명령어 등급 부여를 통한 에이전트의 불법행위 방지에 관한 연구

임 용 성[†]·장 덕 성[††]·정 홍[††]

요 약

컴퓨터 네트워크 보안관리의 중요성이 크게 부각되고, 시스템에 대한 불법침입을 탐지하고자 하는 침입탐지 시스템에 관한 연구가 활발히 진행되고 있다. 또한 이동에이전트들이 돌아다니면서 서버에 해악을 일으키는 행위는 큰 문제점이 되고 있고, 사전에 이를 탐지하는 것이 매우 중요하다. 그러나 에이전트가 서버에서 활동을 개시하기 전에 불법적 행동을 탐지하는 것은 거의 불가능하므로, 본 논문에서는 에이전트마다 수행가능한 명령어의 집합을 정의하고 그 외의 명령어를 사용하는 경우를 침입으로 탐지하는 방법을 연구하였다.

에이전트에 등급을 부여하는 방법과 각 등급에 따른 명령어의 집합, 에이전트의 메시지 교환시 명령어의 사용을 검사하는 방법 등을 제시하였다. 에이전트를 등록하고 에이전트의 관리를 담당하는 ANS와, 상이한 ontology 정보를 분석하여 시스템에 사용 가능한 형태로 바꾸어 주고 필요한 메시지의 내용을 분석하는 OTS가 존재한다. 만약 에이전트의 불법적 행위가 발견되면 접속을 끊거나 에이전트의 등록을 해지한다.

A Study on the Prevent the Illegal Access by Granting the Command Ranks

Yong-Sung Lim[†] · Duk-Sung Jang[††] · Hong Jung[††]

ABSTRACT

The security management of a computer network is becoming important, and many studies have been done regarding Intrusion Detection Systems to detect illegal access to the system. It is significant to prevent mobile agents from an evil influence on the system. It is quite impossible to detect these illegal acts before it activates in servers. In this paper, We have defined allowable command sets of each agent and studied how to detect intrusions when they use disallowable commands on server.

We propose, in this paper, how to grant ranks to agents, how to give command sets according to ranks and how to keep watch on command usages when they exchange messages. The system we proposed consists of a communication module, a message checking module, an ANS(Agent Name Server), and an OTS(Ontology Type Server). We need an ANS to register and manage agents, and OTS to analyze differential ontology information, change it into usable format and analyze the necessary message. If the illegal activities of agent are detected, then the system could disconnect the access or cancel the registration of the agent.

1. 서 론

1981년에 IP프로토콜을 실험적으로 ARPANET에 적용할 때 불과 210개의 호스트를 연결하였던 인터넷이 오늘날 7백만 대 이상의 컴퓨터 시스템이 상호 연결됨에 따라 네트워크에 연결된 모든 컴퓨터가 해커들의 침입대상이 되고 있다. 이로 인하여 컴퓨터 네트워크의 보안관리의 중요성이 크게 부각되고 있으며 보안

† 준 회 원 : 계명대학교 대학원 컴퓨터공학과
†† 정 회 원 : 계명대학교 컴퓨터공학과 교수
논문접수 : 2000년 6월 24일, 심사완료 : 2000년 8월 2일

관리의 일환으로 정보보호를 필요로 하는 문서나 시스템에 대한 불법침입을 탐지하고자 하는 침입탐지 시스템(intrusion detection system)에 관한 연구가 활발히 진행되고 있다[2].

침입(intrusion)에 대한 기본 개념은 1980년에 Anderson이 '비 인가된 정보로의 접근 및 정보 조작, 그리고 시스템 무기력화를 위한 고의적이면서도 불법적인 시도'이라고 규정하였다. 그 후 1987년 Denning은 침입탐지 시스템(IDS : Intrusion Detection System)을 '허가되지 않은 사용자가 컴퓨터 시스템 또는 네트워크 상에서 불법적인 접속, 정보의 조작, 오용, 그리고 남용 등을 시도했을 경우, 의심스러운 행위를 감시하여 조기에 침입을 발견하여 처리하는 시스템'으로 정의하고 있다[9].

일반적으로 시스템의 안전성과 사용 편리성은 서로 상반되는 개념이고 안전한 시스템 설계는 엄청난 비용이 소요되므로, 어떠한 공격에 대해서도 안전한 이상적인 시스템을 설계하는 것은 거의 불가능하다. 또한 시스템에서 불법적 행위에 대한 대처 방법으로 모든 파일을 암호화하여 저장할 수 있지만, 암호 알고리즘 선정, 키관리 문제, 시스템 관리자의 역할 조정 등의 새로운 문제를 발생시킨다[1, 4].

