it selects one of them and fires it. Firing a rule results in executing the set of actions in the rule's RHS, thus generating some change in the WM. The interpreter executes the expert system in cycles. Each of these recognize-act cycles is composed of pattern-matching, conflict resolution, and act phases. In the pattern-matching phase, the LHSs of all rules are matched against the WMEs currently present in the WM. The result of this phase is the conflict set, Containing instantiations of all satisfied rules. A rule instantiation is an ordered list of WMEs that satisfies the LHS of the rule. During the conflict resolution phase, one of the rule instantiations is selected from the conflict set according to some predefined conflict resolution strategy. In the act phase, the actions in the selected rule's RHS are executed, changing some contents of the WM (adding new WMEs, deleting or modifying existing WMEs). At the end of this phase, pattern-matching is executed again. Traditionally, The cycling halts when the conflict set generated by the pattern-matching phase is empty.

Figure 18 is some examples of detection rules which an IDS uses in inference cycle. These rules are formed of several modules like DOS, DDOS etc and each module detects the similar types of attacks.

```
DOS Rules :

R1 :
if protocol == ip ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET ∧
fragbits == "M" ∧ dsize == 408 ∧ state == "passive"
then state = "vulnerable", p_count+=1

R2 :
if protocol == ip ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET ∧
fragbits == "M" ∧ dsize == 408 ∧ state == "vulnerable"
then p_count+=1

R3 :
if protocol == ip ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET ∧
fragbits == "M" ∧ dsize == 408 ∧ state == "vulnerable" ∧ p_count >= threshold
then state = "intrusion", msg("DOS Jolt attack")
                    ⋮                              ⋮

ICMP Rules :

                    ⋮                              ⋮
R15 :
if protocol == icmp ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET
∧ data == "|5768 6174 7355 7020 2d20 4210 4e65 7477|" ∧ itype == 8
then state = "intrusion", msg("ICMP PING WhatsupGold Windows)

R16 :
if protocol == icmp ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET
∧ dsize > 800
then state = "intrusion", msg("ICMP Large ICMP Packet")
                    ⋮                              ⋮
```

```
DDOS Rules :

                    ⋮                              ⋮
R11 :
if protocol == udp ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET
∧ DstPort == 31335 ∧ data == "*HELLO*" ∧ state == "passive"
then state = "vulnerable", msg("DDOS
Trin00/:DaemontoMaster(*HELLO*detected)")

R12 :
if protocol == udp ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET
∧ DstPort == 31335 ∧ data == "PONG" ∧ state == "vulnerable"
then state = "intrusion", msg("DDOS Trin00/:DaemontoMaster(PONGdetected)")

R13 :
if protocol == tcp ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET
∧ DstPort == 27665 ∧ TCPACKFlag == true ∧ data == "gOrave" ∧ state ==
"passive"
then state = "vulnerable", msg("DDOS Trin00/:Attacker to Master default
password")

R14 :
if protocol == tcp ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET
∧ DstPort == 27665 ∧ TCPACKFlag == true ∧ data == "betaalmostdone" ∧
state == "vulnerable"
then state = "active attack", msg("DDOS Trin00/:Attacker to Master default
startup password")

R15 :
if protocol == tcp ∧ IPSrcAddr == EXTERNAL_NET ∧ IPDstAddr == HOME_NET
∧ DstPort == 27665 ∧ TCPACKFlag == true ∧ data == "killme" ∧ state ==
"active attack"
then state = "intrusion", msg("DDOS Trin00/:Attacker to Master default mdie
password")
                    ⋮                              ⋮

Goal Rule :
if state = "intrusion" then passivate(), p_count = 0
```

Figure 18 - Detection Rules of Detector model

As you see, each rule has if-part corresponding to patterns and then-part corresponding to actions. If Detector model detects an intrusion and state transits to Intrusion state, it fires Goal Rule and returns initial state. Above rules, threshold means detection threshold value that makes state transit. It is possible to set up the effective detection threshold value if we perform the simulation with changing detection threshold value

Speeding up the IDES with Rete Algorithm

The rete algorithm is intended to improve the speed of forward-chained rule systems by limiting the effort required to recomputed the conflict set after a rule is fired [14]. The application of rete algorithm improves efficiency of intrusion detection by reducing the time of pattern matching which takes the most time in inference cycle.

