碩士學位論文

계층적인 에이전트 기반의 침입탐지 구조 설계

濟州大學校 大學院

電 氣 電 子 工 學 科

金 榮 均

2001年 12月

계층적인 에이전트 기반의 침입탐지 구조 설계

指導敎授 金 敬 植

金 榮 均

이 論文을 工學 碩士學位 論文으로 提出함

2001年 12月

金榮均의 工學 碩士學位 論文을 認准함.

審査委員長 _____印

委 員 _____印

委 員 _____印

濟州大學校 大學院

2001年 12月

Design of the Hierarchical Agent-based Intrusion Detection Architecture

Young-Gyun Kim

(Supervised by professor Kyung-Sik Kim)

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering

2001. 12.

Department of Electric and Electronic Engineering

GRADUATE SCHOOL

CHEJU NATIONAL UNIVERSITY

목 차

SUMMARY

Recently, the IDS(intrusion detection system) have received much attention from theoretical and practical point of view since information warfare attacks are more increased and sophisticated.

The new intrusion detection architecture which is composed of the multiple layers with hierarchically connected agents is described in this paper. In doing so, the configuration of agent's module suitable for hierarchical structure is employed, KQML(Knowledge Query and Manipulation Language) based-protocol is applied, and the procedure of operation in the architecture is explained.

To illustrate the operation performance of the proposed architecture, the prototype of suggested system is implemented. The prototype considered in this paper consists of Supervisor agent, Cooperator agent, MAN1 agent, MAN2 agent, MAN3 agent, SCAN agent, and SYN agent. The management systems are attacked by the scan and syn attack tools.

The efficient protocol design and the authentication of agents for the IDS would be main issues for the future work.

Ⅰ. 서론

인터넷 사용이 늘어남에 따라 보안에 대한 요구는 점점 증가하고 있다. 이러한 보안 위협과 공격들을 방어하기 위하여 적절한 보안 서비스가 수행되어야 한다. 보안 서비스에는 기밀성(confidentiality), 인증(authentication), 무결성(integrity), 부인봉쇄(nonrepudiation), 엑세스 제어(access control), 가용성(availability) 등(Stallings, 1997)이 있는데, 보안 관리 시스템(security management system)에서는 이러한 보안 서비스들을 적절하게 구현하게 된다. 이러한 보안 서비스들을 제공하기 위해서는 공격에 대한 빠르고 정확한 탐지 및 대응이 필요하게 되는데, 탐지 부분에서 침입탐지 시스템(IDS, Intrusion Detection System)이 이용된다.

침입탐지는 보안문제의 신호를 분석하기 위해 컴퓨터나 네트워크에서 발생하는 이벤트를 모니터하는 과정(Bace, 2000)이다. 침입탐지는 1980년경부터 미국의 System Design Laboratory, SRI international(SRI/SDL)에서 연구들을 수행하고 있다. 대표적인 연구로는 실시간으로 침입탐지 기능을 갖는 IDES(Intrusion Detection Expert System)(Javitz와 Valdes, 1991. Lunt등, 1992), 여러 호스트들을 대상으로 하는 침입탐지 필요성을 알리는 분산된 감사 기록 수집 메커니즘을 수행하는 NIEDS(Next-generation IDES)(Anderson등, 1995), NIDES를 대규모 전산망으로 확대하여 자동으로 침입탐지 결과를 보고하는 EMERALD(Event Monitoring Enabling Responses to anomalous Live Disturbances)(Porras와 Neumann, 1997), eBays, 활동 그래프를 이용한 시스템인 SRI의 UC Davis의 GrIDS(a Graph based Intrusion Detection System for large network)(Staniford-Chen등, 1996), Purdue Univ.의 IDIOT와 자율 에이전트를 이용한 AAFID(Autonomous Agent for Intrusion Detection)(Crosbie와 Spafford, 1995a, 1995b. Balasubramaniyan, 1998), 일본의 IPA(Information technology Promotion Agency)에서 개발한 IDA(Intrusion Detection Agent system)(Asaka등, 1999), Columbia 대학에서 수행하는 DARPA(Defence Advanced Research Projects Agency) 프로젝트 중의 하나인 JAM(Java Agent for

Meta-Learning)(Lee등, 1999), 미국의 NIST(National Institute of Standard and Technology)에서 수행중인 MAIDS(Mobile Agent Intrusion Detection and Security)(Slagell, 2001)프로젝트 등이 있다.

이러한 침입탐지 시스템에는 3가지 중요한 구성 요소인 정보원(information sources), 분석(analysis), 응답(response)이 있다. 분석은 오용 탐지(misuse detection)와 비정상상태 탐지(anomaly detection)로 나눌 수 있는데, 오용 탐지는 잘 알려진 공격의 패턴을 미리 가지고 있다가 패턴을 비교해서 공격을 파악하는 방법이다. 패턴을 비교해서 공격을 파악하므로 공격을 정확하게 탐지해서 매우 효과적이다. 또한, 시스템 관리자가 보안에 대해서 잘 알지 못하더라도 시스템의 보안 문제를 추적할 수 있도록 해주는 장점이 있다(Bace, 2000). 반면에, 알려진 공격만을 탐지할 수 있는 단점이 존재하므로, 새로운 공격의 데이터들을 지속적으로 갱신시켜야 한다. 비정상상태 탐지는 호스트나 네트워크의 비정상적인 행동을 탐지하는 방법이다. 공격에 대한 자세한 정보가 없어도 공격을 파악할 수 있고, 오용탐지 방법을 위해 공격에 대한 signature를 생성할 수 있다는 장점이 있으나, 사용자나 네트워크의 예측하지 못한 행동으로 인해 엄청난 수의 잘못된 탐지를 하게 된다. 즉 경고 오류(false alarm)를 남발하고, 구현 비용이 크다는 단점이 존재한다(Bace, 2000).

인공지능 분야에서 연구가 시작된 에이전트 기술은 에이전트를 이용함으로써, 네트워크 환경에 적응하고 자율적인 작업을 가능하게 한다. 계층화된 에이전트 기술은 에이전트의 하드웨어 의존성을 최소화시킴으로써 공격에 의해 에이전트가 동작을 멈추더라도 전체 시스템에는 최소한의 영향을 미친다. 이러한 기능을 침입탐지 시스템에 적용함으로써 침입탐지 시스템이 효율적으로 동작을 수행하도록 한다. 즉, 어떠한 공격에 의해 하나의 구성요소가 멈추게 되더라도, 전체 시스템은 동작을 계속하게 된다.

본 논문에서는 계층적인 에이전트 기술을 이용한 새로운 침입탐지 구조를 제안하였다. 구조를 계층적인 에이전트를 이용하여 설계하기 위해서, 각 계층의 에이전트 내부의 모듈들을 계층에 맞게 배치하였고, 각 에이전트의 특성을 구현하기 위해 에이전트 내부 모듈을 설계하였으며, KQML(Finin등 1994. Labriu와 Finin, 1997)기반의 간단한 프로토콜을 제시하였고, 상황에 따른 구조의 동작과정을 설명하였다.

제안된 시스템을 두 개의 네트워크 세그먼트와 다수의 관리대상 시스템으로 구성된 보안관리 영역에서 Supervisor agent, Cooperator agent와, Manager agent인 MAN1,

MAN2, MAN3 에이전트와 Tool agent인 SYN, SCAN 에이전트를 이용하여 구조를 생성하고, syn 공격과 scan 공격을 공개된 공격도구를 이용하여 수행하였고, Manager agent인 MAN2를 멈추게 하였으며, Supervisor agent 역시 동작을 멈추게 하였다. 이런 과정으로 시험을 한 결과, 정상적으로 공격을 탐지하고, 구조가 동작함을 확인하였다.

본 논문에서 제안한 새로운 침입탐지 구조는 관리 계층에서 하위 계층을 제어함으로써 하위 계층의 에이전트가 동작을 멈추게 되었을 때, 구조를 재조정해줌으로써 좀 더 능동적인 반응을 보이는 장점을 지녔으나, 계층적인 에이전트를 이용하기 때문에 분산 시스템에서 나타나는 지연시간 문제가 발생하는 단점이 존재한다.

본 논문의 구성을 살펴보면, Ⅱ장에서는 네트워크 보안의 개념과 네트워크 보안을 위한 침입탐지 시스템의 개념, 그리고 이러한 침입탐지 시스템에 적용되는 에이전트의 기술에 대해서 살펴보고, Ⅲ장에서는 기존 구조를 살펴보고, 제안하는 구조를 설계하였으며, Ⅳ장에서는 제안된 구조의 프로토타입 시스템을 구현해서 구현결과를 살펴보았으며, Ⅴ장에서 결론을 맺는다.

Ⅱ. 네트워크 보안

최근에 초고속 통신망이 각 가정에 많이 보급됨에 따라서 인터넷을 통한 네트워크의 사용이 증가하고 있는데, 네트워크 사용량이 증가함에 따라 네트워크 공격기법도 증가하고 있다. 인터넷과 관련된 수많은 어플리케이션의 발전과 더불어 그에 따르는 취약점 및 공격기법이 다양화되고 있다. 새로운 인터넷 매체는 악성 에이전트의 유포에 사용될 수 있으며, 매일 새롭게 등장하는 인터넷 어플리케이션들은 잠재적인 취약성이나 오류를 포함할 수 있다. 그리고 운영체제(OS, operating system), 시스템 또는 네트워크 프로토콜과 관련된 취약점보다 어플리케이션 계층에서의 취약점이 지속적으로 많이 발견되고 공격에 사용될 것이다. 이에 따라 보안에 대한 중요성이 더욱 강조되고 있으며, 새로운 공격기법에 대해서 유연하게 대응하고, 환경에 적응할 수 있는 보안관리 시스템이 필요하게 된다.(이, 2000b. 2001)

1. 기본 개념

데이터를 보호하고 해커를 막기 위한 도구의 집합을 총칭하여 컴퓨터 보안이라고 한다(Stallings, 1997). 이러한 보안을 위협하는 공격의 유형은 다음과 같다(Stallings, 1997).

- 방해(interruption) : 시스템의 일부가 파괴되거나 사용할 수 없게 되도록 하는 것이다.
- 가로채기(interception) : 비인가자들의 불법적인 접근에 의한 신뢰성에 대한 공격으로, 네트워크 상에서 패킷을 가로채거나, 파일과 프로그램의 불법 복사 등을 들 수 있다.
- 불법수정(modification) : 비인가자들의 불법적인 접근/변경에 의한 무결성에 대한

공격이다. 네트워크 상에서 패킷을 가로채기 한 후에, 패킷 내용을 수정하는 방법이다.

- 위조(fabrication) : 네트워크 상에 위조된 패킷을 삽입하거나 파일 등에 내용을 위조하는 방법이다.
- 위장(masquerade) : 비인가자가 인가자로 위장해서 공격하는 형태로 가장 일반적인 방법은 다른 사용자의 인증에 관한 정보를 도용하는 것이다.
- 재전송(replay) : 적법하게 사용되는 패킷을 불법적인 공격에 이용해서 재전송하는 방법이다.
- 서비스부인(denial of service) : 현재 수행되는 서비스를 정상적으로 사용되거나 관리되지 못하게 방해하는 방법이다. 이 공격에 의해서 네트워크 전체가 마비되는 경우도 발생할 수 있다.

한국 침해사고대응팀(CERT-kr)에서 구분하고 있는 유형으로 보면 침입시도, 불법침입, 자료유출, 자료 변조/삭제, 불법자원사용, 홈페이지변조, 시스템 파괴, 시스템 오류, 서비스 거부 등으로 공격유형을 나눌 수 있다. 이러한 공격 유형 말고도 새로운 공격 유형들이 많이 발견되고 있다(김, 2001a. 김등 2000. 홍과 고, 1998. 노등 2000). 이러한 보안 위협과 공격들을 방어하기 위해서는 적절한 보안 서비스가 수행되어야 한다. 보안 서비스에는 다음과 같은 것들이 있다.

- 기밀성 : 컴퓨터 시스템의 정보 및 전송 정보가 인가자만 읽을 수 있도록 해주는 서비스이다.
- 인증 : 메시지가 왔다고 주장하는 곳으로부터 전송되어 왔음을 수신자에게 확인시켜주거나, 시스템 내에 있는 자원을 접근하려는 요구에 대하여 사용자의 신분확인을 수행하는 서비스이다.
- 무결성 : 정보가 인가자에 의해서만 변경이 가능한 것이다. 즉, 허가 없이 변경이 되지 않도록 하는 서비스이다.
- 부인봉쇄 : 송신자나 수신자가 전송 메시지를 부인하지 못하도록 하는 서비스이다.
- 액세스 제어 : 사용자의 식별에 관한 서비스 및 허가된 사용자가 허가된 범위에서 정보나 자원에 접근할 수 있도록 허용하는 서비스이다.
- 가용성 : 허가된 인가자가 필요로 할 때, 컴퓨터 시스템 자원을 이용할 수 있게

하는 서비스이다.

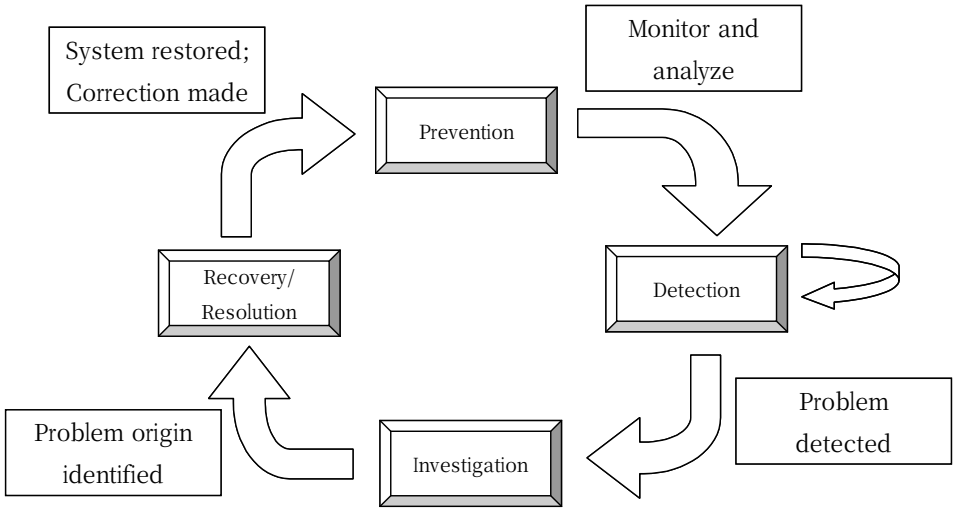

Fig. 1 A generic model of security management

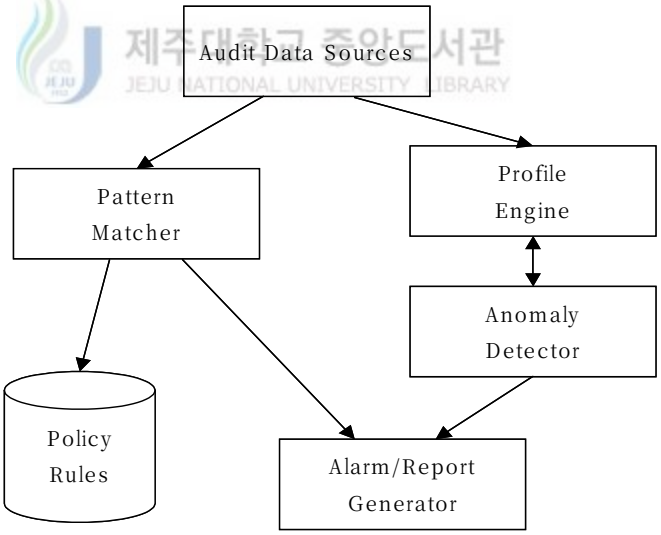

Fig. 2 A generic IDS

Fig. 1과 같은 보안 관리 시스템에서는 이러한 보안 서비스들을 적절하게 구현하게 된다. 예방을 하기 위해 모니터와 분석을 통한 탐지가 이루어져야 한다. 탐지 부분은 공격에 대한 빠르고 정확한 탐지 및 대응이 필요하게 되는데, 침입탐지 시스템이 이용된다. 탐지를 통하여 문제가 발견되면, 문제에 대해 조사를 하게 된다. 문제가 사실인

경우로 판명이 되면, 복구 절차를 통해서 문제를 해결하게 된다.

Fig. 2와 같은 일반적인 침입탐지 시스템은 audit 데이터들을 수집해서 공격패턴과의 일치여부를 확인하고, 프로파일 엔진(profile engine)을 통하여 비정상상태 탐지를 수행한다. 이 2가지 방법을 동시에 수행해서, 탐지가 이루어졌다고 판단이 되면, 탐지 경고를 생성하게 된다.

2. 침입탐지 시스템

침입탐지는 1980년경부터 미국에서 연구를 시작으로, System Design Laboratory의 SRI international에서 IDES, NIDS, EMERALD, eBays, Porras의 STAT, NCSA의 MIDAS, Purdue Univ.의 IDIOT와 AAFID, UC Davis의 GrIDS, IPA의 IDA, Columbia 대학의 JAM, NIST의 MAIDS, Los-Alamos National Lab.의 NADIR, ISS의 Real-Secure, AXENT의 Omniguard ITA, (주)인젠의 NeoWatcher등의 침입탐지 시스템들이 개발되었거나, 개발되고 있다.

1)침입탐지 시스템에 요구되는 특성

침입탐지 시스템은 다음과 같은 특성을 가지고 있어야 한다(Balasubramaniyan등 1998)

- 항상 동작 가능해야 한다.
- 시스템 파괴로부터 회복할 수 있어야 한다.
- 침입탐지 시스템 자신을 파괴하는 행위를 탐지해야 하고, 대응할 수 있어야 한다.
- 최소한의 오버헤드를 가져야 하며,
- 시스템의 보안 정책에 따라 구성되어야 한다
- 시스템 환경의 변화에 적응해야 한다.

2)침입탐지 시스템의 구성 요소

침입탐지 시스템은 다음과 같은 3가지 중요한 구성 요소로 구성된다.

· 정보원 : 침입이 발생했는지를 판단하는 이벤트 정보는 여러 가지 소스(source)로부터 온다. 소스들은 네트워크, 호스트, 어플리케이션 모니터링으로부터 수집된다.
· 분석 : 정보원을 분석하여 침입이 발생하고 있는지, 발생했었는지를 파악하는 부분이다. 분석 시도는 오용 탐지와 비정상상태 탐지로 나눌 수 있다.
· 응답 : 침입을 탐지하면 어떤 활동을 하게 된다. 여기에는 능동적(active) 방법과 수동적(passive) 방법이 있다. 능동적 방법은 침입탐지 시스템이 자동적으로 시스템에 개입을 하는 것이고, 수동적 방법은 관리자에게 탐지보고를 하는 것이다.