본 논문에서는 하나의 조직체에서 각 호스트에 속한 여러 에이전트중에 이 에이전트들이 이동을 하며 접속하는 서버에 불법적인 행위를 탐지하기 위해 에이전트에 등급을 부여하고, 등급에 따른 명령어 집합을 정의하여 서버를 보호하고 에이전트들을 제어하는 방법을 연구하였다.

2장에서는 침입의 방법과 침입탐지, 에이전트, KQML 등 관련연구에 대해 살펴보고, 3장에서는 본 논문에서 제시하고자 하는 침입탐지 방법에 대해 구체적으로 서술하였다. 4장에서는 구현방법에 대하여 설명하였으며, 5장에서는 결론 및 향후과제를 제시한다.

2. 관련 연구

2.1 침입의 정의 및 유형

침입은 크게 두 가지로 정의할 수 있다. 첫 번째 정의는 시스템 자원의 무결성, 기밀성이나 가용성을 저해하기 위한 일련의 동작이라고 할 수 있다[6-8]. 무결성(integrity)은 컴퓨터 시스템 자산(assets)은 오직 인가 받은 자만이 내용을 수정할 수 있도록 보장되어야 한다. 기밀성(confidentiality)은 컴퓨터 시스템내의 정보는 오직 인가 받은 자만이 접근(access)할 수 있도록 보장되어야 한다. 접근에는 읽기와 쓰기 등이 포함되며 어떤 정보의 존재 사실 자체도 노출되어서는 안된다. 그리고 가용성(availability)은 컴퓨터 시스템 자산은 오직 인가 받은 사람만이 사용할 수 있도록 그리고 언제나 사용 가능하도록 보장되어야 한다.

두 번째 정의는 컴퓨터 시스템의 보안 정책을 파괴하는 행위라고 할 수 있다. 관리자나 사용자를 속이는 행위, 권한이 있는 사용자의 권한을 뺏는 행위, 시스템 취약점을 이용하는 행위나 침입을 위해 잘 짜여진 프로그램을 이용하는 수법들이 있으며, 서비스의 취약한 부분을 이용하거나 시스템이 정상동작을 할 수 없도록 방해하는 행위도 침입의 한 수법이라고 할 수 있을 것이다.

일반적으로 해커라고 불리우는 침입자들의 침입수법은 <표 1>과 같이 여러 가지 방법들을 이용하고 있다[3].

〈표 1〉 일반적인 침입 수법

침입 수법	설 명
Social Engineering	관리자나 사용자 속이기
Impersonation	일반 사용자의 권한 뺏기
Exploits	시스템 보안 취약점 이용
Transitive Trust	신뢰하는 호스트나 네트워크 위장
Data Driven	Attack Program, Trojan, Backdoor, Virus
Infrastucture	Protocol/System 기본 기능 취약점
Denial of Services	시스템의 정상 동작 방해

2.2 침입탐지의 요구 사항과 방법

침입탐지 시스템은 최근에 다양한 기법과 모델들이 개발되어 다양해졌지만 기본적인 요 구사항은 다음과 같다[4, 5].

- 침입 탐지 자체는 시스템 운영자 및 보안 관리자의 별도 개입 없이 동작해야한다. 즉, 대상 시스템에 대해 백그라운드 상에서 동작한다. 그러나 필요시 보안 담당자의 요청에 의한 내부적 작업에 대해 외부에서 이를 알 수 있어야 한다.
- 시스템의 완전한 파괴를 피하기 위해 탐지시스템은 스스로 자신을 모니터링하여 방어 시스템이 침해되는 것을 방지할 수 있어야 한다.
- 시스템에 걸리는 부하를 최소화해야 한다.
- 행위에 대해 정상적인 행위와의 차이를 관찰해야

한다.

- 모든 시스템은 서로 다른 사용 패턴을 가지고 있으므로 방어 메카니즘도 이러한 패턴 에 따라 적용될 수 있도록 침입 탐지 시스템은 적용 시스템에 따라 쉽게 가공되어질 수 있어야 한다.
- 새로운 응용프로그램의 추가는 시스템 프로파일의 변화를 초래하므로 기존의 시스템 사용에 대한 허용행위 여부 역시 변화가 가능해야 한다.
- 탐지시스템의 오류발생이나 기능 마비가 매우 힘들어야 한다.

2.3 이동 에이전트

이동 에이전트(Mobile Agent)는 네트워크 에이전트(Network Agent) 또는 순회 에이전트(Itinerant Agent)라고 하며, '프로그램 자체가 네트워크를 돌아다니며 수행되는 프로그램'을 말한다[8]. 이동 에이전트와 유사한 예로는 자바 애플릿을 들 수 있다. 하지만 자바는 웹 브라우저에서 요구할 때 서버에 있는 애플릿이 브라우저로 이동하는 반면, 이동 에이전트는 자신의 판단에 의해 이동하는 것이 다르다.

