안전한 이동 에이전트 게이트웨이의 설계 및 구현
(Design & Implementation of Secure Mobile Agent Gateway)

박 재 경 † 원 유 헌 ††
(Jae-Kyoung Park) (Yoo-Hun Won)

요 약 최근 인터넷 환경 속에서 이동 에이전트와 관련된 기술들이 지속적인 관심과 연구의 대상이 되고 있다. 이동 에이전트는 사용자가 정한 여정 리스트에 따라 자율적으로 이동을 하며 사용자의 요구사항을 대신하는 대리인의 역할을 한다. 이러한 이동 에이전트를 통해 분산 환경 속에서 여러 가지 이득을 얻을 수 있게 되었다. 하지만, 이동 에이전트는 보안상 많은 취약점을 가지고 있다. 특히 이동 에이전트에 대한 불법적인 호스트의 공격은 이동 에이전트가 상업적으로 널리 이용될 수 있는데 있어서 가장 큰 걸림돌이 되고 있다. 이에 따른 이동 에이전트 자체에 대한 보안으로 많은 연구가 진행되고 있다. 이 논문에서는 이동 에이전트가 가지고 있는 이동성에 따른 위험성을 여정 리스트의 분석을 통해 제거하고 코드의 중요도에 따라 코드를 분할 및 재생성하여 이동 에이전트 자체의 보안을 유지하고자 한다. 또한 이러한 이동 에이전트의 분할 및 재생성을 수행하는 역할을 이동 에이전트 게이트웨이(MAG-Mobile Agent Gateway)로 설계 및 구현하였다. MAG을 통해 이동 에이전트는 보다 안전하게 사용자의 역할을 대신함으로 인해 이동 에이전트가 사용자의 역할을 안전하게 수행하는데 기여하고자 한다.

키워드 : 보안, 이동 에이전트, 가상 사설망

Abstract In the course of Internet proliferation, many network-related technologies are examined for possible growth and evolution. The use of Internet-based technologies in private networks has further fuelled the demand for network-based applications. The most promising among the new paradigms is use of mobile agents. The mobile agent is capable of migrating autonomously from node to node in the network, to perform some computation on behalf of the user. The mobile agent paradigm is attractive alternative to traditional client-server programming for a significant class of network-centric applications. It also however, suffers from a major drawback, namely, the potential for malicious attacks, abuse of resources pilfering of information, and other security issues. These issues are significantly hampering the acceptance of the mobile-agent paradigm. This paper describe the design & implementation of secure mobile agent gateway that split and merge the agent code with security policy database. This mechanism will promote the security in mobile agent systems and mobile agent itself.

Key words : Security, Mobile Agent, MAG, VPN

1. 서 론

1.1 이동 에이전트 시스템

인터넷 환경은 사용자로 하여금 많은 정보를 온라인으로 접하여 각종 서비스에 유용하게 이용되고 있다. 최근 인터넷 환경 속에서 네트워크와 관련된 많은 기술들이 연구되어 왔다[1][2][3]. 이 중 클라이언트가 적용할 주문형태의 기능을 제공하는 방법으로 이동 에이전트에 대한 연구가 폭넓은 관심을 가지게 되었다. 이동 에이전트는 임의의 호스트로부터 다른 호스트를 이동하며 사용자의 역할을 대신 수행하는 프로그램이다. 이동 에이전트는 코드, 상태 변수 그리고 다음 이동지의 리스트를 나타내는 여정리스트로 구성되어져 있다. 코드는 에이전트가 수행되는 동안 변경되지 않는 반면에 상태변수는 이동 에이전트가 수행되는 동안 변경된다. 이동 에이전트는 클라이언트의 요구를 수용할 수 있는 기능과 프로그램 내부에 소프트웨어 모듈들을 집합화하여 이동할

† 비 회 원 : (주)퓨처시스템 정보통신연구소 연구원
jkpark@future.co.kr
†† 종신회원 : 홍익대학교 전산학과 교수
won@cs.hongik.ac.kr
논문접수 : 2001년 11월 5일
심사완료 : 2002년 1월 4일

