Modeling and Simulation of Policy-based Network Security

Won-young Lee[a] and Tae-ho Cho[b]

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

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

Abstract

Today's network consists of a large number of routers and servers running a variety of applications. Policy-based network provides a means by which the management process can be simplified and largely automated. In this paper we build a foundation of policy-based network modeling and simulation environment. The procedure and structure for the induction of policy rules from vulnerabilities stored in SVDB (Simulation based Vulnerability Data Base) are developed. The structure also transforms the policy rules into PCIM (Policy Core Information Model). The effect on a particular policy can be tested and analyzed through the simulation with the PCIMs and SVDB.

Keywords:

Security Policy; PBNM (Policy-based Network Management); network security; DEVS formalism; simulation; Data Mining

Introduction

Present-day networks are large complex systems consisting of a variety of elements, which can come from a variety of vendors. But it is more difficult than before for human administrators to manage more and more new network devices. The challenges facing the enterprise managers include network congestion, network complexity and security. A new wave of multimedia applications now begins to enter into corporate Intranets –voice and video can hardly wait to join the bandwidth fray. Every network is made up of a variety of elements, but they must still work together. Because of this heterogeneity and the lack of complete standardization, managing a network with more than a handful of elements can require a significant amount of expertise. The network manger is faced with the difficult task of meeting internal and external security requirements while still providing easy and timely access to network resources to authorized user. The solution to these issues lies in policy-based management [1]. A network manager creates policies to define how resource or services in the network can (or cannot) be used. The policy-based management system transforms these policies into configuration changes and applies those changes to the network. The simplification and automation of the network management process is one of the key applications of the policy framework [2].

Since evaluating the performance of a security system directly in real world requires heavy costs and efforts, an effective alternative solutions is using the simulation model [3]. The simulation models are constructed based on the DEVS formalism and using these simulation models the security models are constructed. We construct the SVDB for analyzing the vulnerabilities. SVDB has vulnerabilities of systems and policy information that can be used by security systems through SVDB interface for the induction of policy rules. We have simulated to verify the performance of the rule induction using SVDB for DOS attack and Probing attack.

The rest of the paper is organized as follows. Section 2 briefly represents the major theories and systems related to this research, section 3 describes the models for the simulation. Section 4 shows the coordination between the policy-based framework and SVDB, section 5 represents the execution of simulation and results. Finally, section 6 contains the conclusion.

Background

This section briefly describes the background related to the current research. Section 2.1 represents the DEVS formalism based on which the simulation models are defined. Section 2.2 shows the policy-based framework, section 2.3 represents the policy representation. Section 2.4 shows the Vulnerability Database

DEVS formalism

The DEVS formalism [4,5] is a theoretically well grounded means of expressing hierarchical, modular discrete-event models. In DEVS, a system has a time base, inputs, states, outputs and functions. The system function determines next states and outputs based on the current states and input. In the formalism, a basic model is defined by the structure:

$$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$$

where X is an external input set, S is a sequential state set, Y is an external output set, δ_{int} is an internal transition function, δ_{ext} is an external transition function, λ is an output function and t_a is a time advance function. A coupled model is defined by the structure:

$$DN = < D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, select >$$

where D is a set of component name, M_i is a component basic model, I_i is a set of influences of I, $Z_{i,j}$ is an output translation, select is a tie-breaking function. Such a coupled model can itself be employed in a larger coupled model. Several basic models can be coupled to build a more complex model, called a coupled model.

Policy-based Framework

The general policy-based administration framework we present can be considered an adaptation of the IETF policy framework to apply to the area of network provisioning and configuration [6-9]. The policy architecture as defined in the IETF consists of four basic elements (as shown in Fig. 1).

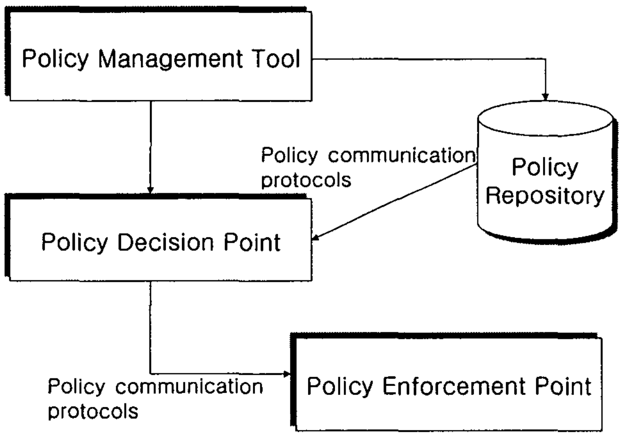

Figure 1 - The IETF policy framework

PMT (Policy Management Tool): The PMT is used by an administrator to input the different policies that are active in the network. The PMT takes as input the high-level policies that a user or administrator enters in the network and converts them to a much more detailed and precise low-level policy description that can apply to the various devices in the network.

