A Framework for the Policy-based Security Surveillance of a Large Scale Network

Eun Young Park¹, Young Chul Shim², Sung Won Kim¹

¹ Electronics and Telecommunications Research Institute, Computer System Department
  161 Gajeong-dong, Yuseong-gu, Daejeon, Korea
  {eypark, swkim}@etri.re.kr

² Hongik University, School of Information and Computer Engineering
  72-1 Sangsudong, Mapogu, Seoul, Korea
  shim@cs.hongik.ac.kr

Abstract. Security management and maintenance are difficult in a large scale network because there are many different kinds of security tools which each has its own information model and policy. We introduce a framework for policy-based security surveillance in a large network which has various security tools. The framework is based on the manager-agent concept. The central manager stores security policies and monitors network topology and states in real time. An agent installed in a managed node reports the security related state to the manager and executes management commands received from the manager. When the manager finds a change in the network topology or state, it checks if the state of network after the change conforms to security policies. If any deviation is found, the manager plans executes management commands to return the network into the state conforming to the policies.

1 Introduction

Nowadays the number of attempts or intruding into network has been increased and the underlying methods have been sophisticated in accordance with the development of network technology. Although security technology has also been developed considerably and delicately in these situations, here is a limit in security maintenance of a large scale and dynamic network. A framework is required which holds hierarchical structure to integrate analysis information related to all systems to overcome all possible intrusions.

In order to achieve this goal, we propose a policy-based framework for maintaining security of a computer network systematically, automatically, and effectively. The proposed approach is based on the manager-agent paradigm. The security manager stores security policies in its database. The security policies are the rules governing the choices in behavior of a system in terms of security. The manager is aware of network topology changes and security related events occurring in nodes. A manager monitors topology changes and the events and checks whether these modified states conform to the security policy. If any discrepancy is found, the manager plans and executes the required actions to make the network states conform to the security
policy again. Agents installed in network nodes detect security related events occurring in nodes and report them to the manager. They receive commands from the manager and execute them, too.

In this paper we present the architecture and Entity-Relationship(ER) model of our framework for overall security management. The design of each of Management Information Tree(MIT) and Management Object(MO) in ER-model will be provided. Monitoring and controlling functions are achieved by polling the agents and notification of manager by the agents. Management information is stored in a virtual information store as Management Information Base(MIB). Extension of management information is also possible by creating new MIBs or increasing existing ones. Afterward, the security policy used to design information model for the suggested framework will be described.

This paper is organized as follows. In Section 2, related work to our framework is presented. Section 3 describes policy-based security surveillance framework. Section 4 designs information model in the framework and also each of detailed MO and MIB modules will be designed. In Section 5 security policy is applied to PDL.

2 Related Work

SNMP(Simple Network Management Protocol) allows for the management of systems in a TCP/IP network within a coherent framework. They define the overall management architecture based on the manager-agent concept. A Management Information Base (MIB) that defines the various information that can be queried and set by the manager. A set of common structures and an identification schema, called the Structure of Management Information (SMI) that is used to reference the variables in the MIB[4].

CIDF(Common Intrusion Detection Framework) architectural model divides an IDS into components, all of which have a persistent identity. It primarily focuses on the cooperation between homogeneous intrusion detection components. CIDF adopts a view of IDSs in which they consist of discrete components that communicate via message passing. Our approach borrows their idea for expressing messages and tries to include the message in security management commands/response[3].

EMERALD(Event Monitoring Enabling Responses to Anomalous Live Disturbances) is a set of tools designed to handle detecting intrusions across large infrastructures. AAFID(Autonomous Agents for Intrusion Detection) takes a different approach from the traditional monolithic intrusion detection systems. This architecture uses multiple autonomous agents working cooperatively to detect intrusions whereby improving the survivability of the system. This project shares many similarities with our research when facing trade-offs in architecture, scalability, performance, and security[9].

Frimato(A Novel Firewall Management Toolkit) presented a design and implementation of a prototype for a new generation of firewall and security management tools. It shows the demonstration that the task of firewall and security configuration/management can be done successfully at a level of abstraction analogous to modern programming languages. We referred to this for design of ER-model in this paper[8].
Ponder is a language for specifying management and security policies for distributed systems. It has been developed as part of ongoing research being carried out by the group into the use of policy in distributed systems management[20]. Our approach proposed a framework for cooperation of diverse security tools in a large network from the viewpoint of policy-based network management.