Rete Network

the rete algorithm uses a special kind of data-flow network to perform pattern matching [15]. The rete algorithm is implemented by building a network of nodes, each of which represents one or more tests found on a rule LHS. Facts that are being added to or removed from the knowledge base are processed by this network of nodes. At the bottom of the network are nodes representing individual rules. When a set of facts filters all the way down to the bottom of the network, it has passed all the tests on the LHS of a particular rule and this set becomes an activation. The associated rule may have its RHS executed (fired) if the activation is not invalidated first by the removal of one or more facts from its activation set.

Within the network itself there are broadly two kinds of nodes; on-input and two-input nodes. One-input nodes perform tests on individual facts, while two-input nodes perform tests across facts and perform the grouping function. Subtypes of these two classes of node are also used and there are also auxiliary types such as the terminal nodes.

To run the network, new facts enter to each node at the top of the network as they added to the knowledge base. Each node takes input from the top and sends its output downwards. A single input node generally receives a fact from above, applies a test to it, and, if the test passes, sends the fact downward to the next node. If the test fails, the one-input nodes simply do nothing. The two-input nodes have to integrate facts from their left and right inputs, and in support of this, their behavior must be more complex. First, note that any facts that reach the top of a two-input node could potentially contribute to an activation; they pass all tests that can be applied to single facts. The two-input nodes therefore must remember all facts that are presented to them, and attempt to group facts arriving on their left inputs with facts arriving on their right inputs to make up complete activation sets. A two-input node therefore has a left memory and a right memory. A convenient distinction is to divide the network into two logical components; the single-input nodes comprise the pattern network, while the two-input nodes make up the join network.

At this point, the following figures are the rete networks constructed with several rules of Detector model by rete algorithm.

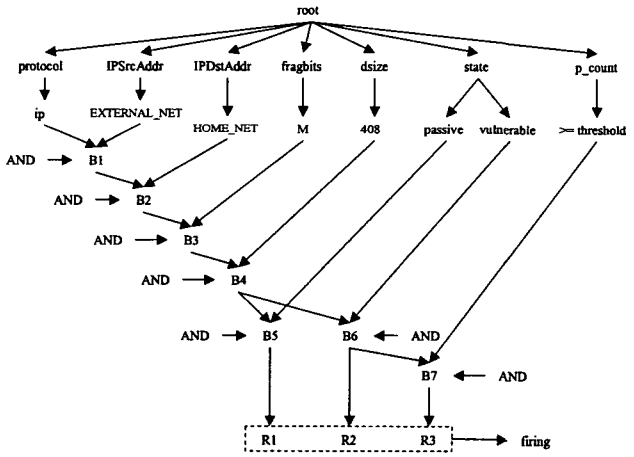

Figure 19 - Example #1 of Rete Network

Figure 19 is the rete network constructed with detection rules of DOS Jolt attack. If packets enter the root node, patterns move to nodes downward. If patterns reach a terminal node, the rule of the terminal node is fired.

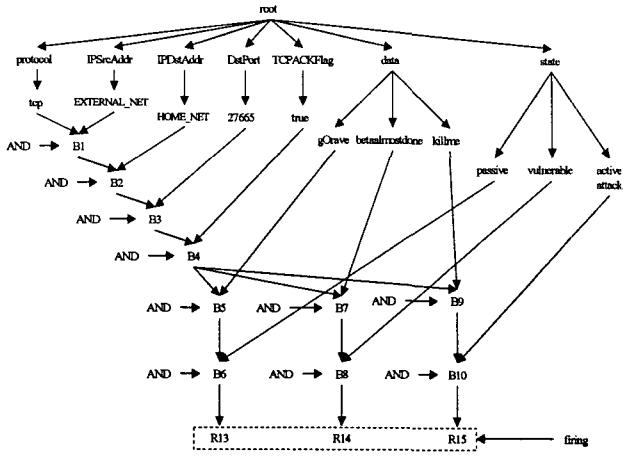

Figure 20 - Example #2 of Rete Network

Figure 20 is the rete network constructed with detection rules of DDOS Trinoo attack.