PDP (Policy Decision Point): The PDP is a process that makes decisions based on policy rules and the state of the services those policies manage. The PDP is responsible for policy rule interpretation and initiating deployment. Its responsibilities may include trigger detection and handling, rule location and applicability analysis, network and resource-specific rule validation and device adaptation functions. In certain cases, it transforms and/or passes the policy rules and data into a form and syntax that the PEP (Policy Enforcement Points) can accept, leaving the implementation of the policy rules to the PEP.

PEP: The PEP are the network devices that actually implement the decisions that the policy decision points pass to them. The PEP are also responsible for monitoring any statstics or other information relevant to its operation and for reporting it to the appropriate places.

Policy Repository: The Policy Repository is used to store the policies generated by the management tool. Either a directory or a database can store the rules and policies required by the system. In order to ensure interoperability across products from different vendors, information stored in the repository must correspond to an information model specified by the Policy Framework Working Group.

Policy Communication Protocols: Different protocols are to be used for various parts of the architecture (e.g., COPS or SNMP can be used for PDP-PEP communications). A repository could be a network directory server accessed using LDAP.

Policy Representation

The high-level and low-level policies required for network management can be specified in many different ways [2].

From a human input standpoint, the best way to specify a high-level policy would be in terms of a natural-language input. Although these policies are very easy to specify, the current state of natural-language processing needs to improve significantly before such policies can be expressed in this manner. The next approach is to specify policies in a special language that can be processed and interpreted by a computer. When policies are specified as a computer interpretable program, it is possible to execute them. However, in general it is quite difficult to determine if the policies specified by two different programs are mutually consistent.

A simpler approach is to interpret the policy as a sequence of rules, in which each rule is in the form of a simple condition-action pair (in an if-then-else format). The rules are evaluated on specific triggers, such as the passage of time or the arrival of a new packet within the network. The IETF has chosen a rule-based policy representation in its specification. IETF Policy Framework WG works especially on the "condition action" part to define Policy Core Information Model (PCIM)[10] for the representation of policy information. The PCIM is the object-oriented information model for representing policy information. This model defines representing policy information and control of policies, and association classes that indicate how instances of the structural classes are related to each other. Policies can either be used in a stand-alone fashion or aggregated into policy groups to perform more elaborate functions. Stand-alone policies are called policy rules. Policy groups are aggregation of policy rules, or aggregations of policy groups, but not both. Fig. 2 illustrates the inheritance hierarchy for PCIMe (PCIM extensions) [11].

An alternative specification of policies is to represent them simply as entries in a table. The table consists of multiple attributes. Some of these attributes constitute the condition part, and others constitute the action part. Such a tabular representation is rich enough to express most of the policies that can be specified with a rule-based notation. Furthermore, it is easier to analyze for dominance and consistency.

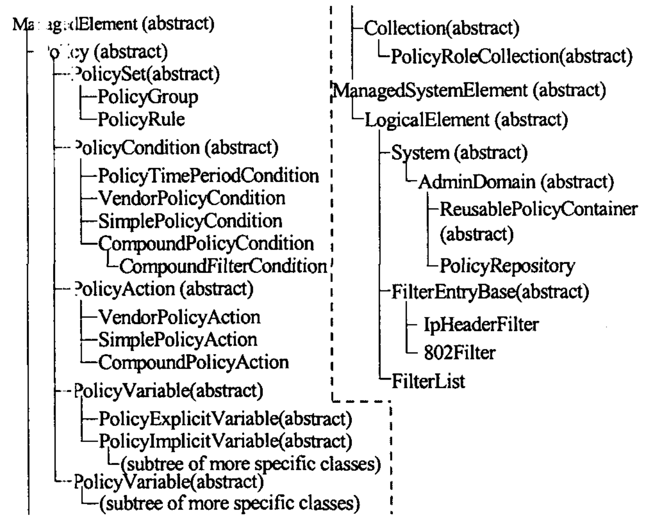

Figure 2 - Class Inheritance Hierarchy for PCIME

Vulnerability Database

A vulnerability is a condition or weakness in (or absence of) security procedures, technical controls, physical controls, or other controls that could be exploited by a threat [12].