수 있다는 장점을 통해 부각되기 시작했다. 이는 서로 다른 분산환경에서 사용자의 역할을 대신하여 특정한 작업을 수행할 수 있다는 것이다. 이동 에이전트는 우선 사용자의 기계에 위치한 후 수행을 위해 원격지로 보내지게 된다. 호의적인 호스트들은 이동 에이전트가 수행하기 위한 적절한 환경을 제공한다. 이를 이동 에이전트 플랫폼이라 한다. 이동 에이전트는 이동 후 호스트 내에서 수행되어 정보를 수집하고 다음 여행지로 이동하기 위하여 자체적으로 상태와 변수들을 실시간적으로 수정한다. 이러한 여행을 마지막 호스트까지 수행한 후 자신의 홈으로 귀환한다. [그림 1]은 이러한 이동에이전트의 동작 및 이동 에이전트 시스템에 대한 구성도를 나타내고 있다[4][5][6]. 또한 [그림 2]는 이동 에이전트의 수행 모델을 도식화하고 있다.

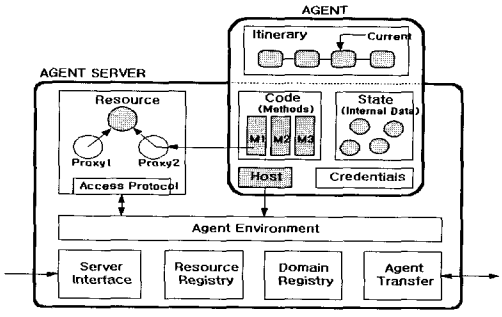

그림 1 이동 에이전트 시스템 구성

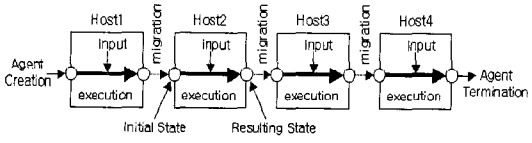

그림 2 에이전트 수행 모델

이동 에이전트는 자율성, 적응성, 이동성과 같은 특성으로 인해 획기적인 패러다임으로 여겨졌으나 보안상 많은 취약점을 가지고 있다. 즉, 이동 에이전트를 통한 호스트에 대한 보안과 호스트에 의한 이동 에이전트에 대한 보안, 또한 이동 에이전트간의 정보 교환 시 발생할 수 있는 이동 에이전트간에 대한 보안 등이다. 예를 들어 네트워크를 이동하는 이동 에이전트에 대한 신뢰성과 보안성에 대한 관심이 일어났다. 즉, 이동 에이전트가 어떠한 유해한 방해나 공격없이 수행할 수 있을 것인가 하는 의문이 제기되었다.

본 논문에서는 이동 에이전트의 취약점 가운데 이동 에이전트가 불법적인 호스트를 통해 공격을 당할 수 있는 위험성을 분석하고 이를 해결하기 위한 방안에 대해 해결책을 제시하고 이를 구현한 이동 에이전트 게이트웨이에 대해서 살펴보기로 한다.

1.2 유해한 호스트의 문제점

이동 에이전트를 수행하는 이동 에이전트 시스템은 이동 에이전트를 자신의 호스트 내부에서 수행하므로 바이러스나 트로이 목마와 같은 공격에 노출될 수 있다. 이러한 이동 에이전트 시스템뿐만 아니라 이동 에이전트 자체도 위험에 노출되어 있다[7][8][9]. 호스트에 의해 제공되는 수행환경을 통해 호스트는 이동 에이전트를 완벽하게 제어할 수 있다. 즉, 코드를 분석하거나 또는 상태 등의 동적 코드나 정적코드를 모두 변경할 수도 있다.