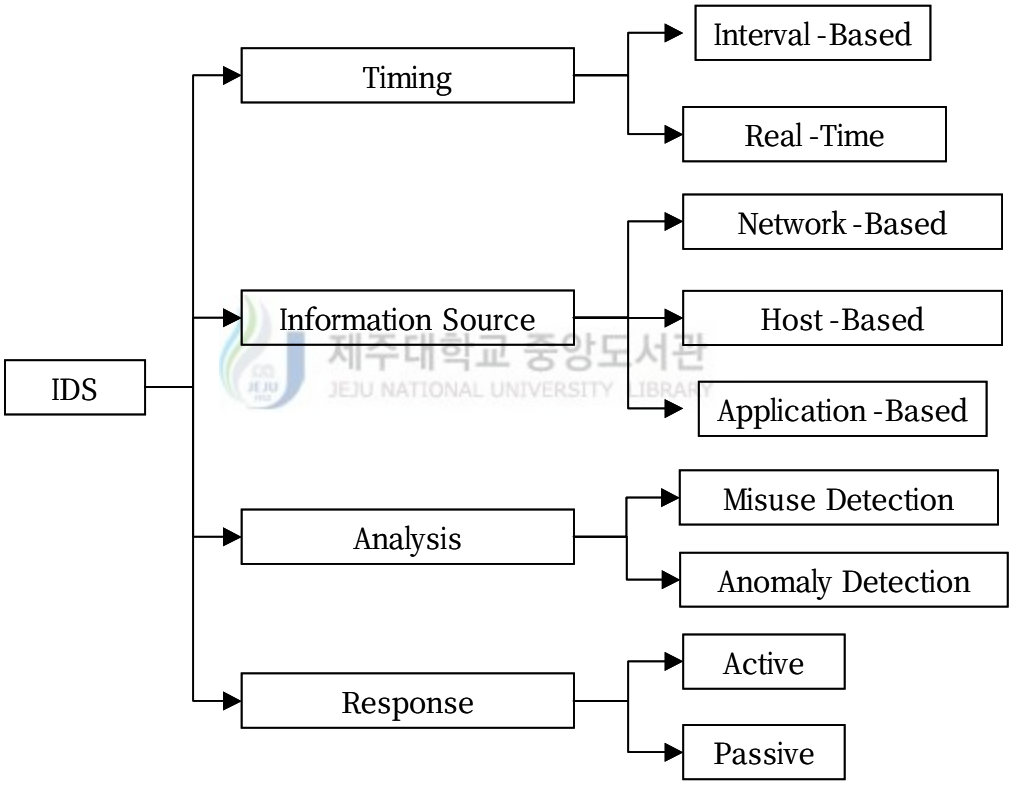

Fig. 3 Classification of IDS

3)침입탐지 시스템의 분류

침입탐지 시스템은 Fig. 3같이 분류될 수 있다.

(1)타이밍(Timing)에 따른 분류

①인터벌 기반의(Interval based) 침입탐지 시스템

이 침입탐지 시스템은 정보를 실시간으로 수집하지 않는다. 예전의 많은 호스트 기반의 침입탐지 시스템들이 이 방식을 이용하였다. 호스트 기반 침입탐지 시스템은 파일과 같은 시스템의 audit trails에 의존한다. 이 침입탐지 시스템은 능동적 응답을 수행할 수 없다(Bace, 2000).

②실시간(Real-time) 침입탐지 시스템

이 침입탐지 시스템은 정보를 실시간으로 수집한다. 네트워크 기반의 침입탐지 시스템에서 이 방식을 사용한다. 실시간 침입탐지 시스템에서는 공격을 탐지하자마자 빠른 시간 안에 보고를 할 수 있어야 한다.

(2)정보원에 따른 분류

①네트워크 기반의(Network based) 침입탐지 시스템

이 침입탐지 시스템은 네트워크 패킷을 수집하고 분석함으로써 공격을 탐지한다. 하나의 네트워크 세그먼트 하에서 하나의 네트워크 기반의 침입탐지 시스템이 네트워크 세그먼트에 연결된 다수의 호스트들을 보호하기 위하여 모니터링 한다. 이 침입탐지 시스템은 커다란 네트워크를 소수의 침입탐지 시스템들로 모니터할 수 있고, 침입탐지 시스템들은 네트워크에 작은 영향만을 준다. 그리고 많은 보이지 않는 공격에 대해 높은 보안을 유지시켜 주는 장점을 가지고 있으나, 트래픽이 많은 네트워크에서는 모든 패킷을 수집하는 것이 어렵고, 사소한 패킷을 놓치게 되면, 공격을 파악하지 못 할 수도 있게 된다. 또한, 스위치 기반의 네트워크에서는 적용을 하기 힘들며, 패킷이 암호화되면 패킷을 분석할 수 없어 공격을 탐지할 수 없게 된다. 그리고, 침입탐지 시스템은 공격이 성공했는지는 알 수가 없고, 공격의 초기 단계만을 탐지할 수 있다(Bace, 2000). 또한, 몇몇 침입탐지 시스템들은 단편화(fragment)를 이용한 공격(정, 2001. 김, 2001b)을 탐지할 수가 없다는 단점이 존재한다. 그리고, 단편화를 이용한 공격을 탐지하더라도 침입탐지 시스템과 네트워크의 성능을 저하시켜 사실상, 네트워크 기반의 침입탐지 시스템에서는 탐지가 불가능하다고 말할 수 있다.

②호스트 기반의(Host based) 침입탐지 시스템

이 침입탐지 시스템들은 컴퓨터 시스템 내에서 정보를 수집한다. 호스트 기반의 침입탐지 시스템들은 보통 운영체제의 audit trails와 시스템 로그들(system logs)을 정보원으로 이용한다. 운영체제의 audit trails는 운영체제의 커널 레벨에서 항상 생성된다. 그리고 시스템 로그들보다 자세하고 보호를 잘 받는다. 시스템 로그들은 audit trails보

다 훨씬 작고 이해하기가 쉽다. 이 침입탐지 시스템들은 호스트 내부에서 이벤트를 모니터링 할 수 있기 때문에 네트워크 기반의 침입탐지 시스템에서 알 수 없는 공격을 탐지할 수 있고, 네트워크 패킷이 암호화되더라도 동작을 수행할 수 있다. 스위치 기반의 네트워크에 영향을 받지 않으며, 트로이 목마를 탐지할 수 있고, 단편화를 이용한 공격을 쉽게 탐지가 가능한 장점이 있다(Bace, 2000). 그러나, 각각의 호스트들을 모니터링 하도록 구성되고 관리되므로, 정보를 관리하는 것이 어렵고, 호스트를 공격하는 공격에 의해서 기능을 상실할 수도 있으며, 자신에게 오는 패킷만을 받으므로 네트워크 스캔을 탐지하는데 적합하지 않다. 또한, DoS(Denial-of-Service) 공격에 의해 기능을 상실할 수도 있다. 운영체제의 audit trails를 사용하므로, 정보를 저장할 시스템의 저장공간을 요구하게 되며, 호스트의 자원을 사용하므로 호스트 성능에 영향을 미치는 단점이 존재한다(Bace, 2000).

③응용프로그램 기반의(Application based) 침입탐지 시스템

이 침입탐지 시스템은 응용프로그램에서 이벤트 발생을 분석하는 호스트 기반의 침입탐지 시스템의 특수한 형태이다. 이 침입탐지 시스템에서 사용하는 대부분의 정보원들은 응용프로그램 처리 로그 파일들이다. 이 침입탐지 시스템은 권한이 있는 사용자가 그들의 권한을 높이는 행위 같은 의심스러운 행동을 탐지한다. 권한이 없는 활동을 하는 사용자를 추적하는데 사용되는 응용프로그램과 사용자간의 상호작용을 모니터링 할 수 있고, 암호화된 환경에서도 사용할 수 있다는 장점이 있다(Bace, 2000). 그러나, 응용프로그램 로그가 잘 보호되지 않음으로써 많은 취약점을 가지고 있고, 트로이 목마나 소프트웨어 조작 공격을 잘 파악하지 못하는 단점이 존재한다(Bace, 2000).

(3)분석에 따른 분류

①오용 탐지

잘 알려진 공격의 패턴을 미리 가지고 있다가 패턴을 비교해서 공격을 파악하는 방법이다. 알려진 공격과 관련된 패턴을 signature라고 하는데, 오용 탐지는 때때로 signature 기반의 탐지라고도 부른다. 이 방법은 패턴을 비교하여 공격을 파악하므로 공격을 정확하게 탐지해서 매우 효과적이다. 또한, 시스템 관리자가 보안에 대해서 잘 알지 못하더라도 시스템의 보안 문제를 추적할 수 있도록 해주는 장점이 있다. 반면에, 알려진 공격만을 탐지할 수 있다는 단점이 존재하므로, 새로운 공격의 signature들을 지속적으로 갱신시켜야만 한다(Bace, 2000).

②비정상상태 탐지

호스트나 네트워크의 비정상적인 행동을 탐지하는 방법이다. 이 탐지방법은 호스트나 사용자나 네트워크 접속의 정상 행동을 프로파일로 구축하고 있다. 이 프로파일들은 정상적인 작동을 하는 범위내의 데이터들로 구성되어 있다. 이 비정상상태 탐지에 사용되는 방법이나 기술들은 다음과 같다.

- 문턱(threshold) 탐지 : 사용자의 어떤 특성이나, 시스템의 행동이 수로써 표현되는 방법이다. 어떤 특성들은 주어진 시간에 사용자에 의해서 억세스 되는 파일의 수, 시스템에 접속 시도의 실패한 수, 프로세스에 의한 중앙처리장치(CPU) 사용률을 포함한다.
- 통계적인 방법들 : 주기적인(프로파일된 특성의 분배가 특정한 패턴에 맞는)것과 비주기적인(프로파일된 특성의 분배가 시간에서 관측되거나, 과거의 수치들의 집합으로 된)것이 속한다.
- 규칙 기반 방법 : 관측된 데이터가 받아들일 수 있는 취급 패턴으로 정의되는 비주기적인 통계적인 방법과 비슷하다. 수나 양이 아니라, 규칙으로써 특별한 패턴과는 다르다.
- 신경망, 유전 알고리즘, 면역 시스템 모델을 포함하는 방법들

이 방법은 공격에 대한 자세한 정보가 없어도 공격을 파악할 수 있고, 오용 탐지 방법을 위해서 공격에 대한 signature를 생성할 수 있다는 장점이 있고, 사용자나 네트워크의 예측하지 못한 행동때문에, 엄청난 수의 잘못된 탐지를 하게 된다. 즉, 경고 오류를 남발하고, 구현 비용이 크다는 단점이 존재한다(Bace, 2000).

(4)응답에 따른 분류

①능동적 응답

이 방법은 크게 3가지 종류로 다음과 같이 나눌 수 있다.

- 의심되는 공격에 대해서 추가적인 정보를 수집한다.
- 환경을 수정한다. 침입탐지 시스템에서는 공격을 차단하기 위해서 여러 가지 방법을 쓰게 된다. 첫 번째, TCP(Transmission Control Protocol) reset 패킷을 공격자의 접속에 끼어 넣어서 접속을 끊어지게 만든다. 두 번째, 라우터나 방화벽을 재구성하여 공격자의 패킷을 통과시키지 않는다. 세 번째, 공격자에 의하여 사용되는 서비스나 프로토콜, 네트워크 포트를 라우터나 방화벽에서 차단 시키도록 재

구성한다. 네 번째, 최악의 경우에는 어떤 네트워크 인터페이스에서 사용되는 모든 접속을 차단시킨다.

- 침입자에 대항하는 행동을 수행한다. 공격자의 호스트나 사이트에 대해 정보를 얻는 것을 시도하거나 공격하도록 하는 방법이다. 그러나, 공격을 하는 것은 좋은 방법이 안되고, 공격자들은 대부분 IP(Internet Protocol) 속이기(spoofing) 방법을 사용해서 공격을 하므로 정보를 알아내는 것은 힘들다.

②수동적 응답

이 방법은 크게 2가지 종류로 다음과 같이 나눌 수 있다.

- 공격을 탐지한 것을 관리자에게 보고하는 방법.
- 몇몇 침입탐지 시스템은 네트워크 관리 시스템(NMS, Network Management System)의 일부 기능으로써 구현되는 경우가 있는데, 이러한 시스템들은 탐지를 보고하는데, 간이 네트워크 관리 프로토콜(SNMP, Simple Network Management Protocol)을 이용해서 보고하는 방법이다.

이러한 침입탐지 시스템은 잘 가공된 보안 관련 감사 기록 데이터를 요구하는데 이러한 것을 만족시키기 위해서 보안 관련 감사 기록 데이터의 수집 기술, 추후 감사를 위한 데이터의 저장기술, 보안 관련 감사 기록 데이터의 분석 및 해석 기술, 사용자에 대한 쉬운 인터페이스 제공 기술이 적용되어야 한다.

3. 에이전트를 이용한 침입탐지 시스템

1)에이전트의 개념

에이전트는 지속적으로 환경에서 지각된 것과 내부 지식을 바탕으로 추론하고, 행동하며, 환경에 영향을 미치고 또한 사용자를 포함한 다른 에이전트와 의사 소통하는 지속적으로 존재하는 소프트웨어이다(이, 2000a). 에이전트 연구는 인공지능 분야에서 파생된 분야인데, 인공지능 분야에서 분리되어 독립적인 연구 주제로 연구되기 시작한 것은 분산 협동 처리(distributed cooperative processing)와 에이전트간 통신(inter agent communication)의 개념이 대두되면서 부터이다(최, 1997). 분산 처리를 하기 위

해서는 에이전트간의 통신이 필수적인데, 에이전트간의 통신의 목적은 작업을 처리하기 위해서 다른 에이전트에게 도움을 요청하거나 정보를 요구하는 과정이 필요한데, 이러한 과정에 에이전트간의 통신이 이용된다. 이러한 에이전트간의 통신에 가장 큰 과제는 에이전트간의 이형질성(heterogeneity)이다. 이형질성이란 에이전트가 서로 다른 사람들에 의해서 다른 시기에, 다른 플랫폼을 이용해서 개발되었기 때문에 이들간의 통신을 위해서는 상호 이해 가능한 언어와 프로토콜이 필요하다. 이러한 문제를 해결하기 위해서 사용되는 통신 프로토콜에는 KQML, 미 국방성의 ACL(Agent Communication Language), SRI의 ICL(Inter-agent Communication Language)등이 있다(최, 1997).

에이전트의 특성을 살펴보게 되면 기본적으로 몇 가지 특성을 가지고 있는데, 대표적으로 자율성(autonomy), 지능성(intelligence), 이동성(mobility), 사교성(social ability) 등을 들 수 있다. 자율성이란 외부의 지시나 도움 없이 에이전트 스스로 환경에 적응해서 능동적으로 작업을 수행하게 되는 것을 말한다. 지능이란 지식 베이스와 추론 기능을 갖추고 사용자의 의도를 파악해서 계획을 세우고 학습을 통해서 새로운 지식을 스스로 터득하는 성질로 인공지능분야에서 파생되어 나왔다는 것을 보여주는 부분이다. 이동성은 클라이언트가 필요로 하는 작업을 하기 위해서 에이전트를 서버로 보내어 수행을 시키는 것을 말한다. 즉, 기존의 클라이언트-서버의 개념과는 다르다. 사교성은 에이전트간 통신을 할 수 있다는 것을 의미한다. 에이전트가 혼자서 작업을 하지 못할 때, 다른 에이전트에게 도움을 청해 서로 협동해서 일을 처리할 수 있게 통신을 주고받는 것을 뜻하는 것이다. 이러한 기본적인 성질 외에도 많은 특성들이 있다. 에이전트는 위의 특성 중에서 어떤 특성이 두드러지는가에 따라서 이동 에이전트, 자율에이전트, 협동 에이전트 등의 이름으로 불리게 된다.

2)자율 에이전트

자율에이전트는 호스트에서 기능을 수행하는 소프트웨어이다. 독립적으로 동작을 하게 된다. 이 자율에이전트를 이용한 침입탐지 시스템은 다음과 같은 장점을 가지게 된다(Balasubramaniyan등 1998).

- 에이전트가 독립적으로 동작하므로, 에이전트가 어떤 이유로 동작을 멈추게 되었을 때, 다른 에이전트에 영향을 미치지 않는다.

· 다중 레벨을 가지고 있기 때문에 확장이 용이하다.

· 에이전트가 현재 구동되고 있는 호스트와 관련된 네트워크 정보를 모을 수 있다면, 침입탐지 시스템을 우회하는 공격을 감소시킬 수 있다.

이러한 장점을 가지고 있는 반면에, 계층 구조를 가지고 있음으로써, 필요한 정보를 모으는데 지연시간이 걸리는 단점이 존재한다. 주요 연구로는 Purdue university의 COAST lab.(COAST)의 AAFID가 있다.

3) 이동 에이전트

이동 에이전트는 에이전트에 이동성을 부여한 것으로, 인터넷을 비롯한 네트워크 환경에서 일정한 임무를 부여받아 자율적으로 시스템 사이를 이동하면서 필요한 정보의 수집 및 처리를 수행하는 실행 가능한 프로그램이다(방, 2000). 이동에이전트는 다음과 같은 장점을 가지고 있다(방, 2000).

· 네트워크 지연의 극복

· 네트워크 부하의 감소

· 비동기적인 실행과 자율성

· 구조 및 조합의 편이성

· 동적인 적응력

· 이질화된 환경에서의 운용가능

· 고장방지능력

· 확장용이성

이동 에이전트를 이용한 침입탐지 시스템에서는 관리대상 시스템으로 이동 후 임무를 수행하므로, 네트워크 대역폭의 효율적인 사용이 가능하다. 탐지에 관련된 규칙에 대한 관리를 중앙의 시스템에서 수행하게 되므로 새로운 공격기법의 등장에 대해 유연성을 부여할 수 있게 된다. 주요 연구성과로는 일본의 IPA에서 개발한 IDA, Columbia 대학에서 수행하는 DARPA 프로젝트 중의 하나인 JAM, 미국의 NIST에서 수행중인 MAIDS 프로젝트 등이 있다

Ⅲ. 침입탐지 구조 설계

제안된 침입탐지 구조는 관리 계층인 Supervisor agent, 중간 계층인 Manager agent, 탐지를 수행하는 Tool agent, 기능 백업 영역인 Cooperator agent로 구성되어 있다. 이 에이전트들은 자율성, 사교성, 지능성의 특성에 맞게 설계되었으며, KQML에 기반한 프로토콜을 통해 통신을 한다.