이동 에이전트가 특정 컴퓨터에서 수행되려면 이동 에이전트 서버가 필요하다. 이동 에이전트 서버는 에이전트 수행 엔진, 장소, 그리고 API로 구성되며, API는 다시 외부 프로그램 인터페이스와 기억장소 인터페이스, 전송 인터페이스로 나뉜다. 외부로부터 전달된 이동 에이전트는 전송 인터페이스를 통해 특정 장소에 위치한다. 특정장소에 위치한 이동에이전트는 다른 에이전트와 통신하면서 작업을 수행한다. 이때 에이전트의 스크립트 파일은 에이전트 수행엔진 내에 있는 인터프리터를 통해 번역, 수행된다. 이동 에이전트는 서버에 존재하는 다른 응용 프로그램의 서비스를 받을 수도 있는데 이때 사용되는 것이 외부 응용 프로그래밍 인터페이스이다. 그리고 이동 에이전트 수행이나 관리를 위해 사용되는 기억장소 인터페이스가 있다.

이동 에이전트는 에이전트 구현 언어가 스크립트 언어로 작성되고 인터프리터로 수행된다는 것이 특징이다. 또한 자신을 다른 컴퓨터로 이동시키는 명령이 있으므로 그 명령을 만나면 다른 서버로 이동할 수 있다. 그 외에도 자신의 판단에 따라 이동하는 능력뿐 아니라, 동일한 에이전트를 복제해 다른 컴퓨터로 보내고 그들이 가져온 결과를 모아 복합적인 결과를 만들기도 한다.

이동에이전트가 갖는 가장 큰 문제점은 보안이다[8]. 자신이 원하는 서비스를 제공하는 서버를 어떻게 효율적으로 찾는 가도 중요한 문제이지만 에이전트가 이동을 하면서 수 많은 서버를 돌아다니며 일으킬 수 있는 문제점도 연구의 중요한 대상이 된다.

2.4 KQML

KQML은 정보와 지식을 교환하기 위한 언어이자 프로토콜이다. 또한 에이전트의 지식을 공유하는 것을 지원하는 메시지 포맷과 메시지 관리 프로토콜이다. 이에 본 논문은 Java로 프로그램하고 KQML을 이용하였다. KQML은 에이전트 통신언어로써 에이전트간의 정보교환과 통신에 있어서 가장 적합한 언어로써 다음과 같은 특성을 가지고 있어서 본 논문에서 이용하였다.

- 정보 교환과 관련한 통신에 중점을 둔 통신언어이다.
- 기존의 프로그램에 쉽게 포장될 수 있다.
- 정보의 내용과 형식에 독립적이다.
- 임의의 지식 베이스들에 대한 응용 시스템을 설계할 때 지식 베이스의 위치와 이를 필 요로 하는 곳의 위치가 제한적이지 않다.

실제 에이전트들 사이에 오고 가는 내용표현이 메시지 포장기에 의해 메시지 표현으로 싸여서, 이것이 다시 통신 포장기에 의해 포장된 것으로 이해할 수 있다. 즉, 기존의 통신 프로토콜의 계층 구조처럼 내용계층(content layer), 메시지 계층(message layer), 통신계층(communication)의 3단계 계층으로 이루어진다.

KQML을 사용할 때 에이전트는 내용(content) 메시지를 자신의 언어로 구성한 후, 이를 KQML 메시지 안에 싸서 보내게 된다. 내용 메시지는 어떤 표현 언어로 쓰여지든지 ASCII 문자열이든지 이진표현법들 중 하나이든지 상관이 없다. 내용이 언제 시작되고 언제 끝나는가 외에는 메시지의 내용 부분에 관심을 가지지 않는다.

메시지층은 내용계층에 부가적인 정보를 포함시킨 것이다. 메시지는 그 성격에 의해 크게 내용 메시지와 선언 메시지로 나뉜다. 내용 메시지는 질의(query)와 같은 직접적인 지식 전달에 관련된 것이고, 선언 메시지는 직접 지식전달은 아니나 이러한 활동에 관련된 메타 정보에 대한 것이다.

통신층은 가장 바깥 계층으로 내용계층과 메시지 계

층을 거친 메시지를 통신에 관련된 부가 정보를 추가하게 된다.

3. 시스템 설계

인터넷의 이용자가 급증함에 따라 네트워크를 통하여 많은 정보가 제공되어지고 이에 다수의 사람들이 이를 이용하고 요청하고 있다. 사용자가 직접 자료를 찾아 돌아다니고 서버에 접속해서 요청하기도 하지만 요즘에는 사용자의 작업을 대신하여 주는 에이전트의 등장으로 보다 편리하게 이용하고 있다[10]. 그러나 사용자의 의도에 따라 제작되는 에이전트가 사용자의 불건전한 목적에 사용되어 에이전트가 접속하는 서버에서 해를 입힌다면 이 또한 큰 문제가 될 수 있다.