- 일반적인 이동 에이전트 프로그램

```
public void startAgent() {
  if (shoplist == null) {
    shoplist = getTrader().getProvidersOf
               ("BuyFlowers");
    go(shoplist[1]);
    break;
  }
  if (shoplist[shoplistindex].askprice(flowers)
      < bestprice) {
    bestprice = shoplist[shoplistindex].
                askprice(flowers);
    bestshop = shoplist[shoplistindex];
  }
  if (shoplistindex >= (shoplist.length - 1) {
    // remote buy
    buy(bestshop, flowers, wallet);
    // go home and deliver wallet
    go(home);
    if (location.getAddress() = home) {
      location.put(wallet);
    }
  } go(shoplist[++shoplistindex];
}}
```

예를 들어 호스트는 이동 에이전트의 수행을 지연할 수 있고 변경된 내용으로 수행할 수도 있다. 따라서, 이동 에이전트의 정적, 동적 코드는 모두 호스트에게 드러나게 된다. 이러한 문제점을 설명하기 위해 예제로 작은 상품구매 에이전트를 사용하겠다. 이동 에이전트는 초기화된 이후 다음과 같이 동작한다.

유해한 호스트에 대해 가능한 여러 가지 공격 중에서

두 가지만 살펴보도록 하겠다. 첫 번째는 "전자지갑의 도난"이다. 만약에 wallet이란 변수 안에 전자적으로 표현되는 비트 스트링을 알 수 있다면 공격자는 돈을 사용한 후 단순히 변수를 수정할 수 있을 것이다. 이동 에이전트는 이러한 공격을 즉시 감지할 수 없고 다른 호스트로 이동 한 후에야 돈이 사라진 것을 알게될 것이다.
다음으로는 에이전트 고유 임무와는 별개인 "다른 주문 주입"이다. 여기에는 두 가지 방법이 있을 수 있다. 첫 번째는 강제로 bestshop의 내용을 조작할 수 있는 가능성이다. 호스트의 임의대로 bestshop을 변경한 후 shoplistindex의 내용을 변경한다면 이동 에이전트는 변경된 이동지로 옮겨갈 것이다. 두 번째는 코드의 수행을 순차적으로 진행하지 않고 변수를 변경 후 go(home)을 호출하는 것이다. 이러한 공격은 소스코드를 변형하지 않고 수행할 수 있으므로 공격여부를 알 수 없다. 또한 코드 자체를 수행하지 않고 방치하여 영원히 이동 에이전트 코드를 수행하지 않는 공격이 있을 수 있다.

1.3 개선 방안

위에서 언급한 것처럼 이동 에이전트 시스템 과 이동 에이전트 자체에 대한 여러 가지 보안적 문제점이 발견되었고 이를 해결하기 위한 여러 연구가 개발되고 있다. 그러나, 이동 에이전트 시스템에 대한 연구는 활발하게 진행되었으나 유해한 호스트로부터 이동 에이전트를 보호하는 연구는 그렇지 못한 실정이다[2][9]. 이 논문에서는 이동 에이전트의 보안적 취약점을 점검하고 그에 대응되는 방안을 제시하고 이를 구현하고자 한다. 무엇보다도 이동 에이전트의 코드 변경에 따른 사용자의 피해를 최소화하고 나아가 민감한 데이타를 안전하게 처리할 수 있는 방안으로 MAG(Mobile Agent Gateway)[6]을 구현하는 방안에 대해 언급하고자 한다. 이는 이동 에이전트 코드를 분석하여 공격에 대한 가능성을 타진하고 이에 따라 이동 에이전트 게이트웨이를 통해 안전하게 송수신하는 방안에 대해 언급하고자 한다.

2. 관련연구

지금까지 이동 에이전트 보안에 대하여 많은 연구가 이루어졌으나 본 논문에서는 이동 에이전트의 코드를 암호화하여 처리하는 연구에 대해 살펴보겠다.

2.1 분리되지 않는 서명(Undetachable Signatures)