The theme of vulnerabilities analysis is to devise a classification, or set of classifications, that enable the analyst to abstract the information desired from a set of vulnerabilities. This information may be a set of signatures, for intrusion detection; a set of environment conditions necessary for an attacker to exploit the vulnerability; a set of coding characteristics to aid in the scanning of code; or other data [13].

Government and academic philanthropists, and some companies, offer several widely used and highly valued announcement, alert, and advisory services for free. Each of these organizations referred to the same vulnerability by a different name. Such confusion made it hard to understand what vulnerabilities you faced and which ones each tool was looking for- or not looking for. The MITRE Corporation began designing a method to sort through the confusion. It involved creating a reference list of unique vulnerability and exposure names and mapping them to appropriate items in each tool and database. We use CVE names in a way that lets a security system crosslink its information with other security systems [14].

The Structure of Target Network and Simulation Model

Fig. 3 shows target network architecture.

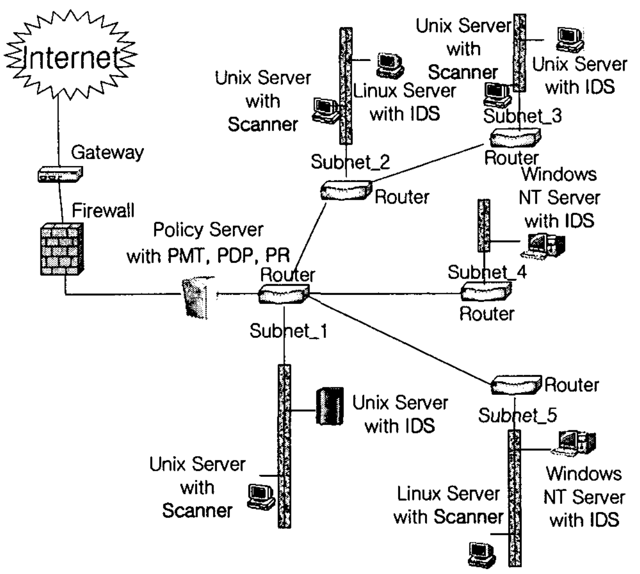

Figure 3 - Structure of target network

Fig. 3 shows the structure of the target network that has five subnets; subnet_1, subnet_2, subnet_3, subnet4 and subnet_5. The types of component models in the network are Policy Server, IDS, Firewall, Router, Gateway, Vulnerability Scanner model. Each subnet has unix server, linux server, windows NT server and etc. These models are constructed based on the DEVS formalism.

Network architecture

The System Entity Structure (SES)[4] is a knowledge representation scheme that combines the decomposition, taxonomic, and coupling relationships. The entities of the SES refer to conceptual components of reality for which models may reside in the model base. Also associated with entities are slots for attribute knowledge representation. An entity may have several aspects, each denoting a representation. An entity may also have several specializations, each representing a classification of the possible variants of the entity.

Fig. 4 is a system entity structure for the overall network model of Fig. 3. Network_Simulation model is decomposed into Network and EF. EF model is decomposed into Generator model and Transducer model. Generator model generates network input packets. Transducer model measures a performance indexes of interest for the target system.

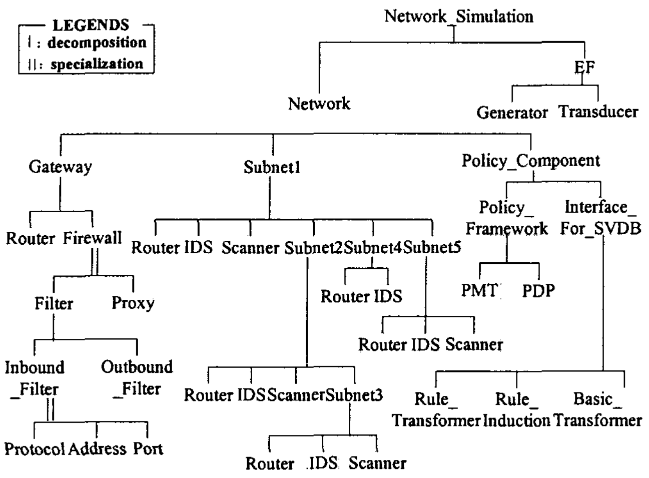

Figure 4 - SES of the target network

SVDB

SVDB has the specific information that can be used by security agents as well as The common information of vulnerability of system. SVDB has four tables; vulnerability information [15], packet information, system information and references information.