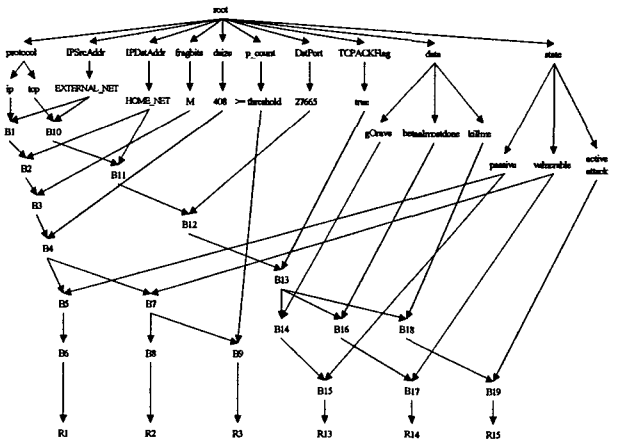

Figure 21 - Construction of Rete Network (#1 + #2)

Figure 21 is the rete network that integrates the rules of example #1 and example #2. A rete network is constructed with the modular rules like this way and Detector model detects an intrusion through the path from root node to terminal node in the rete network.

Simulation

Setting up Indicators for the Simulation

To evaluate the result of simulation, we set up intrusion detection time, false positive error ratio, and false negative error ratio as indicators. Intrusion detection time presents the time that is elapsed to detect an intrusion. False positive error ratio is the ratio that regards no intrusion as an intrusion and false negative error ratio is the ratio that regards an intrusion as no intrusion.

Simulation Result and Analysis

We perform the simulation about single IDS, multi IDS with CNP, and multi IDS with CNP & Rete algorithm.

Figure 22 presents the change of intrusion detection time as the detection threshold value of DOS Jolt attack is increased. According to the result of simulation, the detection of multi IDS with CNP is faster than that of single IDS and the detection time of multi IDS with CNP & Rete is improved much.

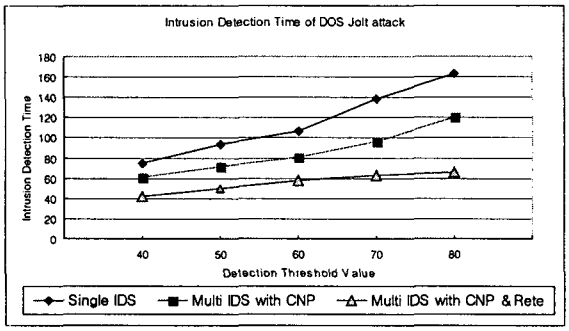

Figure 22 - Intrusion Detection Time of DOS Jolt attack

Figure 23 presents the change of false positive error ratio as the detection threshold value of DOS Jolt attack is increased. Multi IDS with CNP is superior to single IDS and the application of rete algorithm to multi IDS with CNP reduces the false positive error ratio. As the detection threshold value increases, false positive error ratio gets lower.

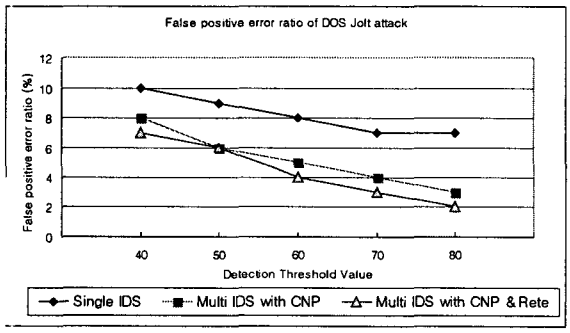

Figure 23 - False positive error ratio of DOS Jolt attack

Figure 24 presents the change of false negative error ratio as the detection threshold value of DOS Jolt attack is increased. False negative error ratio of multi IDS with CNP is lower than that of single IDS and the application of rete algorithm to multi IDS with CNP improves the efficiency. As the detection threshold value increases, false negative error ratio gets higher.