1. 기존 구조

본 논문은 자율 에이전트를 이용한 연구인 AAFID와 AHA! IDS(Adaptive Hierarchical Agent-based Intrusion Detection System)(Ragsdale, 2000)를 기반으로 이루어졌다.

1)AAFID

AAFID는 에이전트(agents), 트랜시버(transceivers), 모니터(monitors)라는 3개의 구성요소로 이루어졌다. 에이전트는 자율 에이전트이며, 침입탐지 시스템의 기능을 가지고 있다. 트랜시버는 에이전트들을 관리하는데 하나의 호스트에 한 개만 존재한다. 에이전트들을 관리하며, 트랜시버는 에이전트를 동작시키거나, 멈추게 할 수 있다. 트랜시버는 에이전트로부터 데이터를 받아들여서 정리를 하게 된다. 트랜시버는 호스트의 외부로 통신을 할 수 있다. 트랜시버는 정리한 데이터를 하나나 그 이상의 모니터에게 보고를 한다. 각각의 모니터는 여러 개의 트랜시버를 가지고 있다. 모니터들은 여러 계층으로 이루어져 있다. 이 구조의 단점은 모니터가 외부의 공격이나 어떠한 이유로 인해 동작을 멈추게 되면, 그 모니터가 관리하고 있는 트랜시버들의 정보는 사라지게 된다. 그리고 이 문제를 해결하기 위해 여러 개의 모니터가 트랜시버의 정보를 가질

수 있게 한다면, 정보를 중복해서 처리하게 되는 단점이 발생한다.

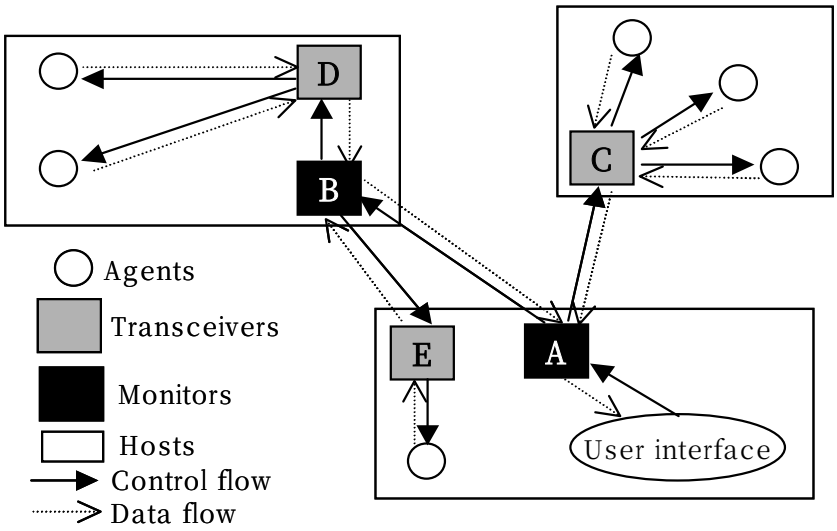

Fig. 4 Physical layout of the components in a sample AAFID system.

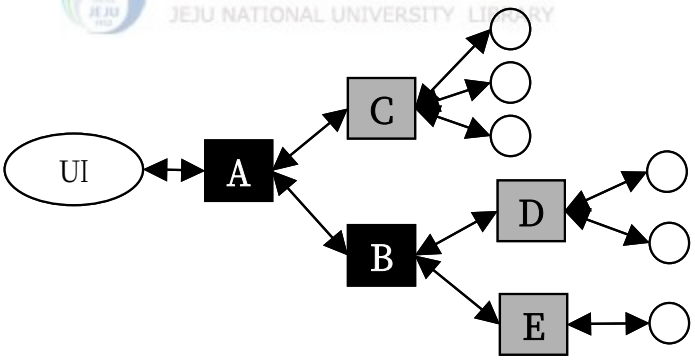

Fig. 5 Logic organization of Fig. 4. showing the communication hierarchy of the components.

Fig. 4는 간단한 AAFID의 물리적인 구조를 보여준다. 트랜시버 D는 에이전트 2개, 트랜시버 C는 에이전트 3개, 트랜시버 E는 에이전트 1개를 관리한다. 모니터 B는 트랜시버 D와 E를 관리하고, 모니터 A는 모니터 B와 트랜시버 C를 관리하고 유저 인터페이스를 제공한다. Fig. 4에서 보이는 실선은 유저 인터페이스에서 명령을 내리면, 계층을 따라서 에이전트까지 명령이 내려가는 것을 나타내며, 점선은 에이전트에서 유저

인터페이스로 탐지 보고와 같은 보고의 흐름을 나타낸다. Fig. 4를 논리적으로 재구성하게 되면 Fig. 5와 같이 나타나게 된다.

2)AHA! IDS

AHA! IDS은 AAFID를 기반으로 이루어진 연구인데, Fig. 6에서 보여주는 바와 같이 계층구조를 갖는 Director 에이전트, Manager agent, Tool agent, Surrogate 에이전트들로 이루어졌다. 대규모 네트워크에서 Director 에이전트들은 여러 계층이 존재할 수 있다. 각 Director 에이전트들은 Surrogate 에이전트와 연결되어 있다. 각각 연결을 함으로써 Director가 동작을 하지 못하게 됐을 때, Surrogate 에이전트가 그 역할을 대신하게 된다. 가장 낮은 계층의 Director 에이전트는 Manager agent의 상위 계층인데, Manager agent는 Director 에이전트가 관리하는 영역을 서로 겹쳐서 관리를 하게 된다. Manager agent는 Tool agent를 관리하는데, 이 Tool agent는 다양한 침입탐지 시스템의 기능을 가지고 있다.

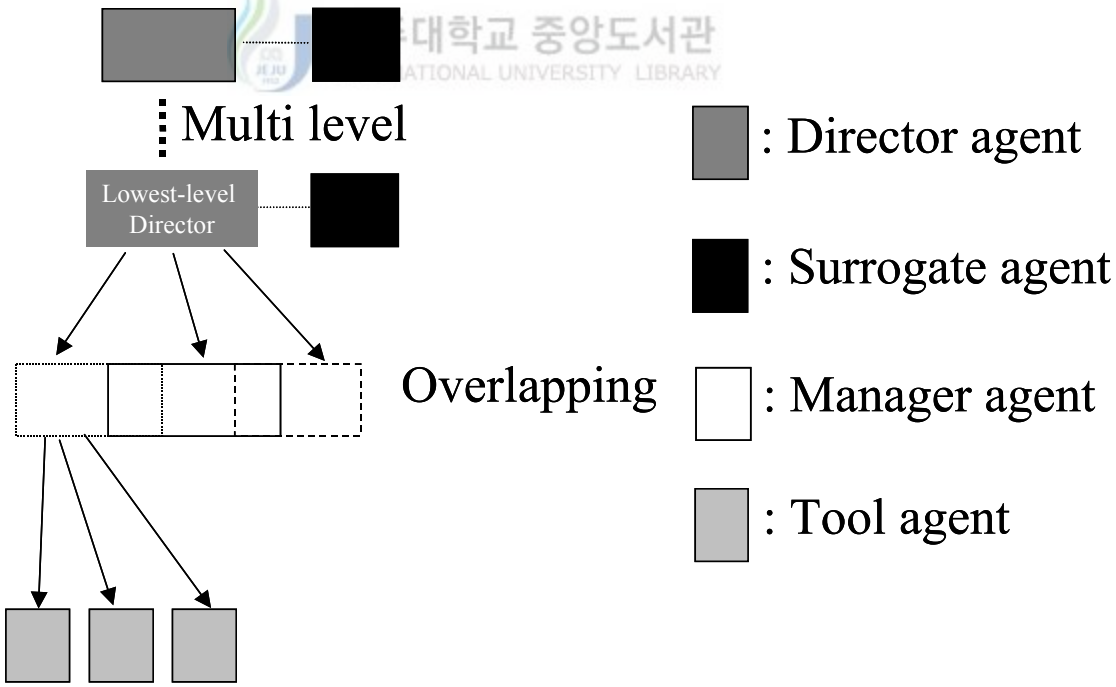

Fig. 6 Architecture of AHA! IDS

AHA! IDS의 단점은 Director 에이전트마다 Surrogate 에이전트가 모두 연결되어

있다는 것이다. Surrogate 에이전트는 Director 에이전트가 동작을 하고 있을 때는 쓸모 없는 에이전트이다. 그러면서 하드웨어적으로 Director 에이전트와는 독립적으로 존재해야 한다. 즉, 하드웨어의 비용이 증가하게 된다. 이에 본 논문에서는 사용자와의 인터페이스와 구조를 조정하는 기능을 가진 Supervisor agent는 하나만 존재하고 중간의 정보의 취합이나 하위 계층을 관리하는 기능을 가진 Manger 에이전트는 여러 계층으로 이루어지게 하고, Manager agent가 외부의 공격으로 인해서 동작을 못하게 됐을 때, Supervisor agent에서 구조를 재조정 해줌으로써 최소한의 하드웨어 비용을 가지고 구조를 유지할 수 있도록 하였다.

2. 제안된 침입탐지 구조

HAID(the Hierarchical Agent-based Intrusion Detection architecture)라고 명명된 제안 침입탐지 구조는 Fig. 7과 같이 나타낼 수 있으며, 주요 구성 요소는 Supervisor agent, Manager agent, Tool agent, Cooperator agent이다.

Supervisor agent는 사용자를 위한 UI(User Interface)를 담당하고, 하위 계층들을 관리, 구조의 재구성을 담당하는 에이전트이다. 하위 계층의 에이전트가 어떤 이유로 동작을 멈추게 되면, Supervisor agent에서는 그 에이전트가 중지되었다고 운영자에게 알려줌과 동시에, 다른 에이전트로 하여금 그 기능을 대신하도록 구조를 재조정한다.

Manager agent는 Tool agent를 관리하며, 여러 계층으로 이루어져 있고, 영역을 관리하며, Tool agent가 보내는 정보를 모아서 분석하는 역할을 담당한다. 자신이 맡은 영역을 관리하고 하위 계층에서 정보를 받아서 정보를 분석함으로써 이 정보가 침입인지 아닌지를 분석해서 상위 계층으로 분석 보고서를 보내주게 된다. 최상위 계층의 Manager agent는 Supervisor agent로 분석 보고서를 보내서 침입탐지 여부를 알려준다.

최하위 계층의 Manager agent는 Tool agent를 관리하게 된다. Tool agent는 침입탐지를 수행하고 침입탐지가 이루어지면 탐지 보고서를 Manager agent로 보내주게 된다. 이 때, Manager agent는 Tool agent로부터 보고서를 받아서 탐지신뢰 매트릭스를

구성하게 된다. 이 탐지신뢰 매트릭스를 참조해서 현재 보고된 침입탐지 상황이 어느 정도의 신뢰성이 있는지 파악하게 된다. 탐지신뢰 매트릭스는 침입탐지 보고서가 사실인지 아닌지에 따라서 공격 유형별로 탐지신뢰 매트릭스를 재구성하게 된다. Manager agent들은 자기 영역을 관리하다가, 다른 Manager agent의 요청이 들어오면 협동해 일을 처리한다.

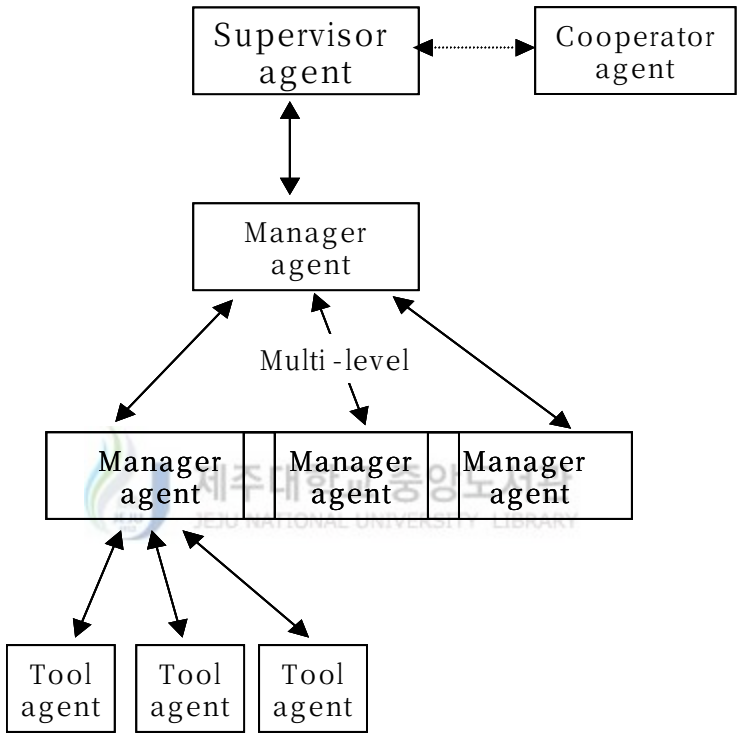

Fig. 7 Suggested intrusion detection architecture

이 구조는 침입탐지 시스템과 방화벽의 차단기능을 모두 수용함으로써 최소한 하나 이상의 Manager agent가 방화벽의 차단기능을 수행하는 Tool agent들을 관리해야 한다. 침입탐지 시스템도 호스트 기반의 침입탐지 시스템과 네트워크 기반의 침입탐지 시스템을 모두 포함할 수 있으므로, 호스트 기반 침입탐지 시스템의 정보를 이용해서 네트워크 기반 침입탐지 시스템 우회공격을 감소시킬 수 있다.

Cooperator agent는 Supervisor agent가 동작을 멈추었을 때, 동작하는 에이전트이다. Cooperator agent는 Supervisor agent와 통신을 유지하면서 기능을 중단하고 대기하다가 어떤 이유로 Supervisor agent가 기능을 수행하지 못하게 되면, 동작을 하게 된다.

1)모듈 설계

(1)구조를 유지하기 위한 에이전트의 특성 정의

에이전트에는 많은 특성들이 있지만 HAID에 고려되어진 에이전트는 자율성(autonomous), 사교성(social ability), 지능성(intelligence)의 특성을 고려해 구성되었다.

본 논문에서 자율성이란 환경 변화에 따라 기능을 수행해야 하는 것을 말한다. 에이전트가 위치하고 있는 시스템의 환경을 파악하는 것이 필요하고, 환경이 허락하면 기능을 수행해야 한다. 에이전트가 위치한 시스템의 자원을 파악하고, 현재 에이전트가 점유하는 자원을 파악하고 다른 모듈이 원하면 제공해 주어야 한다. 자원의 양을 파악해 기능을 수행할 수 있다고 판단이 되면 에이전트에서 필요한 여러 가지 모듈들을 실행하게 되고, 모듈이 필요 없다고 판단되면 해당 모듈을 정지시키는 동적인 모듈 관리를 맡는 관리가 이루어져야 한다. 또한, HAID의 한 에이전트가 외부의 공격이나 다른 이유로 동작을 멈추었을 때, HAID는 계속해서 동작해야 한다.

사교성이란 에이전트간의 통신이 가능한 것을 말한다. 따라서, 통신을 할 수 있는 모듈이 필요하게 되는데, 유의할 사항은 각 에이전트는 다른 에이전트가 어떤 기능을 가지고 있는지 알 수가 없다. 이러한 점을 극복하기 위해 멀티 에이전트에서 에이전트 사이의 통신을 하게 될 때 쓰이게 되는 조정 에이전트 역할을 할 수 있는 모듈이 필요하게 된다. 이 모듈은 하위 계층의 에이전트가 실행될 때 보고하는 메타 지식(meta knowledge)을 보유하게 된다. 메타지식은 에이전트의 성질이나 여러 에이전트간의 상호작용을 위한 제어 지식이다(최등, 1996). 메타 지식을 보유함으로써 하위 계층의 에이전트가 다른 에이전트의 기능을 이용하거나 다른 에이전트와 협동해서 어떤 일을 처리할 때, 각 에이전트의 통신을 중재하는 역할을 하게 된다. 즉, 하위 계층의 에이전트 B는 상위 계층의 에이전트 A로 자신이 필요로 하는 기능을 요청하게 된다. 상위 계층의 에이전트 A는 메타 지식을 검색해서 기능이 등록되어 있으면, 에이전트 C로 그 기능을 요청해 회답을 하위 계층의 에이전트 B로 보내게 된다. 만약 메타지식이 없게되면, 다시 자신의 상위 계층의 에이전트 D로 기능을 요청함으로써 기능을 수행하는 에이전트 E를 찾게 된다. 에이전트간의 통신을 이용하므로 구조의 확장이나 에이전트의 이질성을 극복하기 위해서는 KQML, XML(eXtensible Markup Language) 같은 표준화된 프로토콜을 이용해야 한다.

지능성이란 구조에 어떤 경우에 문제가 발생했을 때, 관리 계층의 에이전트가 하위 계층들을 재구성해서 구조에 영향을 최소화시키고, 하위 에이전트가 실행될 때, 에이전트에게 적당한 상위 에이전트를 연결시켜주는 역할을 하는 것을 말한다.

(2)계층에 맞는 에이전트 내부 모듈 설계

Fig. 8과 같이 관리 계층인 Supervisor agent에서는 UI의 기능을 제공하는 UI 모듈이 들어가게 된다. Supervisor agent는 사용자와의 인터페이스를 담당하고 구조를 재조정하는 가장 중요한 역할을 담당하게 됨으로써 외부의 공격을 당해 멈추게 되면, 구조에 문제가 생기게 된다. 이러한 문제점을 막기 위해, 기능 백업 영역인 Cooperator agent를 두었다. UI 모듈은 구조의 현재 구성상태를 보관하고, 보여주며, 침입 탐지 상태를 보여주고, 침입탐지가 사실인지를 확인해주는 기능을 제공한다.