분리되지 않는 서명 방법은 Sander와 Tschudin에 의해 제안되었다[10][11]. 그리고, 이는 CEF(Computing with Encrypted Function)라 불리는 방법에 기초를 두고 있다. 여기서 이동 에이전트는 s를 자신의 서명 함수로 대동하며 호스트는 이 함수를 통해 호스트의 결정을 서명하게 된다. 그러나 이 서명 함수는 호스트에게 드러나게 되므로 이를 암호화된 함수 f를 통해 암호화하여 s를 참조하는 대신 $s \circ f$를 이용하여 일을 처리하게 한다. 예를 들어 소비자는 인터넷을 통하여 물건을 구매하기 위하여 이동 에이전트를 보내고자 한다고 가정하자. 이때 소비자의 서명 함수 s를 사용할 수 있을 경우만 에이전트는 트랜젝션을 인증한다. 그러나, 에이전트는 잠정적으로 유해한 호스트에서 수행될 수 있다. 따라서 이 s를 보호하기 위하여 다음과 같은 암호화 함수를 이용하여 호스트는 s를 직접적으로 알 수 없게 한다. 즉, 서명을 직접 이용하지 못하며 특정한 암호화 함수를 통해서만 결과를 얻을 수 있다.

$f_{signed} = s \circ f$ (1) - 암호화된 함수를 통해서만 서명 함수를 접근가능(s : 서명 함수, f : 암호화된 함수, o : 합수 결합)

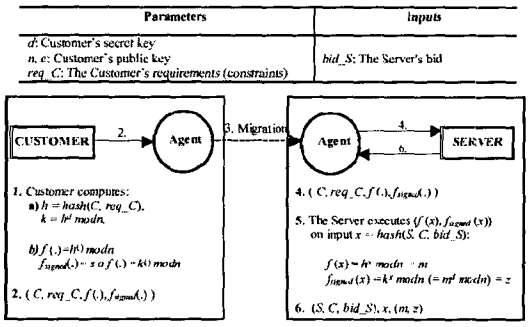

그림 3 RSA를 이용한 안전한 서명 스키마

이러한 암호학적인 방법을 통해 에이전트의 보안성을 개선시켰다. [그림 3]은 RSA 알고리즘을 이용한 전체적인 구조를 나타내고 있다. 즉, 이동 에이전트의 역할을 수행하고 난 후 그에 해당하는 데이타를 호스트에서 제공하게 된다. 이때 이 데이타는 평문형태로 부여되는 것이 아니라 서명을 통해서만 접근하겠다는 방법이다. 또한 서명함수를 이용할 때 이동 에이전트가 제시하는 암호화된 함수를 통해서만 서명이 가능하므로 직접적으로 서명을 할 수 없으므로 데이타 값의 변경이 불가능하다는 장점을 지닌다. 그러나, 이러한 암호화된 서명을 통한 방법은 이동 에이전트의 여정 리스트를 변경하거나 또는 암호 모듈을 전체적으로 다른 암호 모듈로 대체하였을 경우의 단점이 드러나게 된다. 또한 모든 데이타를 암호화 할 수 없으므로 인해 생기는 보안상의 문제가 여전히 남아 있게 된다. 본 논문에서는 이러한 문제를 해결하기 위해 호스트의 결정을 에이전트 자체에서 결정하지 않고

보다 안전한 이동 에이전트 게이트웨이와 인증을 수행한 후 결정하도록 해결하였다. 즉, 호스트는 서명 등의 암호 연산을 통해 해당 암호모듈을 MAG을 통해 인증한 후에야 이용할 수 있도록 변경할 경우 암호모듈을 대체하는 공격에 대해 개선을 할 수 있다.

2.2 스마트 카드를 이용한 에이전트 보호

에이전트의 보호를 위하여 스마트카드를 이용하는 방법이 제안되었다[3]. 이 방법은 에이전트의 보호를 위해 에이전트에 대한 모든 참조 및 연산 등을 컴퓨터상에서 수행하지 않고 스마트카드 내부에서 수행한다. 예를 들어 자바카드와 같은 신뢰할 수 있는 카드를 이용하여 다음과 같은 기능에 따라 수행된다.

- 암호화된 코드 부분을 입력받는다.
- 카드 내에서 데이타를 복호화하고 수행되는 경로상에 데이타를 저장한 후 수행한다.
- 다음 이동될 카드의 키를 이용하여 결과를 암호화한 후 에이전트에게 전송한다.