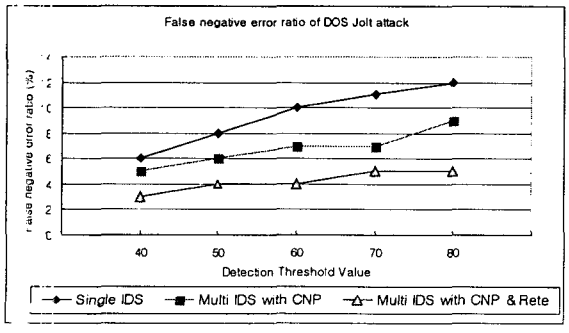

Figure 24 - False negative error ratio of DOS Jolt attack

In Pattern_matcher model, the number of modular rules is related to the time of pattern matching and the error ratio.

Figure 25 presents the intrusion detection time of DDOS Trinoo attack as the number of detection rules is increased. the detection of multi IDS with CNP is faster than that of single IDS. the ability of multi IDS with CNP & Rete is improved much and it has the nearly fixed detection time

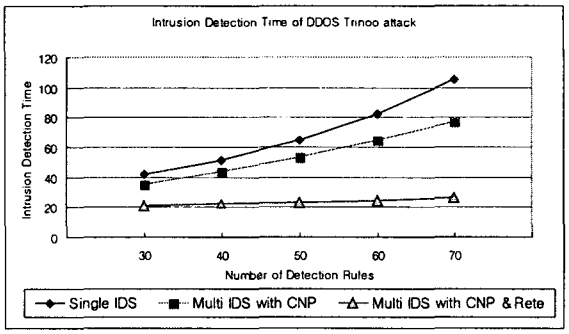

Figure 25 - Intrusion Detection Time of DDOS Trinoo attack

Figure 26 presents false positive error ratio of DDOS Trinoo attack as the number of detection rules is increased. Multi IDS with CNP & Rete has the lowest false positive error ratio and is hardly influenced by the number of detection rules.

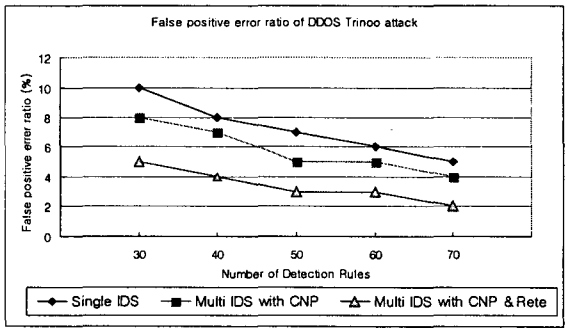

Figure 26 - False positive error ratio of DDOS Trinoo attack

Figure 27 presents false negative error ratio of DDOS Trinoo attack as the number of detection rules is increased. Multi IDS with CNP is superior to single IDS and the application of rete algorithm to multi IDS with CNP improves the efficiency of intrusion detection extremely.