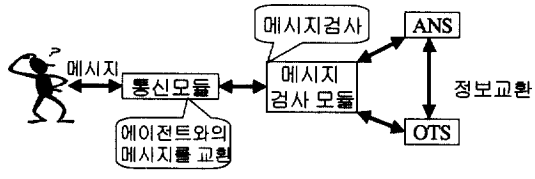
본 연구에서는 사람을 대신하는 에이전트의 침입시 불법적인 행위에 따른 감시 및 행동에 대한 침입탐지 시스템에 대해 설계하고 실험한다. 실험의 대상으로 한 조직내에서 각기 다른 역할을 수행하는 에이전트를 가정하고 호스트에 소속된 에이전트가 서버에 접속할 경우에 대해 실험하였다.

본 실험의 모델은 기존의 사람에게 주어지는 등급과는 달리 그 사람에게 소속된 에이전트에게 등급을 부여함으로써 현재 많이 이용되어지고 있는 에이전트를 제어하고, 서버를 보호하고자는 생각에서 독자적인 모델을 제안하고 실험하였다.

3.1 전체 구조

(그림 1)은 시스템의 전체구조이다. 에이전트와의 통신을 위해 통신모듈을 거쳐 들어온 메시지는 검사과정을 거쳐 메시지를 검사하고 ANS와 OTS로 에이전트에 대한 정보를 전달하여 통신이 이루어진다. 시스템에서 에이전트의 침입과 대응의 단계는 다음과 같다.

① 사용자의 에이전트가 네트워크상에서 접속하고자 하는 서버에 접근한다.
② 에이전트는 사용자를 대신하여 서버에 접속하고, 그에 따른 행동을 한다.
③ 에이전트는 서버와 통신을 위해 메시지를 보낸다.
④ 접속하는 에이전트는 서버에 등록을 하게 되고 등록시 에이전트에게 맞는 등급이 부 여, 사용하는 명령을 제한하게 된다.
⑤ 이 때 메시지의 내용을 분석하여 규정에 어긋나는 내용이 있을 경우 서버의 접속을 끊거나 등록을 해지하게 한다.



(그림 1) 전체 시스템 구조

서버는 에이전트에 대한 등록과 연결등의 모든 정보를 관리하기 위해 ANS(Agent Name Server)를 가진다. 이 서버는 에이전트의 초기 연결시 에이전트의 등록을 담당하고, 에이전트가 서버에 메시지를 보낼 경우 어떤 에이전트가 메시지를 보냈는가에 대한 정보를 보여주는 역할을 수행한다. ANS에 대한 구조는 <표 2>와 같은 구조를 가진다. 이름과 종류, 주소등 기타의 정보가 들어가게 되고, 이것으로 에이전트를 구별하게 된다. 여기에서 각 에이전트를 관리하며, 각 호스트에서 접속한 에이전트의 정보와 로그 온, 오프를 관리를 담당하게 된다.

〈표 2〉 ANS의 구조

AgentName
{name}
: AgentClass
{mailagent | searchagent |...}
: Address
{203.…,…,…}
: Action
{True|False}

또한 서버는 OTS(Ontology Type Server)를 가진다. 이는 상이한 시스템에서 서로 다른 형식 언어를 정의했을 경우에도 정보를 전달하게 하는 역할을 하며, 각 시스템에서 사용되어지는 상이한 ontology정보를 분석하여 자체 시스템에서 사용 가능한 형태로 바꾸어 주고, 필요한 메시지의 내용을 분석하여주는 역할을 수행한다.

접속된 에이전트들의 정보는 ANS에게 넘겨지고 ANS는 어떤 에이전트인지를 판단하여 그정보를 다시 OTS에게 전달하여 에이전트의 메시지를 분석한다.

3.2 통신 구조

에이전트가 서버에 접속하여 메시지를 주고 받을 경

우 에이전트를 등록, 삭제하고 메시지를 주고 받기 위하여 수행되어지는 모든 action들, 즉 프로그램에서 일어나는 행위에 대한 수행어의 포맷은 다음과 같다.