Table 1 - Tables of SVDB

Table	Field
Vulnerability Information	Vulnerability Name(CVE), Summary, Published, Vulnerability Type, Exploitable Range, Loss Type, Vulnerable Software and Versions
Packet Information	IP flags, TTL, Protocol, Source IP, Destination IP, IP options, ICMP code, ICMP type, Source port, Destination port, Sequence number, Acknowledgement number, TCP flag, Offset, Payload size, URL contents, Contents, CVE Name
System Information	Vulnerable Software and Versions Vendor, Name, Version
References Information	Source, Type, Name, Link

SVDB also has particular parts for accuracy and efficiency of security agents. The payload size is used to test the packet payload size. It may be set to any value, plus use the greater than/less than signs to indicate ranges and limits. The offset modifies the starting search position for the pattern match function from the beginning of the packet payload. The payload size and offset have the added advantage of being a much faster way to test for a buffer overflow than a payload content check. URL contents allow search to be matched against only the URL portion of a request. Table. 1 shows tables of SVDB for the simulation.

System Modeling

Policy_Framework model

Fig. 5 shows the structure of Policy_Framework model.

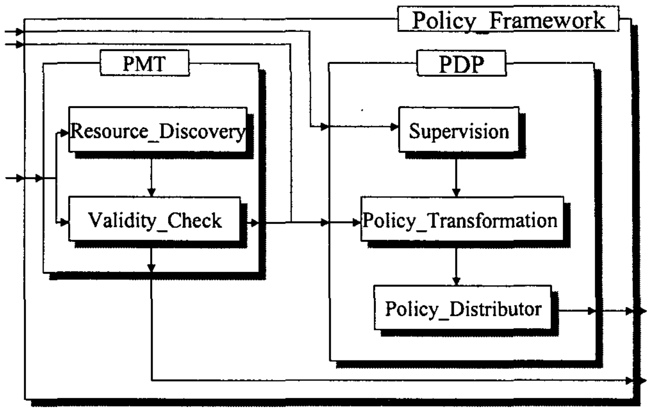

Figure 5 - Structure of Policy_Framework Model

Policy˜Framework model is divided into PMT model and PDP model. PMT model is composed of Resource_Discovery model and Validity_Check model. Resource_Discovery model determines the topology of the network, the users, and applications operational in the network. In order to generate the configuration for the various devices in the network, the capabilities and topology of the network must be known. Validity_Check model consists of various types of checks: Bounds checks, Consistency checks, Feasibility checks.

PDP model is composed into Supervision model, Policy_Transformation model and Policy_Distributor model. Supervision model receives events from network devices and monitors network usage. The PDP can use this information about the network to invoke new policy-based decisions. Policy_Transformation model translates the business-level policies into technology-level policies that can be distributed to the different devices in the network. Policy_Distributor model is responsible for ensuring that the technology-level policies are distributed to the various devices in the network.

IDS model

Fig. 6 shows the structure of IDS. IDS model is divided into Detector model, Response_Generator model and Logger model.

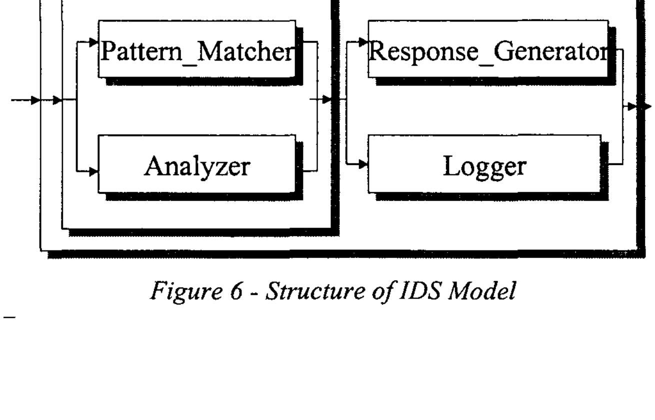

Figure 6 - Structure of IDS Model

Detector model is composed of Pattern_Matcher model and Analyzer model. Pattern_Matcher model is a rule-based expert system that detects intrusions through pattern matching procedure between packet data and rules. 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.

Collaboration between SVDB and Policy-based Framework

This section describes a collaboration between SVDB and policy-based framework. The policy management tool in the common policy-based framework is used by the administrator but we have appended the some module interface for the more automation control. The policy-based framework in the proposed system is accessed by the administrator, SVBD and intrusion detection system. Fig. 7 shows the structure and function of SVDB interface.