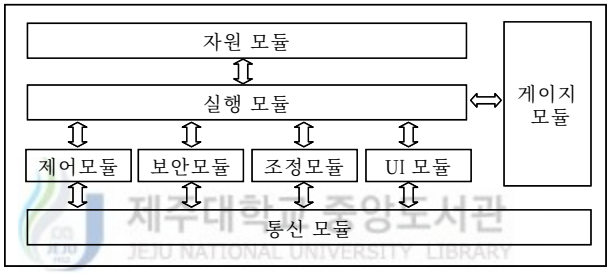

Fig. 8 Structure of Supervisor agent

Fig. 9와 같이 Cooperator agent에서는 Supervisor agent와 같은 기능을 하는 모듈들 외에도 Supervisor agent의 상태를 살펴서 문제가 발생하면, Cooperator agent를 실행시키는 역할을 하는 상태파악 모듈이 들어간다. 상태파악 모듈은 통신 모듈을 이용해 Supervisor agent의 존재 유무를 파악한다 Supervisor agent에서는 구조의 정보를 Cooperator agent에 보내준다. Cooperator agent에서는 이 정보를 가지고 있다가 Supervisor agent가 문제가 발생해서 기능을 발휘하지 못할 때, 최상위 Manager agent에게 자신의 위치를 알린다. 위치를 알린 후에는 Supervisor agent가 아닌 Cooperator agent에 보고를 한다. 상태파악 모듈에서는 Supervisor agent의 상태를 파악하는 모듈인데, Supervisor agent가 동작을 할 때는 구조의 구성상태를 보관하고 있다가, Supervisor agent가 문제가 생기면 UI 모듈로 구조의 구성상태를 보내주며, 제어모듈이 Cooperator agent의 위치를 하위 계층에 알린다.

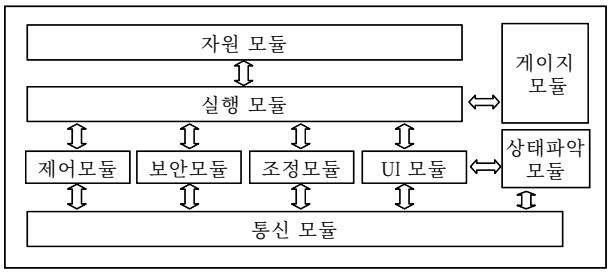

Fig. 9 Structure of Cooperator agent

Fig. 10과 같이 Manager agent에서는 Tool agent에서 올라오는 정보들을 분석하는 분석 모듈이 들어가는데, 이 분석 모듈은 이 구조가 적용되는 보안 관리 영역에 맞게 설계해야 하며, Tool agent에서 올라오는 공격 탐지의 신뢰정도가 어느 정도인지를 알 수 있는 탐지신뢰 매트리스를 분석 모듈에서 유지해야 한다. 이 매트리스는 관리자가 공격이 맞는지 아닌지를 확인해서 다시 알려주는 피드백 기능을 제공해야 한다. 피드백 기능을 제공함으로써 상황에 따라서 탐지신뢰 매트리스가 재구성 되도록 하여 Tool agent의 공격 탐지의 신뢰도를 항상 최신의 것으로 유지시킨다.

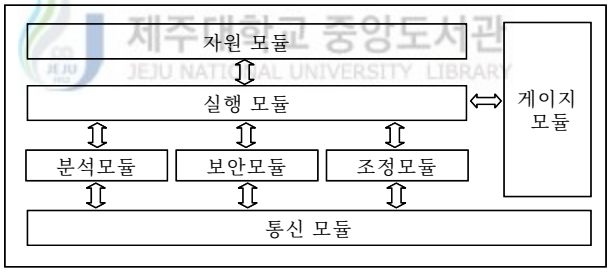

Fig. 10 Structure of Manager agent

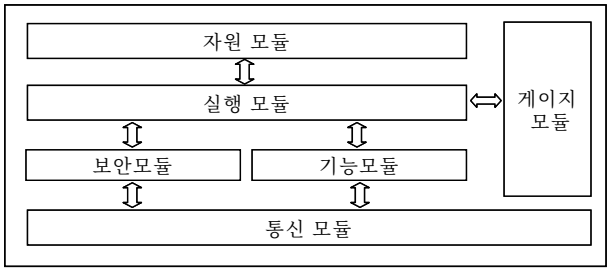

Fig. 11 Structure of Tool agent

Fig. 11과 같이 Tool agent에서는 침입탐지 시스템이나 방화벽의 차단기능을 수행하는 기능 모듈이 구현된다. Tool agent는 최하위 계층으로써 공격 탐지와 차단 등의 기

능을 구현할 수 있도록 기능 모듈이 설계되어야 한다.

(3)에이전트 특성 구현을 위한 에이전트 내부 모듈 설계

HAID에 고려된 자율성, 사교성, 이동성, 지능성을 구현하기 위해 모듈들을 설계하였다. 모듈은 자율성을 위한 자원 모듈, 실행 모듈, 사교성을 위한 통신 모듈, 조정 모듈, 지능성을 위한 제어 모듈과 보안 모듈, 게이지 모듈을 설계하였다.

①자원 모듈

시스템의 자원과 에이전트가 점유한 자원을 파악하는 모듈이다. Fig. 12와 같이 시스템 자원 파악을 요구하게 되면, 현재 남아 있는 시스템 자원의 양을 파악해서 보내주고, 에이전트 자원 소모량 파악을 요구하면, 에이전트가 사용하고 있는 자원의 양을 파악해서 보내준다. 이 모듈은 상황에 따라서 항상 가동을 하게 되거나, 필요할 때만 실행하게 된다.

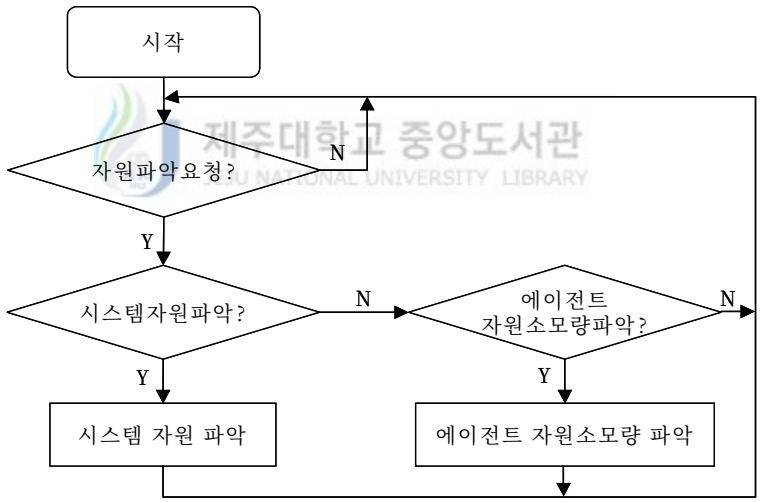

Fig. 12 Flow chart of resource module

②실행 모듈

여러 가지 모듈들을 요청에 따라서 실행/정지를 시키는 역할을 하는 모듈이다. 시스템의 자원의 양에 따라서 동적인 모듈 관리를 함으로써 자원을 최대한으로 이용하는 효과를 가져온다. Fig. 13과 같이 모듈 실행 요청이 있으면 일단 시스템의 자원의 양이 허용되는지를 자원 모듈을 이용해 파악한다. 시스템의 자원이 이용 가능하면 모듈을 실행하고, 시스템의 자원이 모듈을 실행시키기에 부족하면 모듈을 실행시키지 않는

다. 모듈 정지 요청이 오면 해당하는 모듈을 정지시킨다. 이렇게 모듈이 필요하지 않을 때 모듈을 정지시킴으로써 시스템의 자원을 필요로 하는 다른 모듈이나 프로그램이 자원을 이용할 수 있도록 하여 효율적인 모듈 관리를 한다.

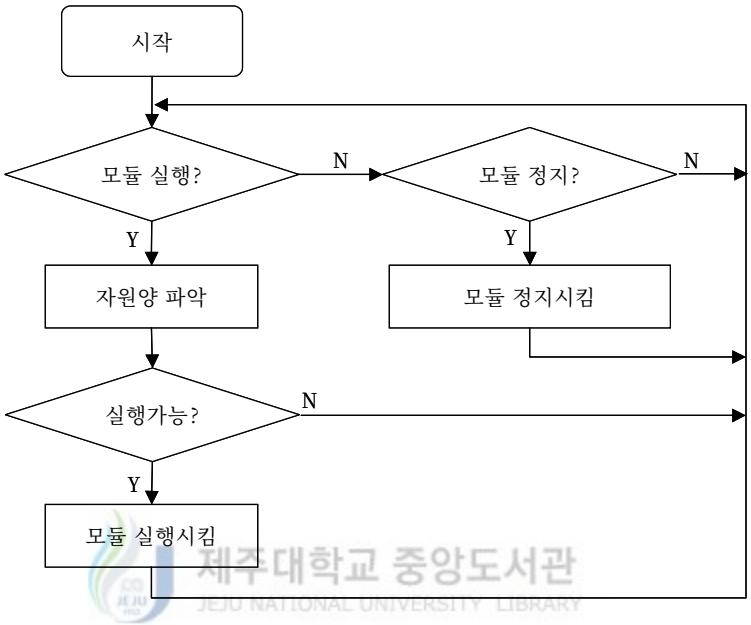

Fig. 13 Flow chart of execute module

③통신 모듈

에이전트끼리의 통신을 담당하는 모듈이다. Fig. 14와 같이 에이전트가 실행되면 관리 계층으로 접속해 자신의 상위 계층을 알게 된다. 자신의 상위 계층을 알게 되면, 상위 계층에 접속해 자신의 메타지식을 보낸다. 통신 모듈을 다른 모듈에서 이용하기 위해서 요청을 하면 다른 모듈에서 보내는 명령/데이터를 원하는 에이전트로 보낸다. 하위 계층의 에이전트가 불시에 멈추면 상위 계층으로 보고를 한다.

통신 모듈은 프로토콜을 이용해서 서로 통신을 주고받는다. 통신 모듈에서 사용하는 프로토콜은 KQML을 기반으로 하여 정의하였으며, KQML은 침입탐지 시스템에서는 적합하지 않으므로, 최소한의 정의만을 사용하고 필요한 부분은 추가하였다. KQML 수행문 중에서 꼭 필요한 ask, tell, insert, delete 수행문을 사용하고, report문과 command문을 추가하였다. report문과 command문은 우선 순위가 다른 수행문들보다 높은 수행문이다. 사용 형식은 아래와 같다.

· ask(sender : <word>, receiver : <word>, in-reply-to : <word>, reply-with : <word>, content : <word>)

ask문은 content에 해당하는 내용을 sender가 receiver에게 요구하는 것이다. ask문은 무조건 tell문으로만 응답해야 한다. reply-with는 보내는 메시지의 순서번호이다. 응답은 in-reply-to에 보내온 메시지의 순서번호를 사용한다. 메타지식을 요청할 때도 사용한다.

· tell(sender : <word>, receiver : <word>, in-reply-to : <word>, reply-with : <word>, content : <word>)

tell문은 ask문의 응답에 사용한다. content에는 응답이 들어간다.

· insert(sender : <word>, receiver : <word>, in-reply-to : <word>, reply-with : <word>, content : <word>)

본 논문에서 insert문과 delete문은 메타지식을 사용할 때만 사용하는 것으로 제한된다. insert문은 메타지식을 상위 에이전트에 보고하는데 사용한다. 상위 에이전트는 이 수행문을 받으면 tell문을 이용해 응답한다.

· delete(sender : <word>, receiver : <word>, in-reply-to : <word>, reply-with : <word>, content : <word>)

delete문은 상위 에이전트에 보고된 메타지식을 삭제하는데 사용한다. 상위 에이전트는 이 수행문을 받으면 tell문을 이용해 응답한다.

· report(sender : <word>, receiver : <word>, in-reply-to : <word>, reply-with : <word>, content : <word>)

report문은 하위 에이전트가 동작을 멈추어서 Supervisor agent로 보고를 할 때, 보안 모듈이 에이전트가 공격을 당한다고 Supervisor agent로 보고할 때, 공격탐지를 보고할 때 사용한다. content에는 보고할 내용이 들어간다.

· command(sender : <word>, receiver : <word>, in-reply-to : <word>, reply-with : <word>, content : <word>)

command문은 Supervisor agent에서 하위 에이전트로 명령을 내릴 때 사용한다. 명령은 content에 들어가며, 하위 에이전트에서는 content를 참고해 정보를 수정하거나 특정한 일을 수행한다.

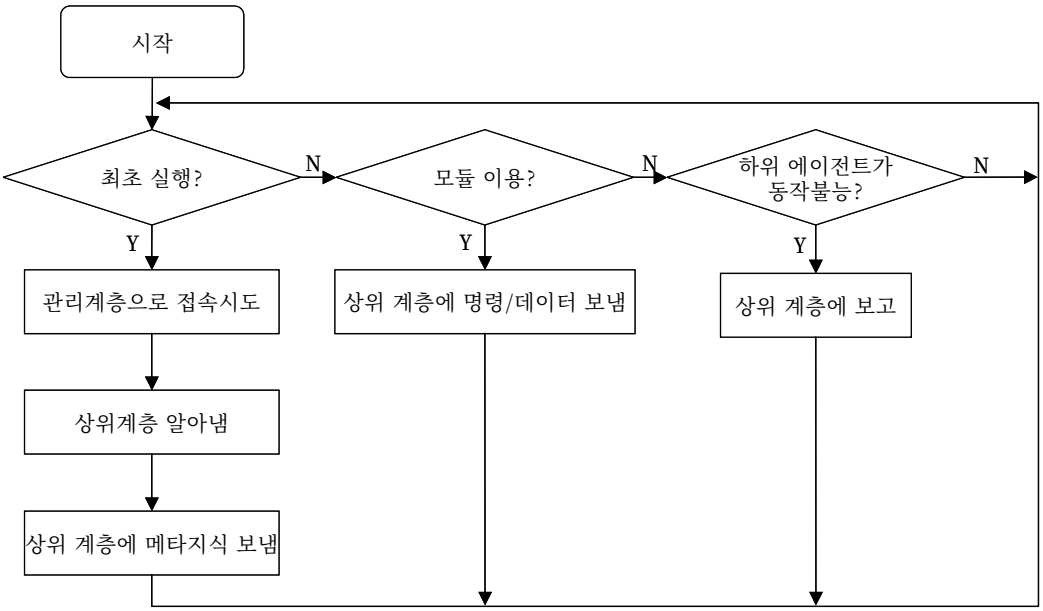

그림 14 Flow chart of communication module

④조정 모듈

블랙보드 기반의 모델을 이용한다(장과 최, 1999. 양과 최, 1999. 최등 1996). 조정자에 해당하는 블랙보드는 전문가에 해당하는 각 에이전트들에게 문제를 분산해서 나누어주며, 각 에이전트는 자신의 답을 블랙보드에 적어놓음으로써 다른 에이전트와 협력하게 된다. 이 모듈은 메타 지식을 보유해서 에이전트끼리 협력시켜주는 모듈이다. Fig. 15와 같이 메타지식을 저장하고 있다 메타지식을 검색하는 요청이 들어오면 메타지식을 검색한다. 자신이 메타지식을 가지고 있으면 해당 에이전트로 메타지식에 해당하는 기능을 요구해서 원하는 값을 요청한 에이전트로 돌려준다. 메타지식이 존재하지 않으면 상위 에이전트로 계속해서 요청한다. 구조의 최고 계층의 에이전트까지 메타지식을 요청하게 되는데, 존재하지 않으면 존재하지 않는다고 요청한 에이전트로 회답을 한다. 이러한 메타지식은 Manager agent와 Supervisor agent에서 가지고 있게 된다. 메타지식을 삭제하라는 요청이 들어오면 해당되는 메타지식을 삭제한다.. 메타지식의 삭제는 구조의 재조정 과정에서 발생한다.

메타지식은 다음과 같은 형식으로 본 논문에서 사용한다(최등 1996).

{}표시는 0 또는 한번 이상 여러 번 반복됨을 나타낸다. []은 나타나지 않거나 한번

나타날 수 있음을 의미한다.

<meta_knowledges> ::= [<m_knowledge> {, <m_knowledge> }]

<m_knowledge> ::= (<agent_name> , [<capability> {, <capability> }])

<agent_name> ::= <alphanumeric_str>

<capability> ::= <capability_name> [(<parameter> {, <parameter> })]

<capability_name> ::= <alphabet> { <alphanumeric> }

<parameter> ::= <alphanumeric_str> | <variable>

<alphanumeric_str> ::= <alphabet> { <alphanumeric> }

<alphanumeric> ::= <alphabet> | <digit>

<variable> ::= <capital> <alphanumeric>

<alphabet> ::= a | b | c | … | y | z

<digit> ::= 0 | 1 | 2 | … | 8 | 9 | '_' | '(' | ')' | ','

<capital> ::= A | B | C | … | Y | Z

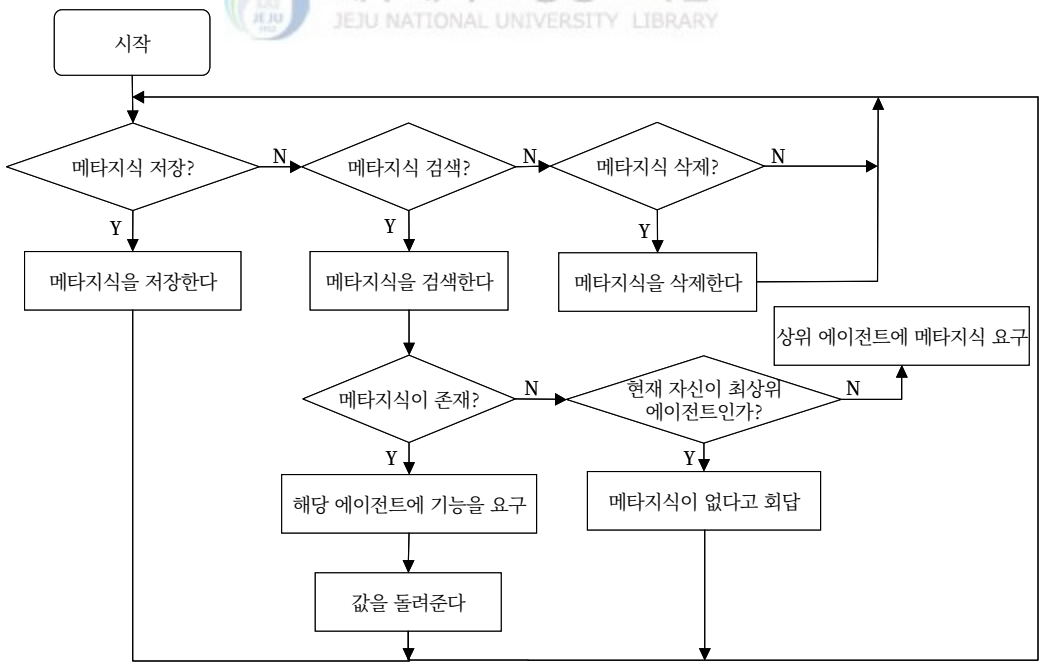

Fig. 15 Flow chart of collaboration module

⑤제어 모듈

Fig. 16과 같이 구조에 문제가 발생할 때, 구조를 재구성하고, 에이전트가 초기에 관리 계층으로 접속할 때, 에이전트에 적당한 상위 에이전트를 연결시켜주는 역할을 하는 모듈이다. 이 모듈은 관리 계층에 존재함으로써 하위 에이전트가 single point of failure 문제에 봉착하지 않도록 해주는 역할을 한다. 즉, 하나의 시스템이 공격을 받아서 그 시스템에서 실행되는 에이전트들이 동작을 멈추면, 관리 계층에 있는 제어 모듈에서 파악을 하여 그 에이전트의 하위 에이전트가 보고를 하는 경로를 바꿔준다. 따라서, 구조에 문제가 발생할 때 구조 전체의 기능이 마비되는 것을 막고, single point of failure 문제를 막기 위해 여러 가지 부가적으로 추가되는 시스템들을 줄일 수가 있게 된다. 이 모듈에서는 구조의 구성을 트리 구조를 이용해 저장하는데, 트리 구조의 root는 Supervisor agent가, 중간 단계는 Manager agent가, 마지막 노드들은 Tool agent가 위치한다.