- Register(register-agent, password) : 에이전트를 등록할 때 사용된다.
  register-agent : 에이전트 이름, password : 비밀번호
- Whoiam(whoiam, content) : 에이전트에 대한 정보에 사용된다.
  whoiam : 에이전트 이름, content : 연결정보
- Reconnect-agent(reconnect-agent, port, agentname, password) : 재연결에 사용되어진다.
  reconnect-agent : host, port : 포트번호,
  agentname : 에이전트 이름, password : 비밀번호
- Disconnect-agent(reserver-message, content, port) : 에이전트의 접속해지에 사용되어진다.
  disconnect-agent : 대상 에이전트 이름
  content : 연결정보, port : 포트번호
- Reserve-message(reserve-message, content, port) : 메시지의 저장에 사용된다.
  reserve-message : sender 에이전트 이름,
  content : 연결정보, port : 포트번호
- Delete-message(delete-message, content) : 메시지 삭제에 사용되어진다.
  delete-message : sender 에이전트 이름,
  content : 연결 정보
- Request-agent(request-agent, content) : 재요청에 사용되어진다.
  request-agent : 에이전트 이름, content : 등록정보
- Unregister-agent(unregister-agent) : 에이전트 삭제에 사용되어진다.
  unregister-agent : 대상 에이전트 이름

서버에 새로운 에이전트가 도착하면 에이전트의 메시지 내용을 추출하여 분석한다. 간단한 분석을 통해 이 메시지의 목적을 알게된다. 이때 추출된 메시지는 메시지 전달 판정을 거쳐 정상유무를 판단하게 된다.

에이전트가 보내온 메시지의 내용을 분석하여 필요한 부분을 찾아 검색한다. 이때 각에이전트에서 보내온 수행어와 내용들의 정보가 전달되어 진다.

(그림 2)는 메시지를 전송하고 메시지 형태에 대한 부분이다. 에이전트가 메시지를 보내면 서버는 응답으로 명령어 사용의 판정과 에이전트의 정보를 다시 화면에 보여주게 된다.

```
/* 메시지를 전송하여 결과를 기다린다.*/
protected void sendResult(String reciever){
  String sendmsg = "(reply : sender :
  sendmsg = sendmsg + getName() + : receiver" + reveiver;
  sendmsg = sendmsg + "lang KQML : content" +
            Integer.toString(_returnVal) +")";
  try {
  sendMessage(senmsg);
  }
```

(그림 2) 메시지의 응답 전송

라우터는 통신상황을 모니터하며 메시지 전달을 조정한다. 라우터는 메시지를 전달할 대상이 연결되지 않았을 때 파일에 기록해 두었다가 에이전트가 연결되면 파일에 누적된 메시지를 차례로 전송한다. 메시지가 잘못 전달되었을 경우는 (그림 3)과 같이 에러메시지를 다시 보내주게 된다.

```
/* 에러 메시지 보내기 */
protected void senderErrorMessage(KQMLmessage kqml) {
  String receiver = kqml.getValue("sender");
  String sendmsg = "(error : sender ";
        sendmsg = sendmsg + getName() +" :receiver" + receiver;
      sendmsg = sendmsg +" :language KMQL : content
        (" + kqml.getSendString() + "))";
        try {
              sendMessage(sendmsg);
                  } catch (ConnectionException c) {
                        System.out.println(c.toString());
                  } catch (ParseException p) {
                        System.out.println(p.toString());
                  }
                  addToDeleteBuffer(0);
        }
```

(그림 3) 메시지의 에러검사와 형태

메시지가 전달되면 라우터에 (그림 4)와 같이 메시지 전달 성공에 대한 판정과 에이전트에 대한 정보가 표현되어 진다. 라우터 클라이언트는 사용자와의 인터페이스 부분으로 본 실험에서는 메시지가 전달되고 이 메시지를 검사한 결과에 대한 응답을 에이전트에게 보내주게 된다. (그림 5)는 라우터의 처리 모듈로 메시지 전송의 중간 역할을 수행하게 된다.

```
Start to register
N 4 Router register-agent 1999 02 : 15
  5 Router register-agent 1999 02 : 16
  6 Router register-agent 1999 02 : 17
Sending message
Sending success
Sending message
Sending success
decription : agent - info : password 111
port : -1
performative : agent - address
reciece : sky
message-method : MesssageRouter
host : null
sender : Router
Name: sky
```

(그림 4) 라우터메시지 전달 화면

```
package RouterLayer.AgentClient.Example.CalcServer;
import java.io.*;
import java.util.*;
import Abstract.*;
import KQMLLayer.*;
import RouterLayer.AgentClient.*;
        ......
public class CalcServer extends RouterClientAction {
    String _returnVal;
        ......
  public CalcServer(Address myaddress,
                    Address routeraddress,
                    Address registeraddress,
                    int durationtime)  {
        .....
}
```

(그림 5) 라우터 처리 모듈

3.3 메시지 검사

서버에 접속한 에이전트의 메시지를 검사하여 위반 여부를 체크한다. 등급에 벗어나거나 허용되지 않은 명령을 사용했을 때는 체크하여 제한하게 된다. 검사에 대한 구조는 if... then... else...구조로 되어있다. 이 구조에는 두 가지 조건이 포함된다. 첫째는 에이전트에 대한 등급이다. 에이전트에 대한 등급은 1, 2, 3, 4 등급으로 분류하여 최상위 '1등급'에서 최하위인 '4등급'으로 구분하였다. 이와 같은 분류작업은 등급에 따른 명령어를 제한하기 위한 작업이다. 또 하나는 명령어에 대해 각 등급마다 사용할 수 있는 명령어들의 집합을 정의하였다. 각 에이전트와 명령의 관계는 다음과 같이 표현된다.