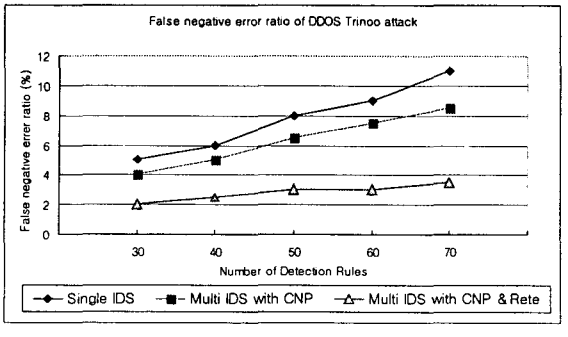

Figure 27 - False negative error ratio of DDOS Trinoo attack

Conclusion

The usage of the network is going to be increased in the future and also the value of information exchanged in the networks is going to get more important. On the other hand, intrusions occur more frequently and become more widespread and sophisticated. In this situation, it is necessary that IDS is imported as the basic component of the network and the utilization of IDS is regarded as a common behavior. We have set up several indicators for this simulation and have constructed the DEVS-based network security simulation environment and have applied the CNP for coordination between IDSes and Firewall. Also we have evaluated the performance of IDES applying rete pattern matching algorithm. Through the simulation, multi IDS coordinated by CNP performs more effectively in the detection of the intrusions than single IDS and can protect the network in advance by cooperating with Firewall. The application of rete algorithm improves the efficiency of multi IDS with CNP completely.

In the future, it will be necessary to construct the General Network Security Simulation Environment that is able to perform various security simulations and to develop the effective algorithms to be applied to inference cycle of the expert system that detects various intrusions.

Acknowledgments

This research is supported by IITA (Institute of Information Technology Assessment).

References

[1] Shan Zheng, Chen Peng, Xu Ying, Xu Ke, "nework state based intrusion detection model," *Computer Networks and Mobile Computing, 2001. Proceedings. 2001 International Conference*, pp. 481-486, 16-19 Oct. 2001.

[2] R. Base, *Intrusion Detection*, Macmillan Technical Publishing, 2000.

[3] E. Amoroso, *Intrusion Detection - An Introduction to Internet Surveillance, Correlation, Traps, Trace Back, and Response*, Intrusion.Net Books, 1999.

[4] Duan Haixin, Wu Jianping, Li Xing, "Policy based access control framework for large networks," *Proceedings. IEEE International Conference on ICON 2000*, Sept. 2000.

[5] Noureldien A. Noureldien, Izzeldin M. Osman, "On Firewalls Evaluation Criteria," *Proceeding of TENCON 2000*, pp. 104-110, Sept. 2000.

[6] K. M. Sim, S. K. Shiu, and B. L. Martin, "Simulation of a Multi-agent Protocol for Task Allocation in Cooperative Design," *IEEE SMC '99 Conference Proceedings. International Conference*, Vol. 3, pp. 95-100, 1999.

[7] Shungeng Hu, Li Zhang and Yixin Zhong, "Theories, Technology and Application of Multi-Agent Systems," *Computer Science*, Vol. 26, No.9, pp. 20-24, 1999.

[8] T. Sandholm, "An Implementation of the Contract Net Protocol based on Marginal Cost Calculations," *in 11th National Conference on Artificial Intelligence (AAAI-93)*, Washington. DC, 1993.

[9] Jihoon Yang, Raghu Havaldar, Vasant Honavar, Les Miller and Johny Wong, "Coordination of Distrbuted Knowledge Networks Using Contract Net Protocol," *Information Technology Conference*, IEEE, pp. 71-74, 1998.

[10]R. Smith, "The Contract Net Protocol: High-level Communication and Control in a distributed problem solver," *IEEE Transactions on Computers*, Vol. C-29, No. 12, pp. 1104-1113, December. 1980.

[11]H. van Dyke Parunak, "Manufacturing Experience with the Contract Net," *Research Notes in Artificial Intelligence: Distributed Artificial Intelligence*, Vol. 1, pp. 285 - 310, Morgan Kaufmann Publishers, 1987.

[12]H.S. Seo and T.H. Cho, "An application of blackboard architecture for the coordination among the security systems," *Simulation Modeling Practice and Theory, Elsevier Science B.V.*, Vol. 11, issues 3-4, pp. 269-284, Jul. 2003.

[13]T. Sandholm "An implementation of the contract net protocol based on marginal cost calculations," *11th National Conference on Artificial Intelligence (AAAI-96)*, Washington. DC, 1993.

[14]V. Devedzic, D. Velasevic, "An architecture for real-time inference engines on personal computers," *System Sciences 1992. Proceedings of the Twenty-Fifth Hawaii International Conference*, Vol. 1, pp. 619-630, 7-10 Jan. 1992.

[15]C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, Vol. 19, pp. 17-37, 1982.