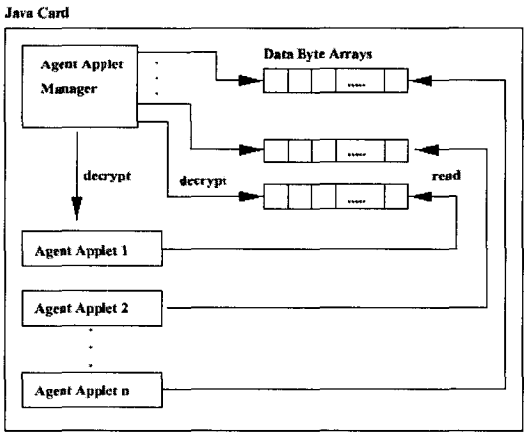

그림 4 스마트카드의 구조

[그림 4]는 이러한 스마트 카드 내부의 구조를 나타내고 있다. 스마트카드를 이용할 경우 외부로 복호화된 데이타가 드러나지 않으므로 신뢰성 있는 에이전트 수행을 기대할 수 있다. 또한 다른 하드웨어 장비에 비해 저렴한 가격과 보편화되어 있다는 장점을 가진다. 그러나, 이러한 스마트 카드를 이용하는 방법은 다른 스마트카드간의 상호적인 키를 알아야 하므로 중앙 집중적인 방법을 통해 관리를 해야 하고 또한 모든 관계된 키를 추가 변경이 생길 때마다 갱신해야 하는 번거로움이 있다. 또한 하드웨어의 처리 능력이 다소 떨어지므로 고속연산이나 큰 데이타를 처리하는데는 다소 문제가 있다고 할 수 있다.

3. 이동 에이전트 게이트웨이(MAG-Mobile Agent Gateway)의 구현

이동 에이전트 보안에 대해 제시된 이론들은 이동 에이전트에 대한 공격을 줄일 수는 있으나 완벽하게 보안을 달성할 수 있는 방안은 아니다. 또한 코드상으로 많은 오버헤드를 감수해야만 한다. 현실 세계에서 다수의 사용자가 호스트를 이용할 경우 호스트에 과중한 부하가 생길 수도 있다. 이 논문에서는 이러한 단점들을 보완하기 위하여 이동 에이전트와 이동 에이전트 시스템 이외에 추가적인 개체로 이동 에이전트 게이트웨이를 제안하고 이에 대한 구현에 대해서 논의하고자 한다.

3.1 이동 에이전트 게이트웨이

MAG은 내부망을 보호하기 위하여 내부망의 게이트웨이 역할을 수행하며 또한 암/복호화를 지원하기 위해 암호 모듈을 탑재하였다. MAG 시스템에서 필요한 구성요소를 다음과 같이 정의한다.

- 클라이언트 : 이동 에이전트를 생성하여 이를 통해 사용자의 역할을 대신 수행하고자 하는 사용자 또는 시스템
- 에이전트 풀 : MAG을 통해 전송된 원본 이동 에이전트를 일시적으로 보관하는 데이타베이스로 에이전트의 분할 및 병합에 사용
- 보안정책(SPD-Security Policy Database) : 보안정책 데이타 베이스로 이동 에이전트에 적용할 규칙들을 저장하는 데이타베이스
- 관리 시스템 : 보안정책 및 에이전트 풀 및 MAG 등을 관리하는 전체적인 운영/관리 시스템
- 호스트 : 이동 에이전트를 전송 받아 수행시키며 이동 에이전트의 역할에 대해 응답해 주는 일종의 서버 시스템

[그림 5]는 MAG의 전체적인 시스템의 구성도를 나타내고 있다[6].

3.2 시스템 구성 및 기능

이동에이전트 게이트에 대한 전체 시스템 구성 및 기능은 다음과 같다.

a. 모든 클라이언트들의 통신은 MAG 장비를 통해서 외부와 연결되며 일종의 게이트웨이 역할을 수행한다.
b. MAG는 Linux 시스템을 탑재하여 이동 에이전트를 모듈 또는 코드 형태로 인식할 수 있는 인터프리터 기능을 내장하고 있다.
c. MAG는 이동 에이전트들을 임시적으로 저장할 수 있는 이동 에이전트 풀을 운영한다. 이러한 풀은 SQL을 통해 데이타를 주고받으며 관리 시스템을

통해 모니터링할 수 있다.

d. MAG은 새로운 이동 에이전트를 생성할 수 있으며 보안 정책에 따라 적용 받는다.