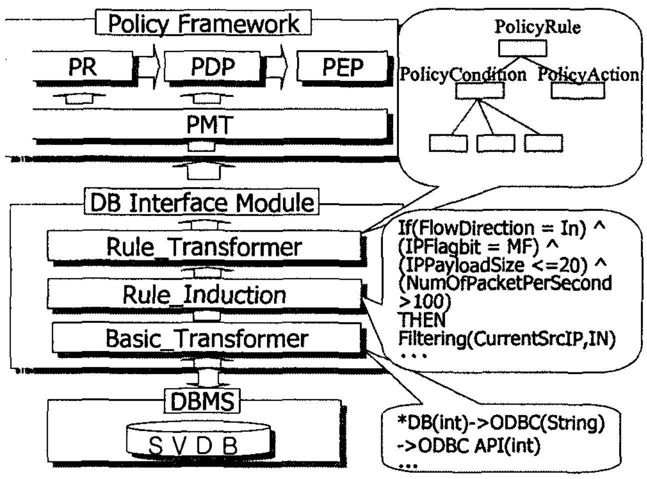
Policy Framework
PR
PDP
PEP
PMT
PolicyRule
PolicyCondition
PolicyAction
DB Interface Module
Rule_Transformer
Rule_Induction
Basic_Transformer
DBMS
S V D B
If(FlowDirection = In) ^ (IPFlagbit = MF) ^ (IPPayloadSize <=20) ^ (NumOfPacketPerSecond >100) THEN Filtering(CurrentSrcIP,IN) ...
*DB(int)->ODBC(String) ->ODBC API(int) ...

Figure 7 - Structure and function of SVDB interface

The main function of each component is as follows:

- Basic_Transformer: Basic_Transformer provides a connection of DB, data type converting and basic type checking.

- Rule_Induction : Rule_Induction builds a decision tree from DB using tree induction algorithm. Rule_Induction serves as a postprocessing of tree induction. A rule is constructed by a particular path from root to a leaf in the decision tree.

- Rule_Transformer: Rule_Transformer provides a transformation between a policy rule and a class of PCIMe. Policy conditions, policy actions, a variable and a value are compatible with the type of each class in PCIMe.

Policy rule induction from SVDB

A well-known tree induction algorithm adopted from machine learning ID3, which employs a process of constructing a decision tree in a top-down fashion. A decision tree is a hierarchical representation that can be used to determine the classification of an object by testing its values for certain properties. The main algorithm is a recursive process. At each stage of this process, we select a property based on information gain calculated from the training set. The skeleton of the ID3 algorithm is shown below [15].

Algorithm ID3
Input: a set of example
Output: a decision tree
Method:

```
ID3_tree (examples, properties)
If  all entries in examples are in the same
    category of decision variable
    Return a leaf node labeled with that category
Else
    Calculate information gain;
    Select a property P with highest information
    gain;
    Assign root of the current tree = P;
    Assign properties = properties - P;
    for each value V of P
        Create a branch of the tree labeled with V;
        Assign examples_V = subset of examples
        with values V for property P ;
        Append ID3_tree (example_V, properties) to
        branch V
```

The decision tree can be constructed as shown in Fig. 8 for the simulation.

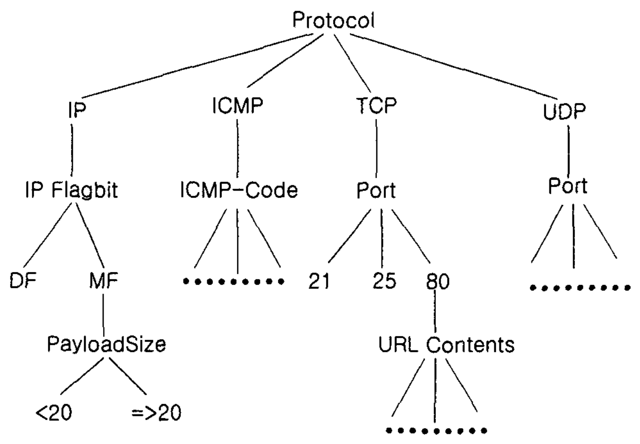

Figure 8 - ID3 tree constructed

In the decision tree, a leaf node denotes the attack name while a non-leaf node denotes a property used for decision. Rule induction can be used in conjunction with tree induction. A rule can be constructed by following a particular path from root to a leaf in the decision tree, with variables and their values involved in all the nonleaf nodes as the condition, and the classification variable as the consequence.

Converting a policy rule into PCIMe

The policy rule is transformed into PCIMe. Policy conditions, policy actions, a variable and a value are converted into the type of each class in PCIMe. To illustrate the basic idea of converting a policy rule into PCIMe, here we use a simple example.