만약 에이전트의 등급 구분이 다음과 같다면

$$A_1 > A_2 > A_3 > A_4$$

각 에이전트가 갖는 명령어 개수는 다음과 같은 관계를 가지게 된다.

$$n(A_1(C)) > n(A_2(C)) > n(A_3(C)) > n(A_4(C))$$

A : 에이전트 i : 에이전트 등급, C : 명령어들,
$n(A_i(C))$: 각 등급의 에이전트가 갖는 명령어 개수

위의 식은 에이전트의 등급이 높으면 높을수록 사용할 수 있는 명령어 수가 늘어난다는 뜻으로서 각 등급마다 명령어수가 제한되어 있다는 것을 알 수 있다. 위반으로 판정되었을 때는 "disconnect"와 "unregister"를 사용하여 연결을 끊거나 에이전트의 등록을 취소한다. 기본적인 명령어만 4등급으로 구분하여 보면 <표 3>과 같이 분류할 수 있다. 1등급은 모든 문서에 대한 권한을 전부 소유하고 있어서 모든 관련된 문서까지 사용할 수 있고 기록과 삭제를 할 수 있다. 2등급은 기록과 삭제는 할 수 없다. 3등급은 단순히 에이전트가 원하는 것이 존재하는지를 검사하고 대화를 요청할 수가 있으며, 마지막 4등급은 단순한 대화만을 요청한다.

〈표 3〉 등급에 따른 기본적인 명령어 분류

명령어	명령어 의미
ask-about	모든 관련 문서를 요청한다.
ask-one	하나의 질문에 대한 대답을 원한다.
ask-all	모든 질문에 관련된 대답을 원한다.
ask-if	문장이 있는가 없는가를 알기를 원한다.
register	기록을 요청한다.
unregister	기록을 거부한다.
delete	파일의 삭제를 요청한다.
forward	수행과정을 알기를 원한다.
rest	이전의 수행된 명령어를 요청한다.
stanby	명령에 대한 응답을 요구한다.
cat	파일의 내용을 요구한다.

에이전트 소유자의 직급별로 4등급으로 주어지고 각 직급에는 다시 4단계로 에이전트의 역할을 구분하여 에이전트 등록과 함께 주어지도록 되어 있다. 이러한 등급은 에이전트의 등록시 ANS에서 AgentClass와 Address를 검사하여 에이전트의 구성원의 직급과 역할에 따라 부여하게 된다. 예를들면, 한 조직내에서 부서장, 과장, 계장, 계원이 있다면 부서장의 관리 에이전트-현재 개발하고 있는 소프트웨어의 모든 document를 종합하는 agent-는 1등급, 과장의 검사 에이전트-하위 직원의 자료의 감독-는 2등급, 계장의 메일 에이전트-메일의 검사-는 3등급, 계원의 질의 에이전트-원하는 문제에 대한 대화 요청-로 4단계로 분류하여 시

스템에 등록하게 된다.

각 명령어에 대한 의미를 알아보면 <표 4>와 같다.

〈표 4〉 명령어와 의미

명령어 \ 등급	A_1 등급	A_2 등급	A_3 등급	A_4 등급
C_1	ask-about	ask-one	ask-if	reply
C_2	ask-one	ask-if	stanby	stanby
C_3	ask-all	stanby	rest	
C_4	ask-if	forward	cat	
C_5	unregister	rest		
C_6	register	cat		
C_7	delete			
C_8	forward			
C_9	rest			
C_{10}	stanby			
C_{11}	cat			

다음은 명령어를 검사하기 위한 검사 구조로 각 에이전트의 등급에 알맞은 명령어를 사용함을 검사하고 (그림 6)과 같이 수행되어 진다.

```
if(authority.equals("1등급")
 if(command.equals("ask-about")
 {_returnVal="사용가능합니다")
  ......
 smtp.to= agent.addr
   ......
  else{
 if (autority.equals("2등급")
  if(command.equals("register")
   ...
 if (autority.euqals("4등급")
  if(command.equals("delete")
{_returnVal="잘못된 명령입니다. 접속을 끊습니다."}
```