3 Architecture of our Framework

In this section we present architecture of a framework for policy-based security and explain how to manage entities in a network model. Fig. 1 shows the overall architecture. A security manager consists of a management interface, a security policy analyzer, a PIB, as security policy information database, a network state database, a network topology database and a management engine. The functions of each module are described as follow[2].

**Management Interface.** A human manager inputs security policies and network topology manually through it. He/She can also query the security policy, network state, and network topology database contents through it.

**Security Policy Analyzer.** It first checks the syntax of the policies input by a human manager. It then analyzes if they are consistent with security policies already existing in security PIB. If the consistency is proved, the new policies are stored in the security PIB.

**Security PIB.** It stores security policies and notifies the management engine of any change in the security PIB contents.

**Network State Database.** It received event reports from agents installed in managed nodes, stores them, and then notifies the management engine.

![Fig. 1. Architecture for Policy-based Security Surveillance](image-url)
Network Topology Database. It stores information about the network topology that it received either from a human manager or the network configuration manager.

Management Engine. It is the brain of the Security Manager. It checks if the network state is conforming to the security policies under the current network topology. If it finds any network nodes whose states are not conforming to security policies, it plans and executes management command on those nodes so that their state many become conforming to security policies again. The execution of the engine is triggered by change notifications from security policy, network state, network topology database.

The Configuration Manager monitors network elements such as nodes and links, detects any topology changes due to new nodes/links or failing node/links, and notifies them to the security manager.

An agent installed in each managed node such as security tools, router, and hosts consists of modules as shown in Fig. 1 and their functions are as follow.

Communication Module. It communicates with the manager

MIB. It stores events and states collected from the node in which the agent is installed.

Translator. The syntax and semantics of the messages exchanged between agents and managers follow standard format. But the message between an agent and a managed node has the format and meaning specific to that node. The translator performs the translation function between these message formats.

Event Detector. It detects the occurrence of events requested by the manager.

Agent Engine. It executes the management commands received from the manager and notifies the manager of the detected events. All the management commands except the requests for event detection are performed directly on either MIB or the managed node.

The Architecture shows operation of key elements of our framework. Detailed configuration and relationship between the components of the framework are shown through Entity-Relationship model in the following section. Also the information model of elements related to the security is presented for standardized operation in this framework.

4 The Modeling Framework

In this section we present our framework model. We start modeling of ER-diagram in Section 4.1 and design MOs and MIBs with an object-oriented way in Section 4.2. Network management protocol as SNMP uses MIBs in order to exchange information. In this paper we focus on information model of security components
4.1 Entity-Relationship Model

The ER-diagram that included a correlation of elements in our framework is same as Fig. 1. Our Framework is consisted of several managers, host, and security components. Manager is composed of Configuration Manager and Security Manager. A Configuration Manager manages all hosts, firewalls, and Intrusion Detection System(IDS)s in our framework for whole network configuration. A Security Manager records any event of logs of security components it and makes and updates rules by gathering information from all components. Also it handles rules for detecting vulnerability when they communicate information in network and it manages the system to apply Policy Rule. Security component is composed of firewalls, and IDSs. Each has unique IP and host name and it has a Role each Role group. A Role-group entity consists of a set of Roles. Firewalls are connected to Router, Security Manager, and Configuration Manager. It filters a suspicious packet in network by rules. IDSs include host-base IDSs, network-based IDSs, and IDS managers. IDS managers check an intrusion through audit data from connected hosts with Configuration Managers, Security Managers, and Host Groups[10].

![Fig. 2. The Entity-Relationship Model](image-url)
4.2 Detailed Design of Each MO’s MIT and MIB

We design MIT of some components in our framework. In this section, we treat Firewall, IDS, Security Manager, and Policy Manager. We designed MIT, MO, and MIB about each component.

Entities such as network, routers, hosts, and security tools can be designed individually as an object class. Object classes are composite by attribute, operation and notification.