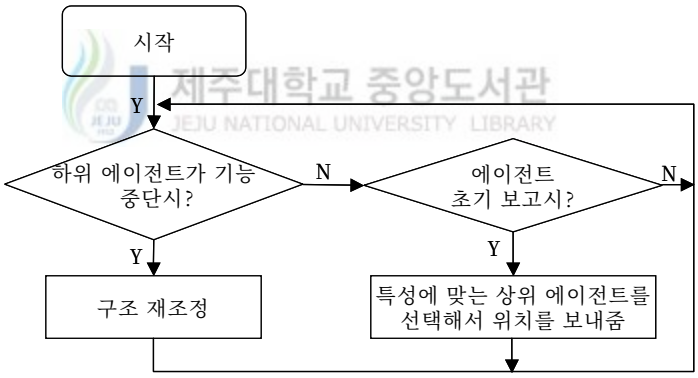

Fig. 16 Flow chart of control module

⑥보안 모듈

Fig. 17과 같이 에이전트 자신이 공격을 당하는지를 파악해내는 능력을 갖추고 에이전트가 공격을 당하고 있는지, 공격을 당하여 고의로 변형은 되지는 않는지를 파악하는 모듈이다. 이 모듈은 에이전트에 문제가 발생될 때, 상위 계층으로 보고를 해준다.

⑦게이지 모듈

Fig. 18과 같이 작업 양이 많아져서 내장된 모듈만으로 작업을 하기 힘들다고 판단이 될 때 구동되는 모듈로써 시스템의 자원이 허락하면 작업 양이 많은 기능 모듈을

스레드 기법을 이용해서 같은 기능을 수행하는 모듈을 여러 개 실행하여 작업 시간을 단축시키고, 손실률을 최소화시키며, 동시에 실행되는 기능 모듈들을 관리하는 역할을 한다. 작업 양이 줄어들면 동시에 실행되던 모듈들은 자원을 반납하고 사라진다.

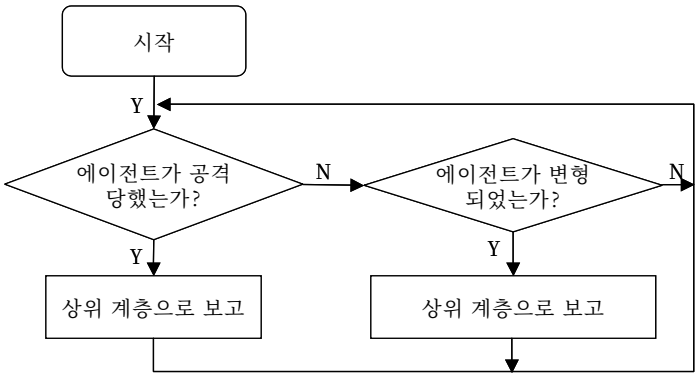

Fig. 17 Flow chart of security module

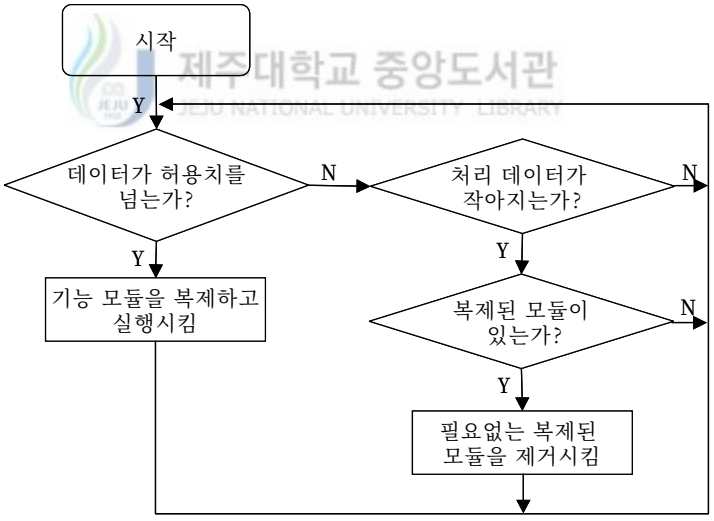

Fig. 18 Flow chart of gauge module

⑧UI 모듈

사용자와의 인터페이스를 담당하는 모듈이다. Fig. 19와 같이 구조의 현재 구성상태를 보관하고, 구성상태를 보여주며, 침입탐지 상태를 보여주고, 침입탐지가 사실인지 아닌지를 피드백 해주는 기능을 제공한다.

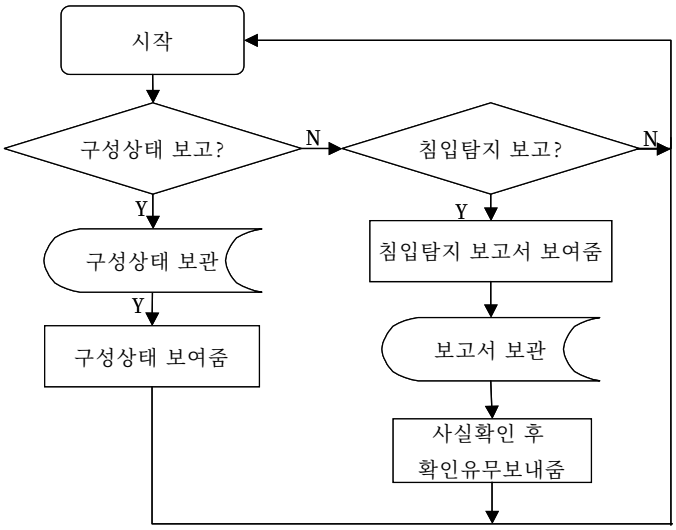

Fig. 19 Flow chart UI module

⑨분석 모듈

분석 모듈은 HAID가 적용되는 보안 관리 영역에 맞게 설계되어야 하며, Fig. 20과 같이 Tool agent에서 올라오는 탐지 보고서를 분석하여, 중복되는 부분은 합치고, 탐지에 대한 탐지신뢰 매트리스를 첨부하는 역할을 한다. Tool agent에서 올라오는 공격 탐지의 신뢰정도가 어느 정도인지 알 수 있는 탐지신뢰 매트리스를 유지해야 한다. 이 매트리스를 유지하려면, 관리자가 공격이 맞는지 아닌지를 확인해서 다시 알려주는 피드백 기능을 제공해야 한다. 피드백 기능을 제공함으로써 상황에 따라서 탐지신뢰 매트리스가 재구성 되도록 하여 Tool agent의 공격 탐지의 신뢰도를 항상 최신의 것으로 유지시킨다.

⑩상태 파악 모듈

Fig. 21과 같이 Cooperator agent에서 Supervisor agent의 상태를 조사하고, Supervisor agent가 동작할 때는 Supervisor agent에서 보내오는 구조의 구성상태를 보관하고 있다가 Supervisor agent가 동작을 멈추었을 때, 운영자에게 그 사실을 알리고, 하위 계층으로 자신의 위치를 알려서, Supervisor agent가 동작을 다시 시작할 때까지 그 기능을 대신한다.

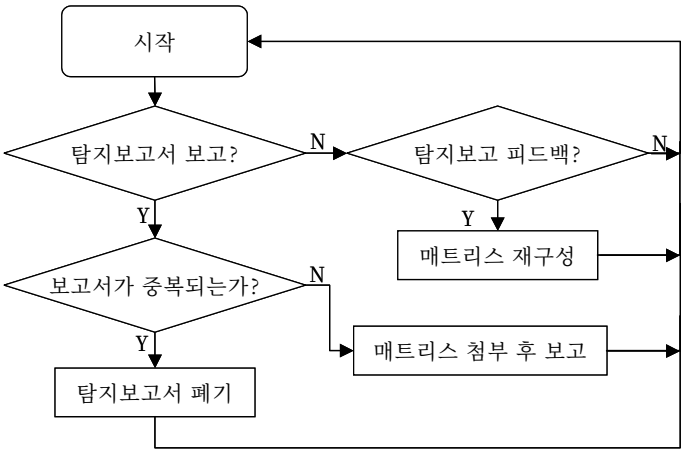

Fig. 20 Flow chart of analysis module

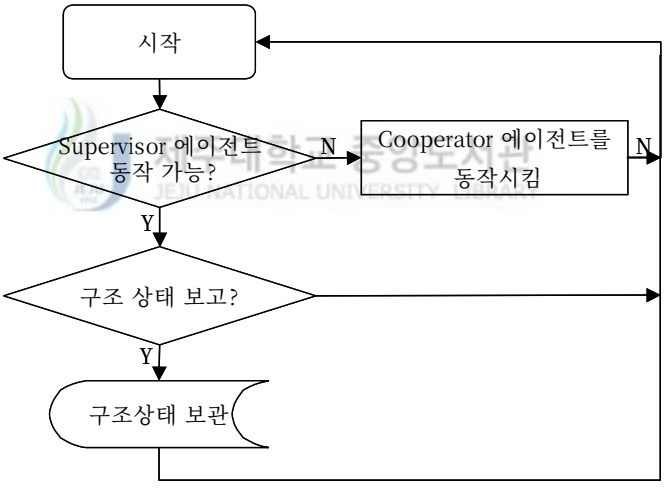

Fig. 21 Flow chart of state detection module

⑪기능 모듈

방화벽의 차단 기능, 네트워크 기반의 침입탐지 시스템, 호스트 기반의 침입탐지 시스템의 기능을 제공하는 모듈이다. 일반적으로 보안관리는 시스템 의존적인 정보와 네트워크 의존적인 정보를 어떻게 조화를 하는가 하는 것이 문제가 된다. 이러한 문제를 시스템 의존적인 정보를 가지고 있는 Tool agent의 기능 모듈과 네트워크 의존적인 정보를 가지고 있는 Tool agent의 기능 모듈이 Manager agent에 올려진 메타지식을 이용해서 서로 협동을 하여 문제를 해결할 수 있다.

2)동작 과정

(1)구조가 생성될 때

에이전트가 처음 실행될 때, Fig. 22와 같이 에이전트 내부의 통신 모듈은 Supervisor agent로 접속을 한다. Supervisor agent는 에이전트가 접속을 해 오면 적당한 상위 에이전트의 위치를 가르쳐 주어서 상위 에이전트를 연결시키게 된다. 상위 에이전트의 위치가 전송이 되면, 통신 모듈은 상위 에이전트로 자신의 메타지식을 보낸다.

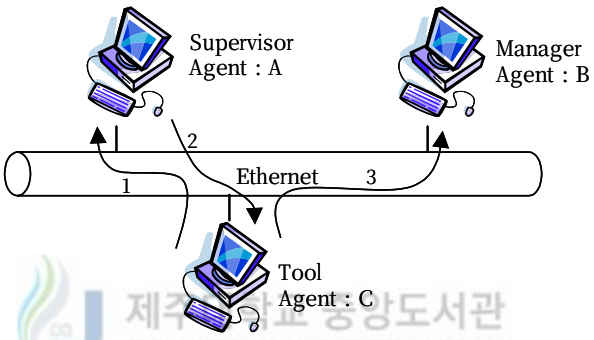

Fig. 22 A diagram that shows procedure for initial connection of Tool agent

Fig. 22에서 Tool agent가 실행될 때, Supervisor agent로 report(sender : C, receiver : A, reply-with : id31, content : "init-connect, name = c, type = tool, address = xxx.xxx.xxx.xxx, port = xxxx") 수행문으로 보고를 한다.

보고를 한 후, Tool agent는 Supervisor agent에게 ask(sender : C, receiver : A, reply-with : id32, content : "parent-agent, name, address, ip") 수행문으로 상위 에이전트의 위치를 요구한다.

Supervisor agent는 구조의 구성을 보아서 Tool agent에게 tell(sender : A, receiver : C, in-reply-to : id32, reply-with : id11, content : "name = B, address = xxx.xxx.xxx.xxx, port = xxxx") 수행문으로 상위 에이전트의 위치를 알려준다.

상위 에이전트의 위치를 알게 된 Tool agent는 자신의 메타지식을 insert(sender : C, receiver : B, reply-with : id33, content : "name = B, meta = xxx") 수행문으로 상위 에이전트에게 알려준다.

상위 에이전트는 tell(sender : B, receiver : C, in-reply-to : id33, reply-with : id21, content : "name = B, mata = xxx") 수행문으로 메타지식을 전송 받았다고 Tool agent에게 알려준다.

(2)공격이 탐지될 때

Fig. 23과 같이 공격자가 관리영역에 있는 시스템들을 공격을 하게 되고, 공격이 Tool agent의 기능 모듈에서 탐지가 되면 상위 계층으로 보고를 한다. 이때, 같은 공격이 계속해서 오게 되는데, 이것을 계속해서 상위 계층으로 보내주면 에이전트사이의 보고에 의한 네트워크 부하량이 많아져서 구조에 문제가 발생할 수 있다. 이러한 문제를 해결하기 위해 기능 모듈은 같은 공격이면 상위 계층으로 한번만 보고를 하도록 설계해야 한다. 상위 계층에서는 Supervisor agent까지 계속해서 보고를 해주고, Supervisor agent에서는 차단기능을 담당하는 Tool agent로 공격자의 트래픽을 차단하라는 명령을 내린다. Supervisor agent에서는 Tool agent로 공격을 차단했다고 통보한다. 이유는 차단이 제대로 되지 않았을 때는 Tool agent에서 다시 보고를 해야 하기 때문이다.

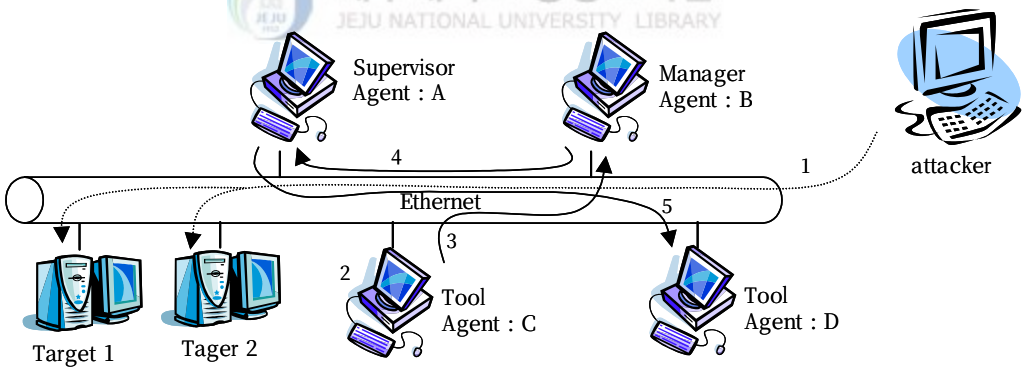

Fig. 23 A diagram that shows procedure for detection of attack

Fig. 23과 같이 Tool agent C가 공격을 탐지하면 report(sender : C, receiver : B, reply-with : id31, content : "attack_type = xxx, attack_IP = xxx.xxx.xxx.xxx, target_IP = xxx.xxx.xxx.xxx") 수행문으로 상위 에이전트인 B에게 공격유형, 공격 IP 주소, 대상 IP 주소를 보고한다.

B 에이전트에서는 report(sender : B, receiver : A, reply-with : id21, content : "attack_type = xxx, attack_IP = xxx.xxx.xxx.xxx, target_IP = xxx.xxx.xxx.xxx") 수

행문으로 상위 에이전트로 보고를 하는데, 이 때 하위 에이전트에서 올라온 내용을 그대로 보고한다.

Supervisor agent는 공격 탐지 보고를 받게 되면 command(sender : A, receiver : D, reply-with : id11, content : "block IP = xxx.xxx.xxx.xxx") 수행문으로 차단 기능이 있는 Tool agent D에게 해당 IP 주소를 차단하라는 명령을 내린다.

명령을 받은 D 에이전트는 차단을 하고 tell(sender : D, receiver : A, in-reply-to : id11, reply-with : id41, content : "block IP = xxx.xxx.xxx.xxx") 수행문으로 차단 여부를 보고한다.

보고를 받은 Supervisor agent A는 tell(sender : A, receiver : B, in-reply-to : id21, reply-with : id12, content : "block attack_type = xxx, attack_IP = xxx.xxx.xxx.xxx, target_IP = xxx.xxx.xxx.xxx") 수행문으로 하위 에이전트 B에게 차단 여부를 전해준다.

B 에이전트는 tell(sender : B, receiver : C, in-reply-to : id31, reply-with : id22, content : "block attack_type = xxx, attack_IP = xxx.xxx.xxx.xxx, target_IP = xxx.xxx.xxx.xxx") 수행문으로 C 에이전트에 전달한다.

(3)공격이 의심스러울 때

Fig. 24와 같이 일단 공격이 의심스러우면 Tool agent에서 Manager agent로 여러 가지 정보들을 요구하는데, Manager agent는 메타지식을 이용하여 그 정보들을 찾는다. 자신에게 해당하는 메타지식이 없을 경우 상위 Manager agent로 메타지식을 요구한다. 메타지식 검색 요청을 받은 Manager agent는 메타지식을 찾으면 해당 Tool agent에게 메타지식에 해당하는 정보를 요구한다. Tool agent는 요구에 맞는 정보를 요청한 Manager agent로 보내준다. Manager agent는 원래 요청한 Manager agent로 정보를 다시 보내주고 정보를 받은 Manager agent는 Tool agent로 보내준다. Tool agent에서는 받은 여러 가지 정보들을 파악함으로써 자신의 에이전트의 정보만으로 발견할 수 없는 공격들을 발견하게 된다. 탐지신뢰 매트리스를 Manager agent의 분석 모듈에서 관리를 함으로써 공격이 제대로 탐지되었는지를 파악할 수 있는 피드백이 되어서, Tool agent에서 오는 공격이 어느 정도의 탐지 신뢰도를 갖는지 파악할 수 있다.