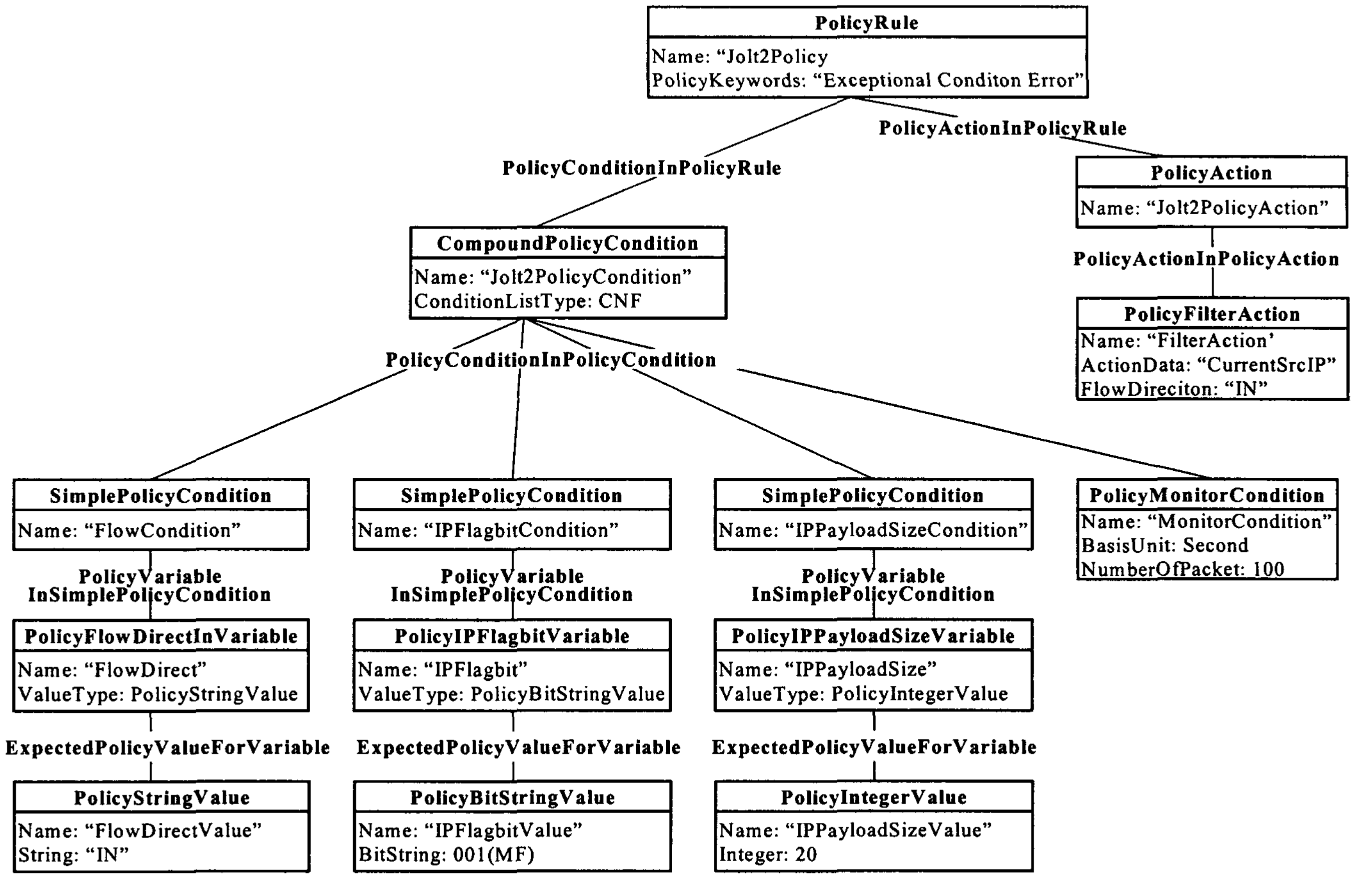
PolicyRule
Name: "Jolt2Policy
PolicyKeywords: "Exceptional Conditon Error"
PolicyActionInPolicyRule
PolicyConditionInPolicyRule
PolicyAction
Name: "Jolt2PolicyAction"
PolicyActionInPolicyAction
CompoundPolicyCondition
Name: "Jolt2PolicyCondition"
ConditionListType: CNF
PolicyFilterAction
Name: "FilterAction'
ActionData: "CurrentSrcIP"
FlowDireciton: "IN"
PolicyConditionInPolicyCondition
SimplePolicyCondition
Name: "FlowCondition"
SimplePolicyCondition
Name: "IPFlagbitCondition"
SimplePolicyCondition
Name: "IPPayloadSizeCondition"
PolicyMonitorCondition
Name: "MonitorCondition"
BasisUnit: Second
NumberOfPacket: 100
PolicyVariable InSimplePolicyCondition
PolicyVariable InSimplePolicyCondition
PolicyVariable InSimplePolicyCondition
PolicyFlowDirectInVariable
Name: "FlowDirect"
ValueType: PolicyStringValue
PolicyIPFlagbitVariable
Name: "IPFlagbit"
ValueType: PolicyBitStringValue
PolicyIPPayloadSizeVariable
Name: "IPPayloadSize"
ValueType: PolicyIntegerValue
ExpectedPolicyValueForVariable
ExpectedPolicyValueForVariable
ExpectedPolicyValueForVariable
PolicyStringValue
Name: "FlowDirectValue"
String: "IN"
PolicyBitStringValue
Name: "IPFlagbitValue"
BitString: 001(MF)
PolicyIntegerValue
Name: "IPPayloadSizeValue"
Integer: 20