보안정책은 다음과 같은 구성 요소로 구성된다.

- 각 클라이언트에서 적용할 이동 에이전트의 보안 클래스
- 각 클라이언트가 방문한 호스트들의 리스트 및 위험성 여부
- MAG에서 테스트 이동 에이전트를 통해서 등록한 호스트들의 리스트 및 위험성 여부
- 각 보안 클래스에 따른 보안설정 데이타

위의 데이타는 클라이언트들의 설치 시점에 관리시스템을 통해 준비되고 운영 시점에 점진적으로 추가 및 갱신된다

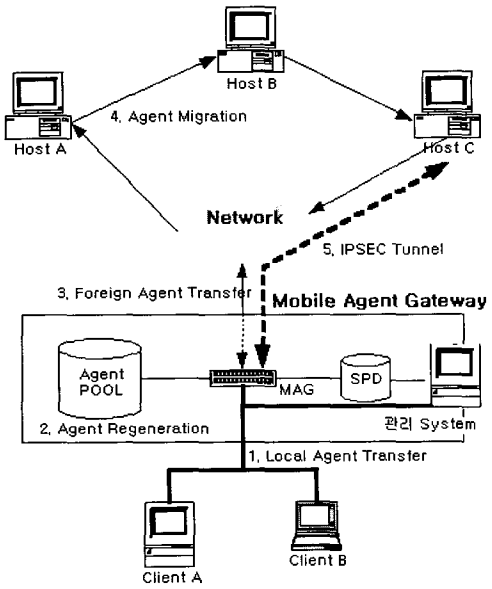

그림 5 이동 에이전트 게이트웨이

3.3 MAG의 구조

이동 에이전트는 이동 에이전트 시스템에서 수행하는 동안 코드를 통해 호스트의 자원을 참조한다. 이동 에이전트는 목표한 결과를 얻었을 경우 내부적으로 상태를 변경시킨 후 여정 리스트에 따라 다음 목적지로 이동한다. 이때 ATP(Agent Transfer Protocol)를 통해 다음 호스트로 전송된다. 이러한 환경 속에서 만약 호스트가 불법적으로 상태를 변경하거나 여정 목적지를 강제로 변경할 경우가 발생할 수 있다. 또한 내부 데이타 중에서 민감한 데이타가 포함되어 있을 경우 문제는 더욱 심각해진다. 따라서 MAG에서는 이러한 단점을 보완하기 위하여 에이전트를 안전한 최소단위로 분할하고 민감한 데이타를 제거하여 호스트로 전송한다. [그림 6]은 이동 에이전트 게이트웨이의 내부구조를 나타내고 [그림 7]은 구성요소별 상관관계에 대해 나타내고 있다. 이동 에이전트는 사용자의 PC를 떠나 이동 에이전트 게이트웨이에 도착하게 되고 그림 6과 7의 구조에 따라 이동 에이전트를 처리하게된다.

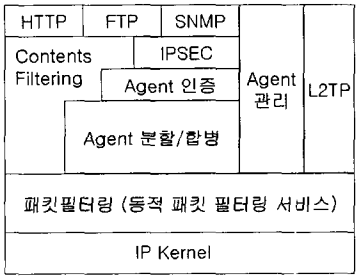

그림 6 구현된 MAG의 내부구조

다음은 이동 에이전트 게이트웨이에서 에이전트를 처리하는 과정을 기술한다.

- 클라이언트로부터 전송된 에이전트에 대해 인증을 수행
- 에이전트 내의 클래스 정보에 따라 보안정책을 적용
- 적용된 보안정책에 따라 여정리스트를 분할하여 에이전트 재구성
- 에이전트의 클래스에 따라 데이타 내부에 정보를 에이전트 풀에 에이전트와 함께 저장
- 재생성된 이동 에이전트에 인증정보와 IPSec정보를 추가하여 호스트로 전송