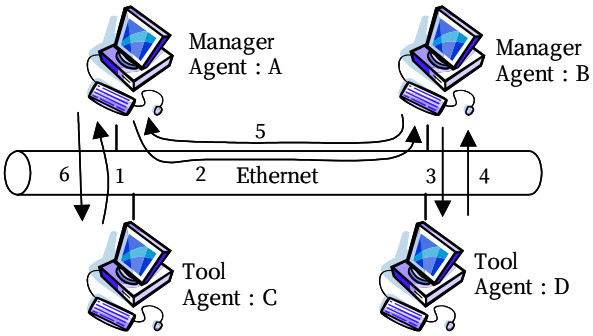

Fig. 24 A diagram that shows procedure for detection of suspicious attack

Fig. 24와 같이 Tool agent C에서 ask(sender : C, receiver : A, reply-with : id31, content : "meta = xxx") 수행문으로 Manager agent A로 메타지식을 요구한다.

A 에이전트에서는 현재 자신에게 메타지식이 없으므로 ask(sender : A, receiver : B, reply-with : id11, content : "meta = xxx")수행문으로 상위 에이전트인 B 에이전트로 메타지식을 요청한다.

B 에이전트에서는 ask(sender : B, receiver : D, reply-with : id21, content : "meta = xxx") 수행문으로 메타지식을 소유하고 있는 Tool agent D에 메타지식을 요구한다.

D 에이전트는 tell(sender : D, receiver : B, in-reply-with : id21, reply-with : id41, content : "xxxxx") 수행문으로 B 에이전트에 요구한 메타지식에 해당되는 정보를 보내준다.

B 에이전트는 tell(sender : B, receiver : A, in-reply-with : id11, reply-with : id22, content : "xxxxx") 수행문으로 A 에이전트에 D 에이전트에서 보내온 정보를 보내준다.

D 에이전트는 tell(sender : A, receiver : C, in-reply-with : id31, reply-with : id12, content : "xxxxx") 수행문으로 메타지식을 요구했던 C 에이전트에 정보를 보내준다.

(4) 에이전트가 공격을 받을 때

Fig. 25와 같이 에이전트가 공격을 받게 되면 에이전트의 내부에 있는 보안 모듈이 상위 계층으로 보고를 한다. 상위 계층에서는 Supervisor 계층까지 계속해서 보고를 한다. Supervisor agent에서는 하위 계층의 Manager agent가 공격을 받고 있다는 보

고를 받게 되면 차단기능을 가진 Tool agent에 해당 에이전트를 공격하는 IP 주소를 차단하라는 명령을 내린다.

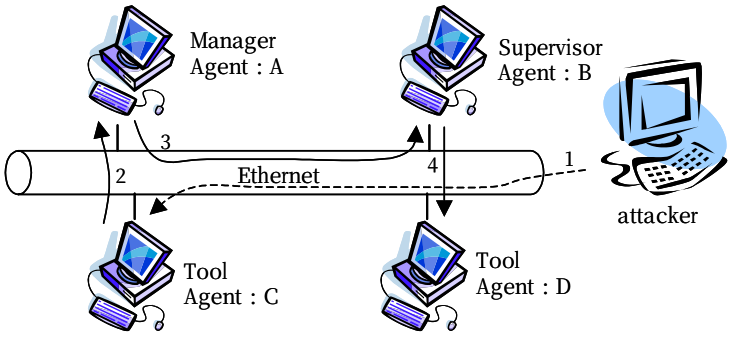

Fig. 25 A diagram that shows procedure when agent is attacked

Fig. 25와 같이 Tool agent C가 공격을 받으면, C 에이전트 내부의 보안모듈에서 report(sender : C, receiver : A, reply-with : id31, content : "agent-attacked name = C, attack_type = xxx, attack_IP = xxx.xxx.xxx.xxx or attack_reason = xxx") 수행문으로 상위 에이전트인 Manager agent A에 보고를 해준다. 이 때, 네트워크 공격이면 attack_IP에 공격 IP 주소가 들어가며, 호스트 내부의 문제일 경우에는 attack_reason에 이유가 들어간다.

A 에이전트에서는 report(sender : A, receiver : B, reply-with : id11, content : "agent-attacked name = C, attack_type = xxx, attack_IP = xxx.xxx.xxx.xxx or attack_reason = xxx") 수행문으로 하위 에이전트 C에서 올라온 정보를 그대로 상위 에이전트 B에 보고한다.

Supervisor agent인 B 에이전트에서는 command(sender : B, receiver : D, reply-with : id21, content : "block IP = xxx.xxx.xxx.xxx") 수행문으로 차단 기능을 수행하는 Tool agent D에 명령을 내리는데, attack_reason으로 올라온 정보는 차단을 수행할 수 없고, 단지 관리자에게 보고만 한다.

명령을 받은 D 에이전트에서는 차단을 수행하고 tell(sender : D, receiver : B, in-reply-to : id21, reply-with : id41, content : "block IP = xxx.xxx.xxx.xxx") 수행문으로 B 에이전트에게 차단 여부를 보고한다.

(5)에이전트가 어떤 이유로 멈추었을 때

Fig. 26과 같이 보안 모듈이 파악을 하지 못하는 공격에 의해서나 에이전트가 실행되는 시스템이 정전 등의 사태에 의해서 동작을 멈추게 될 때, 상위 계층의 에이전트의 통신 모듈이 Supervisor agent로 하위 계층의 에이전트가 불시에 동작을 멈추었다고 보고한다. Supervisor agent의 제어 모듈에서는 이 보고를 받으면, 적당한 에이전트에게 그 기능을 대신하라는 명령을 내린다. 동작을 멈춘 에이전트의 메타지식을 상위 에이전트에서는 삭제한다.

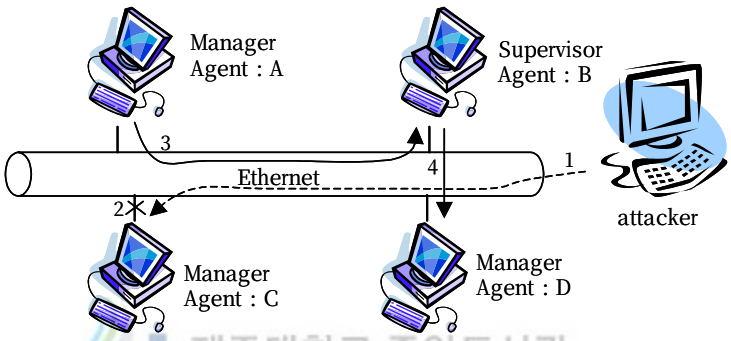

Fig. 26 A diagram that shows procedure when agent is stopped suddenly

Fig. 26과 같이 하위 에이전트 C가 갑자기 동작을 멈추게 되면, 상위 에이전트인 A가 report(sender : A, receiver : B, reply-with : id11, content : "agent-stop name = C") 수행문으로 상위 에이전트 B에 C 에이전트가 멈추었다고 보고를 한다.

Supervisor agent인 B는 command(sender : B, receiver : D, reply-with : id21, content : "parent-agent, name = A, address = xxx.xxx.xxx.xxx, port = xxxx") 수행문으로 C 에이전트의 하위 에이전트인 D에게 상위 에이전트를 조정해준다.

상위 에이전트를 전달받은 D 에이전트는 tell(sender : D, receiver : B, in-reply-to : id21, reply-with : id31, content : "parent-agent, name = A, address = xxx.xxx.xxx.xxx, port = xxxx") 수행문으로 상위 에이전트를 전달받았다고 B 에이전트에 보고한다.

이렇게 유동적으로 에이전트의 구조가 재편성될 수 있음으로 해서 single point of failure 문제를 해결하기 위해서 문제를 발생하는 곳에 또 다른 시스템을 갖다 놓는 것

을 방지할 수 있다. 하드웨어를 더 설치해야 되는 문제를 소프트웨어적으로 해결함으로써 최소한의 시스템을 가지고 문제를 해결할 수 있게 하였다. 최하위 계층에 해당하는 Tool agent가 공격을 당할 때에는 관리자에게 통보를 하고, 방화벽에 공격을 차단하라는 명령을 내린다. Tool agent는 Supervisor agent에서 다른 부분에서 기능을 대신하라고 명령을 내릴 수가 없게 된다. 그 이유는 Tool agent는 저마다 다른 기능을 수행하고 영역을 담당하고 있기 때문에 단지 관리자에게 통보를 하고 공격을 막게 된다. 반면에 Manager agent는 비슷한 기능들을 가지고 있기 때문에 다른 부분에서 그 기능을 대신할 수 있다.

(6) Supervisor agent가 공격을 받아서 기능을 수행 못할 때

Fig. 27과 같이 Supervisor agent는 평상시에 Cooperator agent와 통신을 유지한다. 구조의 구성도를 Cooperator agent에 전송을 한다. 공격을 당해서 Supervisor agent가 작동이 불가능할 때, Cooperator agent에서는 상태 파악 모듈을 통해서 작동 불능을 확인하고 위치를 확보하고 있는 Manager agent로 자신의 위치를 알리고 자신에게 보고를 올리라는 명령을 내린다. 자신의 위치를 알리면 Cooperator agent는 Supervisor agent가 다시 동작을 할 때까지 그 역할을 대신하게 된다.

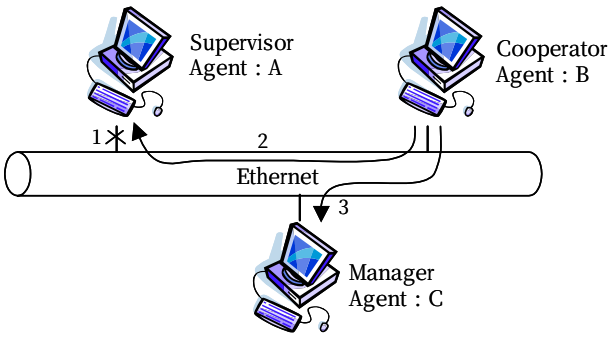

Fig. 27 A diagram that shows procedure when Supervisor agent is stopped

Fig. 27과 같이 Supervisor agent가 갑자기 동작을 멈추게 되면, Cooperator agent는 탐지를 하고 실행을 하게 된다. 동작이 수행되면, command(sender : B, receiver : C, reply-with : id21, content : "parent-agent, name = B, address = xxx.xxx.xxx.xxx, port = xxxx") 수행문으로 하위 에이전트인 C에게 자신의 위치를 알린다.

상위 에이전트 B의 위치를 전달받은 C 에이전트는 tell(sender C, receiver : B, in-reply-to : id21, reply-with : id31, content : "parent-agent, name = B, address = xxx.xxx.xxx.xxx, port = xxxx") 수행문으로 위치를 전달받았다고 상위 에이전트 B에게 보고를 한다. 이렇게 함으로써 Supervisor agent가 동작을 중지하더라도 구조는 Cooperator agent에 의해서 조정이 되게 된다.

Ⅳ. 제안된 시스템의 프로토타입 구현 및 고찰

프로토타입 시스템은 Supervisor agent, Cooperator agent, Manager agent인 MAN1, MAN2, MAN3과 Tool agent인 SCAN, SYN을 구현해서 윈도우, 리눅스 플랫폼에서 동작되었다. 프로토타입 시스템은 구현된 기능모듈이 탐지하는 공격유형에 대하여 탐지를 제대로 수행하였으며, 구조에 대한 실험을 해본 결과 구조에 이상이 생길 경우, 구조를 정상적으로 재조정하였다.

1. 프로토타입 구현환경

HAID의 프로토타입 시스템 구현을 위한 환경은 다음과 같다.

- Supervisor agent : Windows XP on IBM PC
- Cooperator agent : Wwindows Me on IBM PC
- Manager agent : Windows 98 on IBM PC
- Tool agent : Linux Kernel 2.4.2-2 on IBM PC
- Language : SUN's Java 2 SDK Standard Edition 1.3.1(JDK)
  GNU's gcc-2.96.81
- Packet capture library : PCAP library 0.62(PCAP)

Supervisor agent, Cooperator agent, Manager agent는 자바언어를 이용하여 구현을 하였으며, Tool agent는 침입탐지 시스템의 기능을 구현하기 위해 tcpdump와 같은 프로토콜 분석기에 많이 쓰이는 PCAP을 이용하여 gcc와 자바언어로 구현을 하였다. 에이전트들이 자바 언어를 이용함으로써 플랫폼 독립적인 특성을 가지고 있으나, Tool agent는 gcc를 이용해서 구현함으로써 리눅스기반에서만 동작을 한다.

2. HAID의 구성요소 구현

HAID의 동작 및 가능성을 확인하기 위해서 프로토타입 시스템의 기본 구조 및 몇 가지의 탐지 규칙에 대해 구현을 하였다. 공격 탐지를 위해서 사용되는 탐지 규칙은 빈도수가 높은 공격에 대해 구현하였다.

1)Supervisor agent의 구현

Supervisor agent는 사용자 인터페이스와 HAID를 관리하고 공격에 대한 방어를 할 수 있는 역할을 제공한다. 구현상에는 하위 계층에서 전송된 탐지정보를 감시할 수 있는 인터페이스와 하위 계층에 문제가 발생할 때, HAID를 재조정할 수 있게 하였다. Fig. 28은 Supervisor agent의 UI 모듈 부분을 보인 것이다. 구조의 생성, 변경 같은 구조에 대한 메시지가 전송되면, architecture and information window에 정보를 보여주고, 구조에 대한 공격이나 하위 에이전트에서 공격을 탐지하게 되면, 그 메시지는 detection attack window를 통하여 정보가 보여지게 된다.

아래는 구조의 구성도를 관리하는 클래스이다.

```
class arch_store{
  int agent_type;  // 에이전트 타입
  String agent_name;  // 에이전트 이름
  String agent_IP_address;  // 에이전트가 위치한 컴퓨터의 IP 주소
  int agent_port;  // 에이전트가 위치한 컴퓨터의 port 주소
  static int agent_all_ID;  // 에이전트에 배당되는 ID의 전체 수
  static int max_agent_level;  // Manager agent의 계층 수
  int agent_ID;  // 현재 에이전트의 ID
  int parent_agent_ID;  // 부모 에이전트의 ID
  int agent_level;
  int child_counter; // 자식 에이전트들의 수
        // 에이전트가 보고되면 자동으로 할당되는 생성자
  public arch_store(int type, String name, String IP, int port, int level ){
```

```
    agent_type = type;
    agent_name = name;
    agent_IP_address = IP;
    agent_port = port;
    agent_level = level;
    child_counter=0;
    agent_ID = agent_all_ID++;
    if ( level > max_agent_level && type == 1)
      max_agent_level = level;
      }
     // 버퍼용으로 쓰기 위한 생성자
    public arch_store(){}
}
```

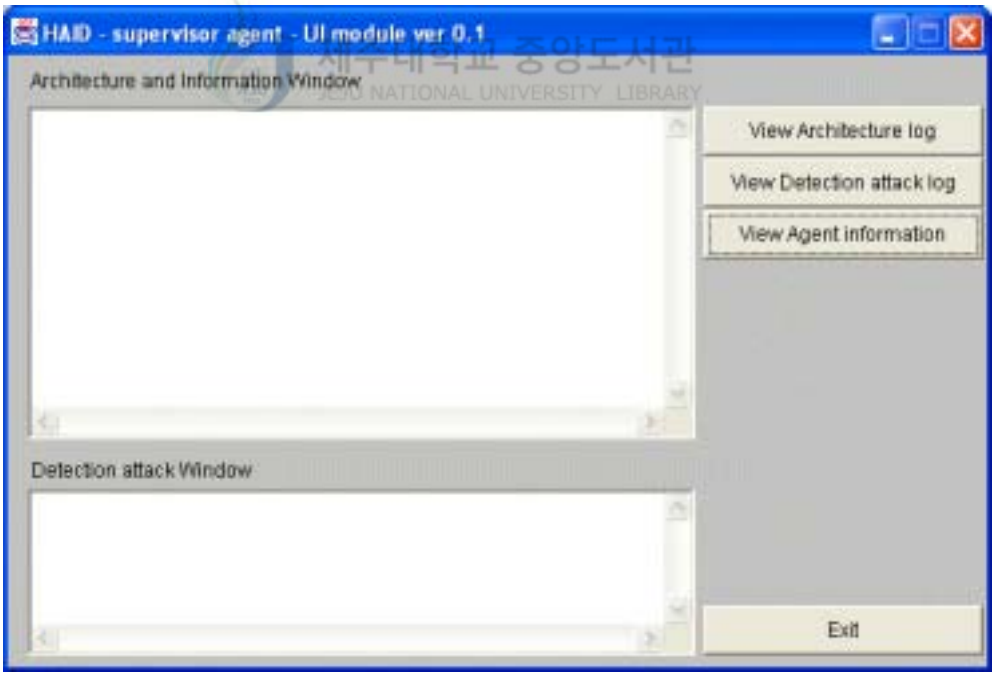

Fig. 28 Snapshot of UI module of Supervisor agent

Fig. 29는 구조에 대한 변경사항에 대한 로그를 보여주는 곳으로써 관련된 정보를 감시할 수 있고, 보고된 정보를 architecture log에 기록함으로써 구조가 생성된 다음

부터 구조에 어떠한 변경사항이 발생했는지 참고할 수 있다.

Fig. 29 Snapshot of architecture log window

Fig. 30은 하위 에이전트가 공격을 탐지하여 보고한 메시지를 보여주는 곳으로써 언제 공격을 탐지했는지 detection attack log에 기록함으로써 이후의 관리작업에 참고할 수 있다.

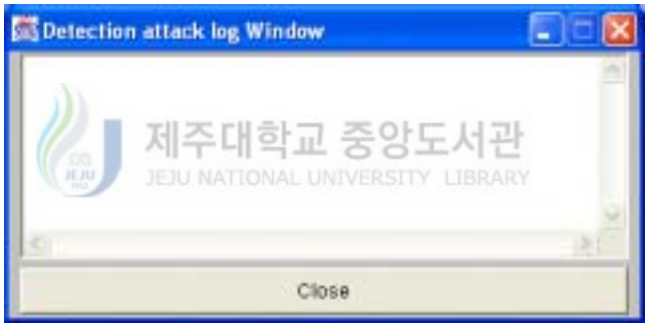

Fig. 30 Snapshot of detection attack log window

Fig. 31은 현재 구조에 구성된 에이전트들의 정보를 보여주는 창으로써, 이 창을 통하여 에이전트들의 정보를 상세히 알 수 있다.