Figure 9 - Policy rule of the jolt2 attack

Jolt2 attack is a type of DOS attacks. It slices a IP datagram into small pieces and then transmits the packets of these pieces to the target system. As a result, the utilization of the CPU reaches close to 100 percents and can't handle any other processes. We obtain the following rule of jolt2 attack through rule induction: "IF (FlowDirection = IN) ∧ (IPFlagbit = MF) ∧ (IPPayloadSize <= 20) ∧ (NumOfPacketPerSecond > 100) THEN Filtering(CurrentSrcIP, IN)". Fig. 9 shows an object model to represent the security policy for jolt2 attack. The classes comprising the PCIMe are intended to serve as an extensible class hierarchy (through specialization) for defining policy objects that enable network administrators and policy administrators to represent policies of different types. The PolicyIPPayloadsizeVariable class, PolicyMonitorCondition and PolicyFilterAction are inserted into the inheritance hierarchy the original PCIMe classes. This object model is distributed to network devices to be enforced, and is used to map into proper security configurations.

Simulation Result

Two main categories of attacks were simulated, they are:

DOS: smurf, ping-of-death, jolt2.

Probing: ping-sweep, port-scan.

The blocking ratio, false positive and false negative error ratio are measured for the performance indexes in the simulation

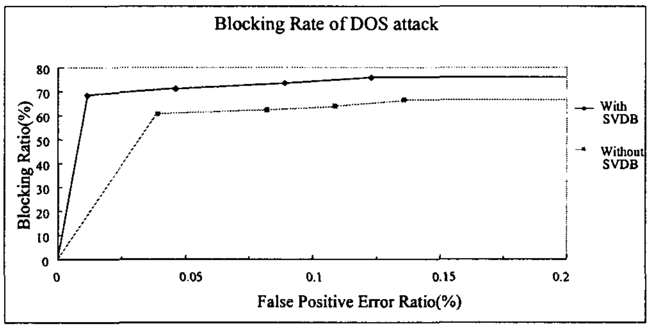

Figure 10 - False Positive Error Ratio of DOS attack

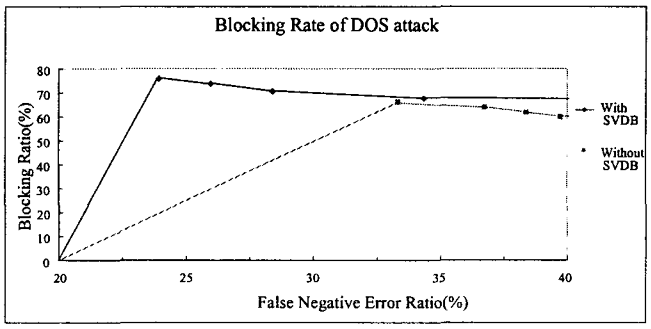

Figure 11 - False Negative Error Ratio of DOS attack

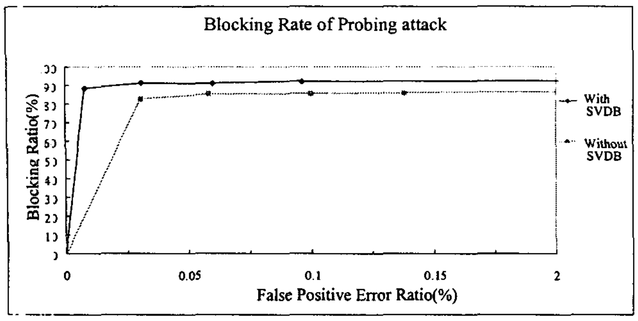

Figure 12 - False Positive Error Ratio of Probing attack

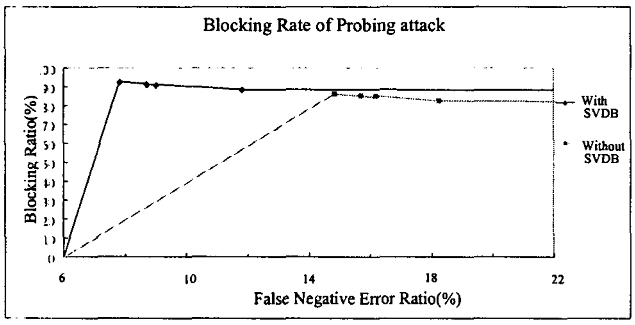

Figure 13 - False Negative Error Ratio of Probing attack

Fig 10, 11 are the case of DOS attack and Fig. 12,13 are the case of Probing attack. The result shows that the blocking ratio of the system with SVDB is higher than the blocking ratio of system without SVDB. The false negative and positive error ratio of system with SVDB are also lower than the false negative and positive error ratio of system without SVDB. This phenomenon related to the additional information in SVDB. The proposed system detects effectively attacks through the additional system information in SVDB. Fig. 10 shows that the false positive error ratio is increased by strengthening of the security level. This increase in the error ratio is due to the fact that the higher the security level, the more error security systems make in that case. But the blocking ratio of probing attack has fixed false positive error ratio, and the blocking ratio of probing attack is higher than the blocking ratio of DOS attack. Probing attacks have relatively limited variance because they all involve making connections to a large number of hosts or ports in a given time frame. On the other hand, DOS has a wide variety of behavior because they exploit the weaknesses of a large number of different network or system services. As a result, the security system can detect a high percentage of probing attack. These results means that proposed system allows to have simplified network management and provide the added accuracy and efficiency of security systems.