- Attribute: has a value as property of an object class.
- Operation: is an action for security surveillance management.
- Notifications: notify to managers the event that was happened in the object.

All object classes conform to object-oriented concept. An object can make a new object having in heritance using the feature of encapsulation. Also, it can easily manage objects using the feature of homogeneity. This means that one object can include one or more other objects and the object can be the sub-object of another object. But one object must be included in only one upper object. Each MO is stored in object-oriented database(MIB). MIB specifies inheritance and aggregation relationship of MIB, any hierarchical situation, names of property and type, notifications, and actions on each MO class.

4.2.1 Firewall MO’s MIT and MIB

Firewall MIT is composed of Hardware MO, Software MO, Rule MO, Log MO, Event Forwarding Discriminator(EFD) MO, and Log Record MO in Fig. 3. The event or state log that passed through an actual logs are stored in Log Record.

![Firewall MIT](image)

**Fig. 3. Firewall MIT**

**Firewall MO.** This store information about the type of a firewall, name, version, information about installation, test and reboot.

**Hardware MO.** This treats information hardware of a firewall, that is, a kind of the system which the firewall is installed, hardware state, hardware full, and hardware error.

**Software MO.** This treats information about software of a firewall.

**Rule MO.** This defines series of rules that firewall pass the packet or not. Rule MO has source a destination address, a service address, and an action according to a rule base of a general firewall.
Log MO. This defines a standard to control log of information about the status that is for a log to be generated, test about log become records, validity of log.

EFD MO. This reports and logs events when an event happens in the system management side or controls another MOs.

BasicEvent MO. This defines the basic table containing information that is logged for every event on the firewall.

NetEvent MO. This is a detailed table with information related to network events or events involving user of the firewall resources and services.

HealthEvent MO. This treats health of a firewall and information related to state of the firewall. This event occurs while managing hardware or software resource state. It treats detailed information about a resource type having some troubles.

MgmtEvent MO. This treats all information related to general management operations in firewall. For example it is used when configuration changed or a patch varied.

ResourceStat MO. This provides information of resources about query and includes a hardware or software module in the firewall. It operates when has a trouble in hardware, operating system or service.

PacketStat MO. This includes information of the packet that is treated having been related to resources and service of a firewall. For example, store a number of ignored packets, encrypted packets.

The firewall MIB is defined as Fig. 4.

<table>
<thead>
<tr>
<th>firewall MANAGED OBJECT CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATTRIBUTES</strong></td>
</tr>
<tr>
<td>ProductName</td>
</tr>
<tr>
<td>RestartEvent</td>
</tr>
<tr>
<td>UpEvent</td>
</tr>
<tr>
<td>DownEvent</td>
</tr>
<tr>
<td><strong>ACTIONS</strong></td>
</tr>
<tr>
<td>Test</td>
</tr>
<tr>
<td>Reboot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rule MANAGED OBJECT CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATTRIBUTES</strong></td>
</tr>
<tr>
<td>RuleIndex</td>
</tr>
<tr>
<td>RuleObject</td>
</tr>
<tr>
<td>RuleSourceIP</td>
</tr>
<tr>
<td>RuleSourcePort</td>
</tr>
<tr>
<td>RuleDstIP</td>
</tr>
<tr>
<td>RuleDstPort</td>
</tr>
<tr>
<td>RuleAction</td>
</tr>
<tr>
<td>RuleProtocol</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NetEventLog MANAGED OBJECT CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATTRIBUTES</strong></td>
</tr>
<tr>
<td>NetEventInterface</td>
</tr>
<tr>
<td>NetEventProtocol</td>
</tr>
<tr>
<td>NetEventICMPCommand</td>
</tr>
<tr>
<td>NetEventSrcIPAddress</td>
</tr>
<tr>
<td>NetEventMappedSrcIPAddress</td>
</tr>
<tr>
<td>NetEventDstIPAddress</td>
</tr>
<tr>
<td>NetEventMappedDstIPAddress</td>
</tr>
<tr>
<td>NetEventSrcIPPort</td>
</tr>
<tr>
<td>NetEventMappedSrcIPPort</td>
</tr>
<tr>
<td>NetEventDstPort</td>
</tr>
<tr>
<td>NetEventMappedDstIPPort</td>
</tr>
<tr>
<td>NetEventService</td>
</tr>
<tr>
<td>NetEventServiceInformation</td>
</tr>
<tr>
<td>NetEventAuthdEntity</td>
</tr>
<tr>
<td>NetEventRuleID</td>
</tr>
<tr>
<td>NetEventActionReasion</td>
</tr>
<tr>
<td>.....</td>
</tr>
</tbody>
</table>