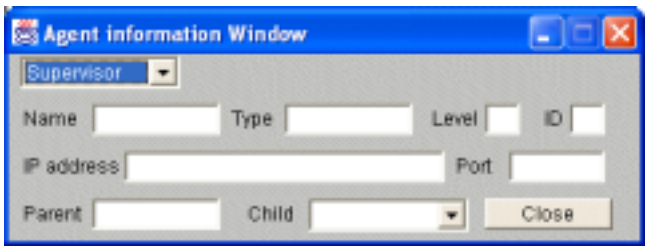

Fig. 31 Snapshot of agent information window

Supervisor agent는 구조의 관리를 책임지는 핵심적인 부분이므로, 플랫폼 독립적인 특성을 가져야 하는 경우가 많다. 이러한 특성을 만족시키기 위해 Supervisor agent는 자바로 구현되어서, 자바가상머신(JVM, java virtual machine)이 동작하는 시스템에서

는 플랫폼 독립적인 특성을 가진다.

2)Cooperator agent의 구현

Cooperator agent는 Supervisor agent의 기능백업 영역으로, Supervisor agent와 같은 기능을 가지며, Supervisor agent의 상태를 파악할 수 있게 하였다. 이 Cooperator agent는 Fig. 32와 같이 Supervisor agent와 같은 UI 모듈을 사용하기 때문에 외형은 같지만, Supervisor agent가 동작을 수행하는 동안에는 UI 모듈은 동작하지 않는다. Supervisor agent가 동작을 멈추게 되면, 그 순간에 Cooperator agent의 UI 모듈은 동작을 수행한다.

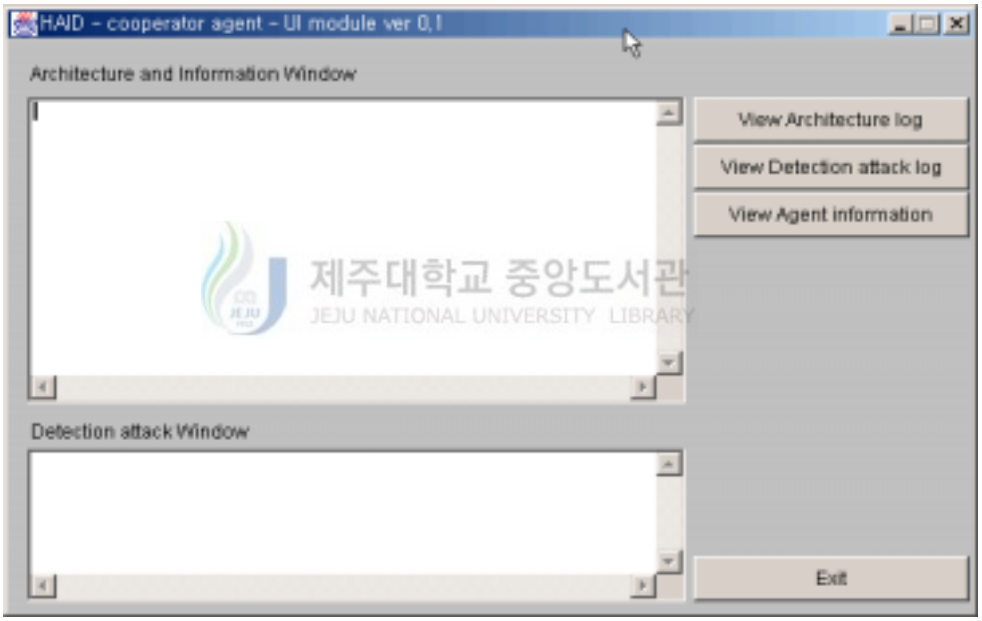

Fig. 32 Snapshot of ui module of Cooperator agent

3)Manager agent의 구현

Manager agent는 Tool agent에서 올라오는 정보를 분석하고 메타지식을 검색할 수 있도록 구현되었다. 3개의 Manager agent가 구성된다. 다음은 메타지식을 처리하기 위한 클래스이다.

```
class blackbord_module{
  public void insert_meta(String meta){}  // 메타지식을 저장하는 메소드
```

```
  public String search_meta(String meta){}  // 메타지식을 찾는 메소드
  public void delete_meta(String meta){}  // 메타지식을 삭제하는 메소드
}
```

다음은 하위 에이전트와 상위 에이전트 사이의 통신을 유지하기 위한 기본 정보의 저장을 위하여 정의한 클래스이다.

```
class common{
  static client_main client_parent;  // 상위 에이전트 연결 함수 포인터
  static String parent_IP;  // 상위 에이전트 IP 주소
  static int parent_port;  // 상위 에이전트 port 번호
  static String agent_name;  // 자신의 에이전트 이름
  static String super_IP;  // Supervisor agent의 IP 주소
  static int super_port;  // Supervisor agent의 port 주소
}
```


4)Tool agent의 구현

Tool agent는 침입탐지 시스템의 기능이 들어가는 부분으로, 2개의 Tool agent가 구현되었다. 2개의 Tool agent는 각각 syn 공격을 탐지하는 기능모듈과 scan 공격을 탐지하는 기능모듈을 포함하도록 구현되었다. 다음은 통신모듈 내에서 메타지식을 상위 계층에 전송하고, 공격을 탐지하는 메소드이다.

```
class server_service extends Thread{            // 통신 모듈
  public void init_meta_report(String data){}  // 메타지식을 상위 계층에 보고하는 메소드
  public void detect_attack(){}  // 공격을 탐지하는 메소드
}
```

3. 구현결과 및 고찰

구현된 프로토타입 시스템은 두 개의 이더넷 세그먼트와 다수의 관리대상 시스템을 포함한 보안관리 영역에서 공격도구를 활용한 공격을 수행하고, 전달된 결과를 바탕으로 구조의 동작을 확인하고, 구조의 특성을 파악하기 위해 여러 가지 테스트를 수행하는 형태로 수행되었다. Supervisor agent, Cooperator agent와, Manager agent인 MAN1, MAN2, MAN3 에이전트와 Tool agent인 SYN, SCAN 에이전트를 이용해서 구조를 생성하고, syn공격과 scan공격을 공개된 공격도구를 이용해 수행하였고, Manager agent인 MAN2와 Supervisor agent의 동작을 멈추게 하였다.

Fig. 33은 Supervisor agent의 UI 모듈 부분으로, Cooperator agent, Manager agent인 MAN1, MAN2, MAN3, Tool agent인 SCAN, SYN이 구조에 등록되었으며, MAN2가 동작을 멈추게 되는 것을 보여주고, syn 공격과 scan 공격이 탐지되었음을 보여준다.

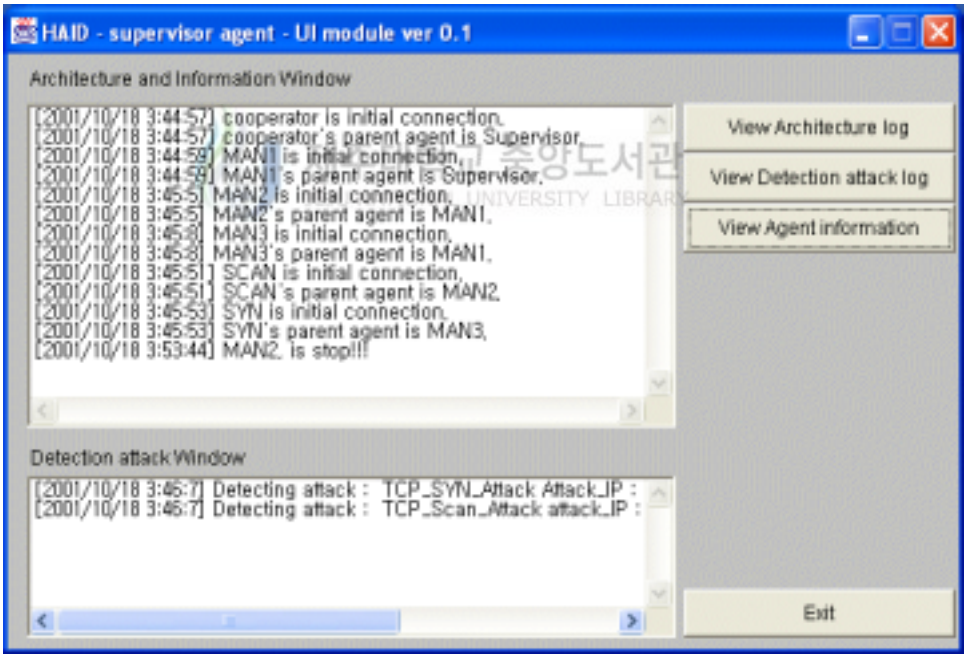

Fig. 33 Snapshot of Supervisor agent of prototype system testing

Fig. 33의 architecture and information window를 보면, ‘cooperator is initial connection.’은 Cooperator agent가 실행이 되어 Supervisor agent로 보고를 한 것이다. ‘cooperator's parent agent is Supervisor.’는 Supervisor agent에서 Cooperator agent의 상위 에이전트를 Supervisor agent로 지정을 한 것이다.

'MAN1 is initial connection.'은 MAN1 에이전트가 실행이 된 것이고, 'MAN1's parent agent is Supervisor.'는 MAN1 에이전트의 상위 에이전트가 Supervisor 에이전트이다.

'MAN2 is initial connection.'은 MAN2 에이전트가 실행이 된 것이고, 'MAN2's parent agent is MAN1.'은 MAN2 에이전트의 상위 에이전트가 MAN1이다.

'MAN3 is initial connection.'은 MAN3 에이전트가 실행이 된 것이고, 'MAN3's parent agent is MAN1.'은 MAN3 에이전트의 상위 에이전트가 MAN1이다.

'SCAN is initial connection.'은 SCAN 에이전트가 실행이 된 것이고, 'SCAN's parent agent is MAN2.'는 SCAN 에이전트의 상위 에이전트가 MAN2이다.

'SYN is initial connection.'은 SYN 에이전트가 실행이 된 것이고, 'SYN's parent agent is MAN3.'는 SYN 에이전트의 상위 에이전트가 MAN3이다.

'MAN2, is stop!!!'은 MAN2 에이전트가 갑자기 동작을 멈추었다는 것을 보여준다.

Fig. 33의 detection attack window를 보면, 'Detecting attack : TCP_SYN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103'은 syn 공격을 탐지하였으며, 공격 IP 주소는 203.200.200.201이고 대상 IP 주소는 203.253.213.103이라는 것을 알려준다.

'Detecting attack : TCP_Scan_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103'은 scan 공격을 탐지하였으며, 공격 IP 주소는 203.200.200.201이고 대상 IP 주소는 203.253.213.103이라는 것을 알려준다. 이렇게 공격 탐지 보고가 올라오면, 차단 기능을 가진 Tool agent로 해당 IP 주소를 차단하라고 Supervisor agent에서 명령을 내린다.

Fig. 34 Snapshot of detection log window of prototype system testing

Fig. 34는 Fig. 33에 보이는 detection attack window에 보이는 내용을 로그에 저장

한 내용을 보여주고 있는데, Fig. 33에 보이는 view detection attack log 버튼을 누르면 보여지는 창이다.

Fig. 35 Snapshot of agent information window of prototype system testing

Fig. 35는 Fig. 33에 보이는 view agent information 버튼을 누르면 나오는 창으로 구조를 구성하고 있는 에이전트에 대해 보여주는 창이다. Supervisor와 MAN1, MAN2, MAN3, SCAN, SYN 에이전트가 등록되어 있다는 것을 보여주고 있으며, 에이전트 이름, 형태, 계층의 레벨, 고유 식별 번호, 에이전트가 위치한 컴퓨터의 IP 주소와 포트번호, 에이전트의 상위 에이전트와 하위 에이전트들을 확인할 수 있다.

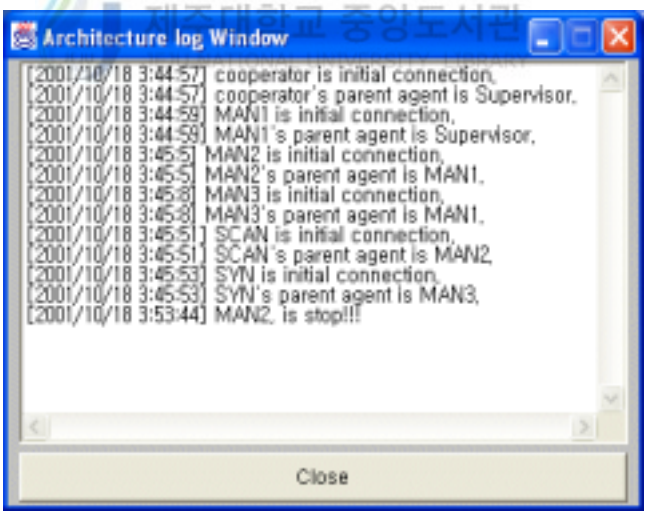

Fig. 36 Snapshot of architecture log window of prototype system testing

Fig. 36은 Fig. 33에 보이는 architecture and information window에 보이는 내용을 로그에 저장한 내용을 보여주고 있는데, Fig. 33에 보이는 view architecture log 버튼을 누르면 보여지는 창이다.

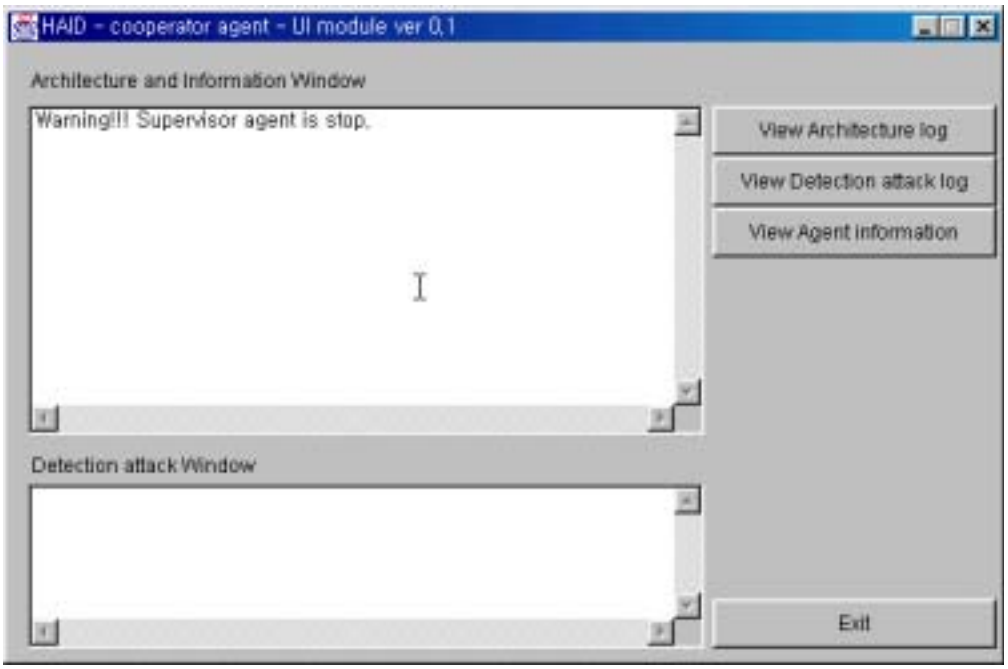

Fig. 37 Snapshot of Cooperator agent of prototype system testing

Fig. 37은 Supervisor agent가 동작을 멈추었을 때, Cooperator agent가 실행이 되는 것을 보여주고 있다.

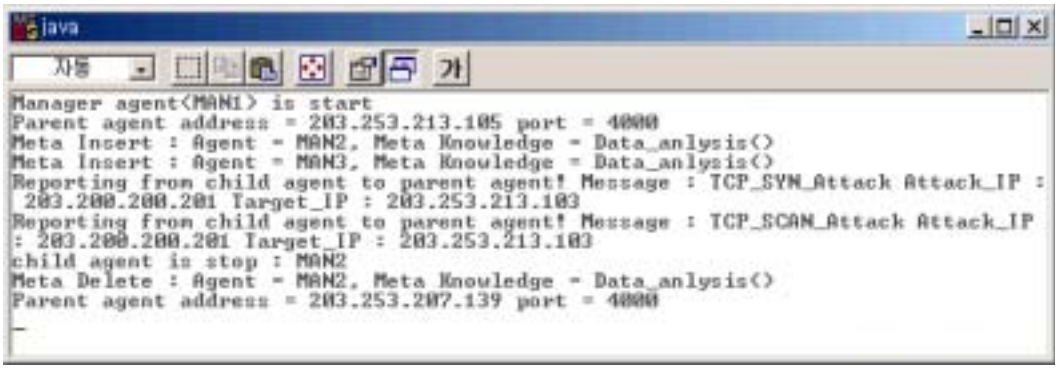

Fig. 38 Snapshot of Manager agent(MAN1) of prototype system testing

Fig. 38은 Manager agent인 MAN1 에이전트를 보여주고 있는데, 'Manager agent<MAN1> is start'는 MAN1이 실행되었음을 보여주며 MAN1이 실행되면서 Supervisor agent로 자신의 IP 주소와 포트 번호를 전송하고, 상위 에이전트의 정보를 얻게 된다. 'Parent agent address = 203.253.213.105 port = 4000'은 상위 에이전트가 위치한 컴퓨터의 IP 주소와 포트 번호를 Supervisor agent로부터 전송 받은 것이다.

‘Meta Insert : Agent = MAN2, Meta Knowledge = Data_anlysis()’은 하위 에이전트 MAN2가 메타지식을 전송해 온 것이다.

‘Meta Insert : Agent = MAN3, Meta Knowledge = Data_anlysis()’은 하위 에이전트 MAN3이 메타지식을 전송해 온 것이다.

‘Reporting from child agent to parent agent! Message : TCP_SYN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103’은 하위 에이전트에서 상위 에이전트로 ‘TCP_SYN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103’ 메시지를 보내는 것을 보여주고 있다.

‘Reporting from child agent to parent agent! Message : TCP_SCAN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103’은 하위 에이전트에서 상위 에이전트로 ‘TCP_SCAN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103’란 메시지를 보내는 것을 보여주고 있다.

‘child agent is stop : MAN2’는 하위 에이전트인 MAN2가 동작을 멈추었다는 것을 보여주며, 이렇게 하위 에이전트가 동작을 멈추게 되면 MAN1에서 Supervisor agent로 하위 에이전트인 MAN2가 동작을 멈추었다고 보고를 하게 된다.

‘Meta Delete : Agent = MAN2, Meta Knowledge = Data_analysis()’는 하위 에이전트인 MAN2가 동작을 멈추게 되면, 보관하고 있던 MAN2의 메타지식을 삭제하게 된다.