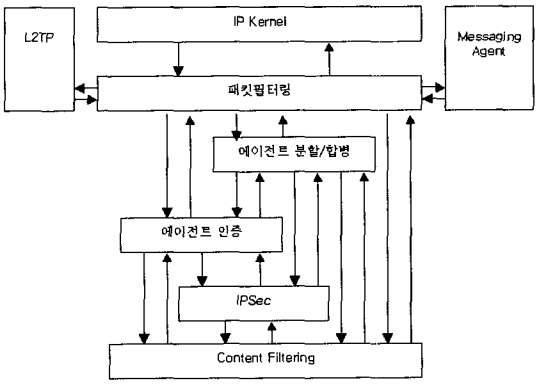

그림 7 구성요소별 상관관계

위의 과정을 거친 이동 에이전트는 호스트에 도착한 후 IPSec정보를 통해 MAG과 통신을 수행하게 된다. 이때 에이전트 풀에 저장된 데이타를 안전하게 호스트로 전송하게 되며 호스트 또한 에이전트에게 부여할 결과 데이타를 MAG으로 보내게 된다.

이동 에이전트는 MAG으로 귀환한 후 에이전트 풀에 있는 원본 에이전트의 내용을 토대로 다시 합병을 수행한다. 합병을 마친 후 MAG은 호스트로부터 전송 받은 결과 데이타를 에이전트 데이타에 첨가하고 다시 서명한 후 클라이언트에게 전송한다.

3.3.1 패킷 필터링

패킷필터링에서는 정해진 규칙에 따라 수신된 패킷을 필터링하는 기능을 수행한다. IP 커널을 거친 모든 패킷은 특정한 패킷(Broadcasting 패킷, Non-IP packet 등)을 제외하고는 모두 패킷필터링을 거치게 된다. 여기에서 특정한 패킷은 수신한 직후에 바로 미리 설정된 설정 사항에 따라 바이패스를 시킬 것인지, 또는 버릴 것인지를 결정한다. 패킷필터링에서는 정해진 규칙에 따라 패킷을 필터링하는 일반적인 필터링 부분과, 정해진 규칙으로부터 파생된 부가적인 규칙을 이용하여 동적으로 패킷을 필터링 할 수 있는 동적 패킷필터링이 있다. 이러한 패킷 필터링 기능을 이용하여 이동 에이전트에 대한 필터링을 수행한다.

3.3.2 에이전트 코드 분할/합병 및 재생성

이동 에이전트는 사용자의 요구를 대신하여 작업을 수행하는 자율적인 프로그램으로 요구에 부합되는 결과를 얻었을 경우 특정한 행동을 취할 수 있다. 예를 들어 예약과 관련되어 신용카드의 번호를 호스트에게 제공한다든지 등의 작업을 수행할 것이다. 이 데이타는 호스트의 공개키로 암호화되어 해당 비밀키를 소유한 호스트만 복호화 할 수 있다. 그러나, 이 과정에서 호스트는 이러한 민감한 정보를 악용할 우려가 있다. 실제 예약을 수행할 수 없는 조건을 가진 호스트도 이 정보를 알 수 있다는 것이다. 따라서 얼마든지 이러한 정보는 유출가능하고 악용될 우려가 있다. 이러한 문제를 해결하기 위해서 이 논문에서는 민감한 정보가 포함되었을 경우 MAG에서 이를 보관한 후 최종적으로 결정된 경우에만 이를 코드에 적용하여 실제 작업을 수행하게 한다. 그리고, 이동 에이전트를 생성하는 과정에서 코드에 대한 민감성 여부를 판단하기 위해 코드를 다음과 같은 클래스 형태로 분류한다. 이와 같은 분류는 관리자 및 사용자의 설정에 따라 조절 가능하도록 이동 에이전트 생성 프로그램 및 이동 에이전트 게이트웨이에서 사용자 인터페이스로 제공한다. 이러한 분류 이외에도 다른 기준으로 코드를 분할할 수도 있을 것이다.