Fig. 4. An Example of firewall MIB

4.2.2 IDS MO’s MIT and MIB
Most of MO classes take a format to be same as a firewall MO except for Analyzer and a Sensor part which is added in software. Information about event and alarm occurring in IDS is stored in Log Record like firewall.
The MIT of IDS is same as Fig. 3. We now define Alert MO, Analyzer MO and Sensor MO in IDS MO in Fig. 5.
Alert MO. This defines the basic table containing information that is logged for every event on IDSs.

Analyzer MO. It stores detailed information of analyzers and scans information about a suspicious intrusion in IDSs.

Sensor MO. It stores information about a sensor and detecting an event in IDSs.

Rule MO. It stores information of policies of IDSs. It includes attack types, information of rule related IDS alert

![IDS MIT diagram](image)

**Fig. 5. IDS MIT**

<table>
<thead>
<tr>
<th>IDS MANAGED OBJECT CLASS</th>
<th>ALERTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentIndex</td>
<td>AlertID</td>
</tr>
<tr>
<td>ProductName</td>
<td>AlertLocalAddressType</td>
</tr>
<tr>
<td>Version</td>
<td>AlertLocalAddress</td>
</tr>
<tr>
<td>AgentDescription</td>
<td>AlertInterfaceIndex</td>
</tr>
</tbody>
</table>

**NOTIFICATIONS**
- Reboot
- UpEvent
- DownEvent

<table>
<thead>
<tr>
<th>alertEventLog MANAGED OBJECT CLASS</th>
<th>ALERTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlertID</td>
<td>AlertAttackName</td>
</tr>
<tr>
<td>AlertLocalAddressType</td>
<td>AlertSrcAddres</td>
</tr>
<tr>
<td>AlertLocalAddress</td>
<td>AlertSrcPort</td>
</tr>
<tr>
<td>AlertInterfaceIndex</td>
<td>AlertDstAddres</td>
</tr>
<tr>
<td>AlertTimeStamp</td>
<td>AlertDstPort</td>
</tr>
</tbody>
</table>

**Fig. 6. An Example of IDS MIB**

4.2.3 Security Manager MO’s MIT and MO

Security Manager MO stores information of security in Log record. Fig. 7 describes the structure of Security Manager MO. Each of MOs which related policy are refer to policy core information model[18].

SecurityEventLog MO. It stores information about attack action detected by IDS or contents about the event that occurred here.

Policy MO. A set of rules, usually concerned with common Policy Group that make a general security management goal.

Policy Rule MO. A rule is being evaluated at discrete points in time given by events. At each occurrence of an event, all rules that are triggered by that event are processed in the order given by the priority values. If no priority is given, there is no preferred order for this rule.
Fig. 7. Security Manager MIT

Policy Condition MO. An expression that is evaluated to a boolean value each time the rule is triggered. If the condition contains variables that specify elements, it is evaluated for each combination of those elements.

Policy Action MO. An operation that is executed each time the condition has been evaluated to true. Attributes of the event that triggered the execution and variables specifying the elements that lead to the matching condition can be referred within the action.

Policy Group MO. An area which of security management, targeted by a policy.

This is some parts of Security Manager MIB.