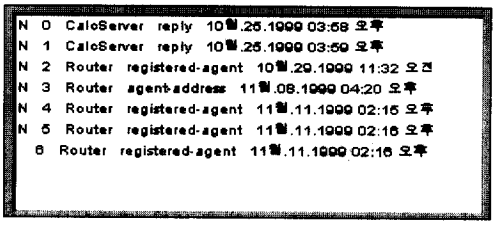
(그림 6) 검사 모듈 구조

4. 구 현

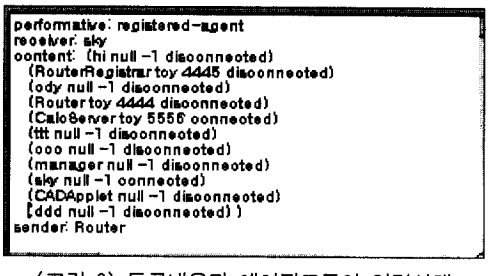
먼저 서버에 접속하기 위해서는 에이전트를 등록하여야 한다. 에이전트의 이름과 패스워드를 입력하고 Register버튼을 누르면 인터세서(intercessor)에 등록이 된다. 등록이 되면 (그림 7)과 같이 메시지를 보여주면서 에이전트의 등록 여부를 알려준다.

또한 (그림 8)은 등록에이전트에 대한 정보이다. 등록된 에이전트의 리스트를 보여주고 지금의 연결상태를 보여준다.

서버에 에이전트의 접속을 하기 위해서는 연결을 요청하여야 한다. 이미 등록되어 있는 에이전트는 에이전트 이름을, password 영역에 패스워드를 기입하고 Connect버튼을 누르면 인터세서(intercessor)와 연결이 되고 하단에는 연결메시지가 뜨게 된다. 만약 연결이 이루어지지 않으며 그에 해당하는 메시지가 나타나게 된다.

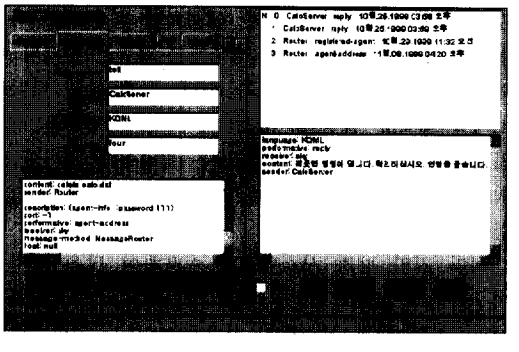


(그림 7) 에이전트의 등록 확인 화면



(그림 8) 등록내용과 에이전트들의 연결상태

(그림 9)는 메시지를 보내고 그 결과를 보여주는 것으로 화면의 좌측부분은 메시지를 보내는 에이전트에 대한 정보 즉, 에이전트의 이름, 대상 서버, 메시지 형태, 비밀번호가 나타나있다. 우측부분은 회신결과로서 (그림 10)과 같이 메시지에 대하여 처리된 날짜와 시간, 메시지의 처리를 보여준다. 이와 같이 등록된 에이전트의 메시지를 체크하여 에이전트에 대한 명령어를 검사하여 이상유무를 판정하여 보았다.



(그림 9) 메시지 보내기와 회신

N 0 CalcServer reply 10월 25 1999 03:58 오후
1 CalcServer reply 10월 25일 1999 03:59 오후
2 Router register-agent 10월 26 1999 11:32 오전
.....

language : KQML
performative : reply
receive : sky
content : 잘못된 명령어입니다.
sender : Calcserver

(그림 10) 검사화면의 내용

5. 결론 및 향후 과제

인터넷과 네트워크의 발달에 따라 컴퓨터의 사용이 증가함으로써 생활의 편리함이 제공되었지만, 이면에는 이로 인한 많은 문제점이 생겨나게 되었다. 컴퓨터와 네트워크를 통한 범죄행위가 사회의 큰 문제가 되고 있다. 이러한 위협에 대처하기 위한 정보보호를 필요로 하는 문서나 시스템에 대한 불법 침입을 분석하고 탐지하여 문제점을 사전에 방지하는 감사 기술의 발전적 형태인 침입 탐지 시스템(intrusion detection system)에 관한 연구가 활발히 진행되고 있다.

보안의 문제는 침입보다는 앞서갈 수는 없다. 하나의 침입에 대해 보안정책을 설정하고 또 다른 침입에 대해 보안은 따라갈 수 밖에 없는 입장이다. 보안은 단지 침입에 대한 효율성을 가장 큰 방패로 삼고 있다.