Conclusion

We presented a policy-based network simulation environment for the security system. The security system makes a various network situations-the policies should be applied to change the network states. These situations include the response of intrusion detection system and policy change by the firewall, etc. We also proposed the structure for policy rule induction from vulnerabilities stored in SVDB. The security policy rule provides the added accuracy and efficiency in safe guarding the network. The simulation environment can be a good tool to analyze or test a policy, it can help a manager to know that if the applied policies work as expected, and further, it can optimize use of the current network infrastructure.

Our future work includes simulation on diverse types of intrusions, and devising a user interface to verify the policy.

Acknowledgments

This work was supported by grant No.R05-2002-000-00107-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

[1] Wang Changkun, (2000). "Policy-based network management," *Communication Technology Proceeding, 2000. WCC-ICCT 2000, International Conference on*, Vol. 1. pp. 101-105.

[2] Verma, D.C. (2002). "Simplifying network administration using policy-based management," *Network, IEEE*, Vol 16, pp 20-26, March-April.

[3] F. Cohen. (1999). "Simulating Cyber Attacks, Defences, and Consequences," *Computer & Security*, Vol.18, pp. 479-518,

[4] E. D. Zwicky, S. Cooper and D. B. Chapman. (2000). *Building Internet Firewalls second edition* :O'reilly & Associates.

[5] T.H. Cho and Bernard P. Zeigler. (1997). "Simulation of Intelligent Hierarchical Flexible Manufacturing: Batch Job Routing in Operation Overlapping," *IEEE trans. Syst. Man, Cyber. A*, Vol. 27, pp. 116-126.

[6] Dinesh C. Verna. (2001). *Policy-Based Networking: Architecture and Algorithm*, New Rider.

[7] Dave Kosiur. (2001). *Understanding Policy-Based Networking*, John Wiley & Sons, Inc.

[8] M. Stevens. (1999). "Policy Framework". Internet Draft, draft-ietf-policy-framework-05.txt.

[9] B. Moore, et al. (2000). "Policy Core Information Model-Version 1 Specification," IETF RFC 3060.

[10]B. Moore, et al. (2003). "Policy Core Information Model (PCIM) Extensions," IETF RFC 3460.

[11]NIST. (1995). "An Introduction to Computer Security : The NIST Handbook," Technology Administration, U.S.A.

[12]M. Bishop. (1999). "Vulnerabilities Analysis," Proceedings of the Recent Advances in Intrusion Detection pp. 125-136.

[13]Robert A. Martin. (2001). "Managing Vulnerabilities in Networked Systems," *IEEE Computer*, Vol.34, No.11,

pp. 32-38.

[14]http://icat.nist.gov, ICAT Metabase

[15]Zhengxin Chen. (2001). *Data Mining And Uncertain Reasoning: An Integrated Approach*, John Wiley & Sons