‘Parent agent address = 203.253.207.139 port = 4000’은 상위 에이전트인 Supervisor agent가 동작을 멈추어서 Cooperator agent가 동작을 하면서 MAN1에 위치를 전송한 것이다. 이후부터는 MAN1은 전송 받은 위치인 Cooperator agent로 보고를 한다.

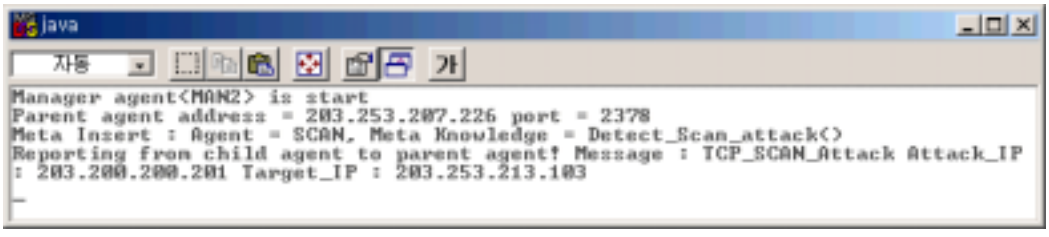

Fig. 39 Snapshot of Manager agent(MAN2) of prototype system testing

Fig. 39는 Manager agent인 MAN2 에이전트를 보여주고 있는데, ‘Manager agent<MAN2> is start’는 MAN2가 실행되었음을 보여주며 MAN2가 실행되면서

Supervisor agent로 자신의 IP 주소와 포트 번호를 전송하고, 상위 에이전트의 정보를 얻게 된다.

'Parent agent address = 203.253.207.226 port = 2378'은 상위 에이전트가 위치한 컴퓨터의 IP 주소와 포트 번호를 Supervisor agent로부터 전송 받은 것이다.

'Meta Insert : Agent = SCAN, Meta Knowledge = Detect_Scan_attack()'은 하위 에이전트인 SCAN이 메타지식을 전송해 온 것이다.

'Reporting from child agent to parent agent! Message : TCP_SCAN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103'은 하위 에이전트에서 상위 에이전트로 'TCP_SCAN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103'이란 메시지를 보내는 것을 보여주고 있다.

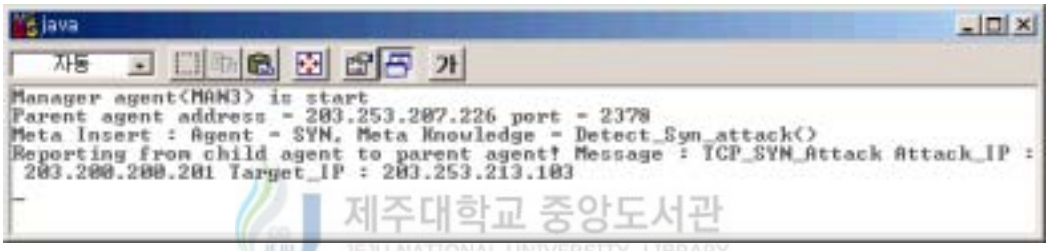

Fig. 40 Snapshot of Manager agent(MAN3) of prototype system testing

Fig. 40은 Manager agent인 MAN3 에이전트를 보여주고 있는데, 'Manager agent<MAN3> is start'는 MAN3이 실행되었음을 보여주며 MAN3이 실행되면서 Supervisor agent로 자신의 IP 주소와 포트 번호를 전송하고, 상위 에이전트의 정보를 얻게 된다.

'Parent agent address = 203.253.207.226 port = 2378'은 상위 에이전트가 위치한 컴퓨터의 IP 주소와 포트 번호를 Supervisor agent로부터 전송 받은 것이다.

'Meta Insert : Agent = SYN, Meta Knowledge = Detect_Syn_attack()'은 하위 에이전트인 SYN이 메타지식을 전송해 온 것이다.

'Reporting from child agent to parent agent! Message : TCP_SYN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103'은 하위 에이전트에서 상위 에이전트로 'TCP_SYN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103'이란 메시지를 보내는 것을 보여주고 있다.

Fig. 41은 Tool agent인 SYN 에이전트를 보여주고 있는데, 'Tool agent(Syn) is start'

는 SYN이 실행되었음을 보여주며, 'Parent agent address = 203.253.207.221 port = 6456'은 상위 에이전트가 위치한 컴퓨터의 IP 주소와 포트 번호를 Supervisor agent로부터 전송 받은 것이다. 이렇게 상위 에이전트의 위치를 전송 받으면 상위 에이전트로 메타지식을 전송하게 된다.

Fig. 41 Snapshot of Tool agent(SYN) of prototype system testing

'Detect attack : TCP_SYN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103'은 SYN 에이전트에서 IP 주소 203.200.200.201에서 IP 주소 203.253.213.103으로 syn 공격을 하는 것을 탐지한 것이다. 이렇게 탐지를 하면, 상위 에이전트로 보고를 해 주게 된다.

'Parent agent address = 203.253.207.226 port = 2378'은 상위 에이전트인 MAN2가 동작을 멈추어서 Supervisor agent에서 SYN 에이전트의 상위 에이전트를 다시 전송해 준 것이다. 이렇게 전송을 해줌으로써 구조의 에이전트가 멈추게 되더라도 구조를 재조정해서 에이전트의 동작불능으로 인한 구조의 오동작을 막을 수 있다.

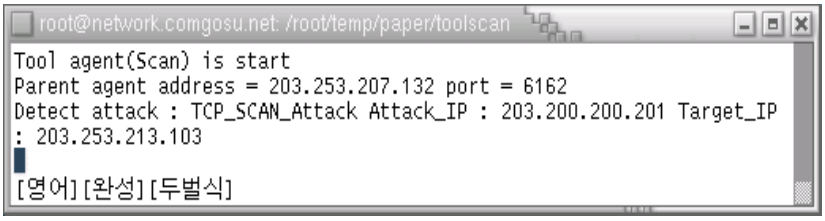

Fig. 42 Snapshot of Tool agent(SCAN) of prototype system testing

Fig. 42는 Tool agent인 SCAN 에이전트를 보여주고 있는데, 'Tool agent(Scan) is start'는 SCAN이 실행되었음을 보여주며, 'Parent agent address = 203.253.207.132 port = 6162'는 상위 에이전트가 위치한 컴퓨터의 IP 주소와 포트 번호를 Supervisor agent로부터 전송 받은 것이다.

‘Detect attack : TCP_SCAN_Attack Attack_IP : 203.200.200.201 Target_IP : 203.253.213.103’은 SCAN 에이전트에서 IP 주소 203.200.200.201에서 IP 주소 203.253.213.103으로 scan 공격을 하는 것을 탐지한 것이다. 이렇게 탐지를 하면, 상위 에이전트로 보고를 해 주게 된다.

지금까지 프로토타입 시스템을 통하여 동작과정을 확인하였다. 프로토타입 시스템은 정해진 순서대로 행해진 테스트에 대해 정상적으로 동작을 하였으며, 하나의 에이전트가 중단되었을 경우, Supervisor agent에서 구조를 재 설정함으로써 구조가 정상적으로 동작을 하게 됨을 확인하였다. 기존의 침입탐지 시스템들은 보통 중앙집중식의 시스템으로 구현되고 있지만, 중앙의 시스템이 멈추게 되었을 경우, 침입탐지 시스템이 동작을 멈추게 된다. 이러한 특성을 이용하여 침입탐지 시스템을 겨냥한 허위 공격들이 늘고 있는 상태에서 제안된 시스템은 계층적인 에이전트와 구조를 재조정하는 과정을 통해 이러한 단점을 개선했다고 볼 수 있고, 일반적으로 네트워크 의존적인 정보와 시스템 의존적인 정보가 모두 필요한 경우가 많이 발생하는데 이러한 경우에도 네트워크 의존적인 정보를 수집하는 Tool agent의 기능모듈과 시스템 의존적인 정보를 수집하는 Tool agent의 기능모듈을 구현함으로써 침입탐지 시스템 우회 공격들을 발견할 수 있게 된다.

Ⅴ. 결론

본 논문에서는 계층적인 에이전트를 이용한 새로운 침입탐지 구조를 제안하였다. 구조를 계층적인 에이전트를 이용하여 설계하기 위해서, 각 계층의 에이전트 내부의 모듈들을 계층에 맞게 배치하였으며, 각 에이전트의 특성을 구현하기 위해 에이전트 내부 모듈을 설계하였으며, KQML기반의 간단한 프로토콜을 제시하였고, 상황에 따른 구조의 동작과정을 설명하였다.

제안 시스템은 네트워크 세그먼트에 설치되어진 Tool agent에서 공격을 탐지하며, Tool agent에서 필요한 정보는 메타지식을 보유하고 있는 상위 계층인 Manager agent에 요청을 하여 침입탐지 시스템을 우회하는 공격에 대해 탐지를 할 수 있게 하였다. 새로운 공격기법이 등장할 경우에는 해당 공격기법에 대한 탐지기능을 수행하는 Tool agent의 기능모듈을 추가해서 새로운 공격에 대하여 탐지를 수행할 수 있다.

Tool agent의 기능 모듈을 제외하고는 모두 Java 언어를 이용해 구현하였으므로, 이질적인 네트워크 환경에서 효율적으로 운영이 가능하다. 기능 모듈을 시스템 의존적인 c언어(gcc)를 이용해 구현했으므로, 시스템 의존적인 정보를 Java 언어에서 접근할 때 발생할 수 있는 문제를 해결하였다.

제안된 시스템을 두 개의 네트워크 세그먼트와 다수의 관리대상 시스템으로 구성된 보안관리 영역에서 실제의 공격 도구를 이용하여 실험한 결과, 정상적으로 공격을 탐지함을 보았으며, 구조에 대한 여러 가지 시험과정을 통해 구조가 정상적으로 동작함을 확인하였다.

기존의 AHA! IDS는 대규모 네트워크에 적당하나, 하드웨어 비용이 많이 드는 단점과 중간 계층의 에이전트들이 두개 이상이 멈추게 되었을 경우, AHA! IDS가 동작을 제대로 수행하지 못하는 단점이 존재하였다. 본 논문에서는 이러한 단점을 최상위 계층인 Supervisor agent에서 에이전트가 동작을 멈추었을 때, 구조를 재조정하여 AHA! IDS에서의 단점인 하드웨어 비용 문제와 동작 수행 문제를 개선하였다. 그러나, 계층적인 에이전트를 이용하였기 때문에 분산 시스템에서 나타나는 지연시간 문제가 발생

할 수 있다.

제안 시스템의 에이전트들은 상호인증이 되었다는 가정 하에서 구현되었다. 좀 더 완전한 시스템을 위하여 에이전트간의 인증에 대한 연구와 다양한 오류제어를 수행할 수 있는 프로토콜 구현에 대한 연구가 필요하겠다.

참고문헌

Anonymous, 2000, 리눅스 보안의 모든 것, 인포북 SAMS

CERT-kr, http://www.certcc.or.kr/

COAST, http://cerias.purdue.edu/coast/

CSRC, http://csrc.nist.gov/focus_areas.html

daemon9 AKA route, 1996, Project Hades: TCP weaknesses, Phrack Magazine Volume 7, Issue 49, article 06 of 16

D. Anderson, T. Frivold and A. Valdes, 1995, Next-generation intrusion detection expert system(NIDES), Technical Report SRI-CLS-95-07

Daniel J. Ragsdale, 2000, Adaptation Techniques for Intrusion Detection and Intrusion Response Systems, IEEE

Fyodor, 1997, The Art of Port Scanning, Phrack Magazine Volume 7, Issue 51 September 01, 1997, article 11 of 17

H, Javitz and A. Valdes, 1991, The SRI IDES statistical anomaly detector, In Proceeding of the IEEE Symposium on Research in Security and Privacy, pp316-326

horizon, 1998, Defeating Sniffers and Intrusion Detection Systems, Phrack

Magazine Volume 8, Issue 54 Dec 25th, 1998, article 10 of 12

hybrid, 1999, Distributed Information Gathering, Phrack Magazine Voume 9, Issue 55, article 09 of 19

Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, Devid Isacoff, Eugene Spafford, Diego Zamboni, 1998, An Architecture for Intrusion Detection using autonomous Agents, Tech Report 98-05, COAST Laboratory, Department of Computer Scientce, Purdue University, West Lafayette, IN

JDK, http://java.sun.com

Karanjit Siyank, Chris Hare, 1999, 인터넷 방화벽과 네트워크 보안, 이한출판사

Karanjit S. Siyan, 1998, TCP/IP 완전정복, 성안당

Lars Klander, 1998, HACKER PROOF, 정보문화사

lifesux, 1996, Port Scanning without the SYN flag / Uriel Maimon, Phrack Magazine Volume 7, Issue 49, article15 of 16

Mark Crosbie and Gene Spafford, 1995a, Defending a computer system using autonomous agents, In Proceedings of the 18th National Information Systems Security Conference

Mark Crosbie and Gene Spafford, 1995b, Active defense of a computer system using autonomous agents, Technical Report 95-008, COAST Group, Department of Computer Sciences, Pursue University, West Lafayette, IN 47907-1398

Mark Slagell, 2001, The Design and Implementation of MAIDS, Computer Science Department, Iowa State Univ.

M. Asaka, S. Okazawa, A. Taguchi, and S. Goto, 1999, A Method of Tracing Intruders by Use of Mobile Agents, Proceedings of 9th Annual Internetwoking Conference(INET'99), San Jose, California

Ofir Arkin & Fyodor Yarochkin, 2001, ICMP based remote OS TCP/IP stack fingerprinting techniques, Phrack Magazine Volume 0x0b, Issue 0x39, Phile #0x07 of 0x12

orabidoo, 1997, File Descriptor Hijacking, Phrack Magazine Volume 7, Issue 51 September 01, 1997, article 05 of 17

P. A. Porras and P. G. Neumann, 1997, EMERALD: Event monitoring enabling responses to anomalous live disturbances, In National Information Systems Security Conference, pp353-365, Baltimore, MD

PCAP, http://www.tcpdump.org

PLUS, 2000, Security PLUS for UNIX, 영진.com

Rebecca Gurley Bace, 2000, Inrusion Detection, MACMILLAN TECHNICAL PUBLISHING

solar designer, 1998, Port Scan Detection Tools, Phrack Magazine Volume 8, Issue 53, article 13 of 15

SRI/SDL, www.sdl.sri.com

S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip and D. Zerkle, 1996, GrIDS-A Graph based intrusion detection system for large networks, In Proceedings of the 19th National Information Systems Security Conference

T. G. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. G. Neumann, H. S. Javitz, A. Valdes, and T. D. Garvey, 1992, A Real-Time Intrusion Detection Expert System(IDES)-Final Technical Report, CSI, SRI International, Menlo Park, California

Tim Finin, Rich Fritzson , Don McKay and Robin McEntire, 1994, KQML as an Agent communication Language, CIKM '94

William Stallings, 1997, 통신망 정보보호, 도서출판 그린

W. Lee, S. J. Stolfo, and K. Mok, 1999, A Data Mining Framework for Building Intrusion Detection Models, Proceedings of the IEEE Symposium on Security and Privacy 1999. http://www.cs.columbia.edu/~sal/JAM/PROJECT

W. Richard Stevens, 1999, TCP/IP 네트워크, 도서출판 진영사

Yannis Labrou, Tim Finin, 1997, A proposal for a new KQML Specification, TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250.

김거수, 2001a, 철벽보안을 위한 역공격 해킹, 베스트북

김병구, 정태명, 2000, 침입탐지 기술의 현황과 전망, 정보과학회지 18권 1호, pp29-39

김상철, 2001b, Abnormal IP Packets, CERT-kr

김심기외 A.H.C팀, 2000, Haking@Linux, 마이트Press

김영균, 김성윤, 장경훈, 김경식, 2001a, 자율에이전트를 이용한 능동적인 침입탐지 시스템 구조, 한국통신학회 하계종합학술발표회 논문집, pp.1921-1924

김영균, 김성윤, 장경훈, 김경식, 2001b, 자율에이전트를 이용한 침입탐지 시스템, Journal of Research Institute of Industrial Technology Cheju National University, Vol. 12 No.1 pp.106-111

노용환, 정해윤, 최길준, 2000, 해킹과 보안, 영진.com

박현미, 신웅경, 이현우, 2000, 네트워크 스니핑 기술 및 방지대책, CERT-kr

방성민, 2000, 이동에이전트 기술을 이용한 네트워크 패킷기반 침입탐지 시스템

양재영, 최중민, 1999, 협동 에이전트 시스템, 전자공학회지 26권 1호, pp. 25-33

이경하, 은유진, 임채호, 정태명, 1998, 네트워크 패킷 정보를 기반으로 한 보안 관리, 정보과학회논문지(A) 제 25권 12호, pp1405-1412

이재호, 2000a, 에이전트 시스템의 연구 및 개발 동향, 정보과학회지 18권 5호, PP4-9

이현우, 2000b, 네트워크 공격기법의 패러다임 변화와 대응방안 -Part Ⅰ: 네트워

크 공격기법의 패러다임 변화 v1.0-, CERT-kr

이현우, 2001, 네트워크 공격기법의 패러다임 변화와 대응방안 -Part II : 대응방안 v0.1-, http://www.securitymap.net/

이현우, 박현미, 2000, 네트워크 스캔공격 탐지 통계 분석, CERT-kr

이현우, 정현철, 1999, 분산 환경에서의 서비스거부 공격 분석보고서, CERT-kr

장지훈, 최중민, 1999, 이질적 에이전트의 동적 관리를 위한 자바 기반의 에이전트 관리 시스템, 정보과학회 논문지(A), 26권 7호, pp. 778-787

정보보호진흥원, www.kisa.or.kr/technology/sub3/IDS.htm

정현철, 1996, TCP Connection Hijacking 공격 및 대책, CERT-kr

정현철, 2001, IP Fragmentation을 이용한 공격기술들, CERT-kr

최종호, 조성배, 2001, 침입탐지 시스템을 위한 은닉 마르코프 모델의 적용, 정보과학회논문지 : 소프트웨어 및 응용 28권 6호, pp429-438

최중민, 1997, 에이전트 개요와 연구방향, 1997년, 정보과학회지 15권 3호, pp7-16

최중민, 백순철, 장명욱, 박상규, 임영환, 1996, 이형 분산 환경에서 에이전트들간의 이질성과 분산성을 극복하기 위한 멀티에이전트 기반구조, 정보과학회논문지(C) 2권 1호, pp1-19

홍승필, 고제욱, 1998, 정보보안 기술과 구현, 파워북

한국정보보호센터, 2000, 정보보호 기술표준연구 최종 연구개발 결과보고서, 정보통신부