요즘은 심부름꾼이라고 불리는 에이전트에 대한 연구로 수많은 이동에이전트들이 네트워크를 사용자를 대신하여 돌아다니고 있다. 본 논문은 기존의 침입탐지시스템과는 달리 에이전트에 대한 침입을 정의하고 이에 대한 대처방안을 제시하고자 했다. 에이전트의 접속시 에이전트가 보내준 메시지를 검사하고 내용을 분석하여 침입에 대한 판단을 하였다. 침입에 대한 판단은 에이전트의 등록시 에이전트에 등급을 부여하고 각 등급에 알맞은 명령어를 제한하여 사용하게 하고 잘못된 명령어를 사용했을 경우에는 접속을 끊거나 에이전트의 등록을 해지하도록 했다. 이를 통해 바이러스처럼 해악을 끼칠 수 있는 에이전트의 침입을 정의하고 제어하여 보았고 보다 많은 명령어의 제어와 키워드 분석의 효율성을 높이기 위한 노력이 좀더 필요함을 알게 되었다.

여러 침입 탐지 시스템들의 이용하여 많은 탐지가 이루어지고 있다. 새로운 탐지 항목에 대한 지속적인 개발, 침입 탐지 시스템에 사용되는 보안 데이터베이스의 효율적인 관리, 그리고 특히 사용자 인터페이스 기능을 확장시켜 시스템 관리자가 시스템 운영이나 침입탐지에 있어 보안 규칙 설정, 시스템 및 데이터 베이스관리의 편이성 및 다양한 기능을 제공할 수 있도록 지속적인 연구가 진행되어야 한다. 또한 침입 탐지 시스템뿐만 아니라 침입자 역추적 시스템과의 연계에 노력하여 이 두 가지가 동시에 가능하게 함으로써 보안 기능을 좀더 강화하는 것이 필요하다.

참 고 문 헌

[1] 강현석, "하이브리드 에이전트 기반 침입 탐지 시스템의 메시지 처리 구조의 설계 및 구현", 한국외국어대학교 경영정보대학원, 1998.

[2] 김회준, "조정자 에이전트 기술을 이용한 침입탐지 시스템의 개발", 상명대학교 정보 통신대학원 정보통신학과 네트워크관리전공, 1998.

[3] 류경춘, "실시간 침입탐지 판정엔진 모듈에 관한 연구", 숭실대학교 대학원 전자계산학과, 1997.

[4] 정상수, "UNIX 환경에서 해킹방지를 위한 침입탐지 시스템의 설계에 관한 연구", 국방대학원 전자계산학과, 1995.

[5] 정현진, "네트워크 보안관리를 위한 이동에이전트 시스템의 설계와 구현", 홍익대학교 대학원 전자계산학과 프로그래밍 언어전공, 1998.

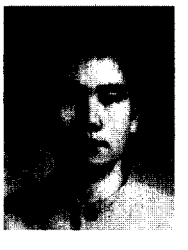
[6] 조경훈, "LAN 환경에서의 침입 탐지 시스템의 설계 및 구현", 충북대학교 대학원 전자계산학과, 1998.

[7] 한국정보보호센터, "시스템 보호 기술", http://www.kisa.or.kr.

[8] 한국정보보호센터, "정보시스템 해킹", http://www.kisa.or.kr.

[9] "대학원생 정보보호 기술교육 -시스템/네트워크 보안-", 한국정보보호센터, 1999.

[10] Mukherjee B., Herberline L. T. and Levitt K., "Network intrusion Detections," IEEE Network, May/June, 1994.



임 용 성

e-mail : lysung@jinri.kmu.ac.kr

1998년 계명대학교 수학과 졸업(학사)

2000년 계명대학교 컴퓨터공학과 졸업(석사)

2000년~현재 계명대학교 컴퓨터공학과 박사과정 재학 중

관심분야 : 보안, 침입탐지, 에이전트, 음성인식

장 덕 성

e-mail : dsjang@kmu.ac.kr

1979년 경북대학교 컴퓨터공학과 졸업(학사)

1981년 서울대학교 전산학과 (이학석사)

1988년 서울대학교 컴퓨터공학과 (공학박사)

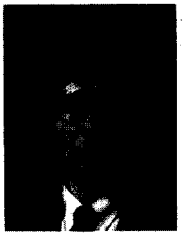
1982년~1985년 동아대학교 전산공학과 조교수

1985년~현재 계명대학교 컴퓨터공학과 교수

1992년~1993년 University of Coiorado 방문연구교수

1998년~현재 계명대학교 기획정보처 전산원장

관심분야 : 컴파일러, 시각프로그래밍, 자연어처리, 정보검색, 에이전트 등



정 홍

e-mail : jhong@kmu.ac.kr

1972년 한양대학교 원자력공학과 (학사)

1976년 고려대학교 경영대학원 (석사)

1996년 대구카톨릭대학교 전산통계학과(석사)

1999년 대구카톨릭대학교 전산통계학과(박사)

1972년~1981년 한국과학기술연구원 선임연구원

1981년~현재 계명대학교 컴퓨터공학과 부교수

관심분야 : 지능정보시스템, 소프트웨어공학