![Security Manager MIB diagram](attachment:security_manager_mib.png)

### analyzer MANAGED OBJECT CLASS

**ATTRIBUTES**
- AnalyzerIndex
- AnalyzerID
- AnalyzerType
- AnalyzerNode
- AnalyzerProcessIndex

**NOTIFICATIONS**
- Shutdown
- Restart
- UserGroupChanged

### generator MANAGED OBJECT CLASS

**ATTRIBUTES**
- GeneratorIndex
- GeneratorID
- GeneratorNode
- GeneratorProcessIndex

**NOTIFICATIONS**
- Shutdown
- Restart
- UserGroupChanged

### policy MANAGED OBJECT CLASS

**ATTRIBUTES**
- PolicyIndex
- PolicyID
- PolicySource
- PolicyDestination
- PolicyDestinationGroup
- PolicyService
- PolicyLogging
- PolicyAction
- PolicyCondition
- PolicyAlias
- PolicyQos

**NOTIFICATIONS**
- PolicyModified

### PolicyCondition MANAGED OBJECT CLASS

**ATTRIBUTES**
- PolicyCondition
- PolicyTimePeriodCondition
- VendorPolicyCondition
- SimplePolicyCondition
- CompoundPolicyCondition

**NOTIFICATIONS**
- PolicyConditionChanged
5 Adaptation to Policy Specification Language

We use policy specification language in order to express our security policy. The Ponder Policy Specification language uses the term subject to refer to users, principals or automated manager components, which have management responsibility. The subject of a policy specifies the human or automated managers to which the policies apply and which interpret obligation policies. The target of a policy specifies the objects on which actions are to be performed. Domains are a means of grouping objects and are similar to file system directories[19]. We use this idea on our framework.

We use security policy language of a form as Fig.9. We do not distinguish between authorization and obligation policies. The event is generated event on the target domain and the action is the operation what is defined beforehand of subject. The constraint is the condition to determine of policy. All entries are not indispensable.

![Fig. 9. Authorization Policy Syntax](image)

We show several examples of security policies of firewall and IDS with our policy specification language.

**Example 1.**
The following is an example of security policy when approach of a specific port is controlled in a firewall, and the second is an example when specific IP is blocked off by a firewall.

```plaintext
policy FirewallAccessControlPolicy{
  subject  /Group_a/Firewall_a
  target   GroupA+GroupB
  action   ClosePort()
  constraint Service.Port()=21
}

policy FirewallAccessControlPolicy{
  subject  /Group_b/Firewall_b
  target   AllGroup
  action   BlockOff()
  constraint SourceIPAddress()=129.243.221.10
}
```
Example 2.
The following example is a kind of misuse detection in an IDS. When the user who does not have authority access to the password file and change it, the following security policy is applied. The IDS lets block off the user and reports it to security manager.

```
policy IDSMisuseDetectionPolicy{
  subject /Group_a/IDS_a
  target /Administrator/passwd
  event changedPasswdfile(userid)
  action BlockUser(userid), Log(userid), report(securityMgr)
  constraint accessPasswdFile(userid)
}
```

Example 3.
This security policy is about anomaly detection of an IDS. When the CPU usage is exceeded for long period(300 sec.) threshold, the IDS regards as an anomaly action. And then the IDS lets operation stop and reports it to security manager.

```
policy IDSabnormalDetectionPolicy{
  subject /Group_a/IDS_b
  target /Group_a/Host_a
  event getStat.cpu_usage(host_id) >= threshold(cpu_usage)
  action stop(host_id), report(securityMgr)
  constraint keeping_time() > 300
}
```

6 Conclusion

Security management of a large computer network is a difficult problem because it uses many security tools of different types and its topology and states are dynamic. In this paper we introduce a framework for managing security of a large dynamic network. The framework is based on a manager agent concept. The central manager stores security policies and monitors network topology and security related states in real time. An agent installed in a managed node such as hosts, routers, firewalls, IDSs, etc. reports the security related event to the manager and executes management commands received from the manager. When the manager finds a change in the network topology or state, it checks if the network state after the change conforms to security policies. If any deviation is found, the manager plans/executes management commands to return the network into the state conforming to policies. In this paper we presented the architecture for the policy-based security management of computer networks, the information modeling of a managed network, and a new policy description language that suit for our framework.
7 References

4. W. Stallings, SNMP, SNMPv2 and CMIP, Addison-Wesley, 1993
7. Clifford Kahn, Don Bolinger and Dan Schnackenberg, “Communication in the Common Intrusion Detection Framework v 0.7,” Internet Draft, June 1998