- 신상 정보 코드(Personal Data)
- 보호 코드(Sensitive Data)
- 일반 코드(General Data)

'MAG는 클라이언트로부터 이동 에이전트를 수신한 후 분석기를 통해 원본 코드를 분류한다. 구분된 코드는 보안정책상에 클라이언트의 정책에 따라 재구성된다. 이러한 정책은 관리자에 의해 설정 가능한 정보로 유지한다. 이 과정을 통해 보안정책상에 외부로 유출을 금지한 코드를 포함하였을 경우 코드 상에서 제외되고 안전한 코드만을 이용하여 이동 에이전트는 재구성된다. 원본 코드는 마지막 결정을 위해 에이전트 풀에 위치시킨다. 또한 코드를 재구성할 때 여정리스트를 보안정책 리스트에서 검사하여 위험한 호스트로 등록이 되어있을 경우 이를 분할한다. 즉, 위험 가능성이 있는 호스트일 경우는 다른 호스트와의 연계성을 배제함으로 인해 보다 안전한 통신을 하기 위함이다. 이러한 불법적인 호스트의 등록은 테스트 이동 에이전트를 통해 점진적으로 구축해 나간다. 이렇게 재구성되고 분할된 이동 에이전트는 외부 호스트로 이동되어 작업을 수행한다. 다음은 에이전트 코드 분할 및 재생성 알고리즘이다. 이러한 코드는 JAVA를 이용하여 설계하였다.

- 이동 에이전트 분할 및 합병 알고리즘

```
SplitAgent()
{ Recieve_Agent_Data(pData); // MAG에서 데이타를
                    // 처리하여 에이전트 코드를 전송
  pList = Get_Itinerary_List(pData);
  Get_SPD_Data(pSpdData); // SPD 데이타 적재
  for (int I = 0; I < pList->nCount; I++) {
      if (IsValidSplit_Agent(pList[i], pSpdData))
          pNewAgent[] = Split_Agent(pList[i]);
  }
  Send_Agent(pNewAgent); // 에이전트 재구성&전송
}
MergeAgent()
{
  Recieve_Agent_Data(pData); // 외부로부터의 전송
  if (IsMerge_Agent(pData, pSpdData))
      AddingAgent(pOrgAgent, pData);
  if (pOrgData->IsEnd)
      Send_Agent_To_Entity(pOrgData);
}
```

3.3.3 이동 에이전트 인증

이동 에이전트 인증은 망을 사용하는 대부분의 패킷에 대해 IP 주소를 기반으로 통신을 제어하던 기존 방식과 달리 사용자 인증서 즉, PKI를 기반으로 통신을 제어하는 기능이다. 사용자는 자신의 인증서를 이동 에이전트에 일임함으로 인해 이동 에이전트가 대리인의 역할을 함과 동시에 전자서명의 특성을 가질 수 있도록 인증서 기반의 인증을 수행한다. 인증서는 X.509에서 지정한 표준 인증서를 준용한다. 따라서 이동 에이전트의 코드에는 다음의 형식과 같은 메시지 형식을 따른다.

Mobile Agent Code	$_{sign}$Client

3.3.4 IPSec

IPSec(Internet Protocol Security)은 TCP/IP 프로토콜의 구조적인 결함을 극복하고 IP 수준에서 제공되는 보안 서비스를 표준화할 목적으로 개발된 인터넷 보안 표준이다. 이동 에이전트에서는 통신 데이타의 비밀성과 무결성 등을 보장하기 위하여 IPSec 기능의 암복호화 기능을 이용하여 외부의 호스트로 전송된 후 데이타에 대한 송수신을 IPSec을 통하여 이동 에이전트 게이트웨이와 수행한다.

이 때 송/수신되는 정보는 이동 에이전트가 호스트에게 제공해야 하는 민감 데이타와 호스트로부터 이동 에이전트에게 전송할 결과데이타를 암/복호화여 호스트와 MAG 사이에 진행된다. 이 논문에서는 IPSec을 이용한 암호화 프로토콜을 사용하였으나 SSL과 같은 프로토콜로 변형하여도 가능할 수 있다.

3.4 에이전트 이동 및 수행

이동 에이전트의 코드는 MAG에서 재생성시 클래스에 따라 민감한 데이타를 제외한 상태이므로 에이전트가 결정을 내려야할 사항 즉, 사용자 요구에 대한 결론에 도달하였거나 또는 마지막 호스트에서 홈으로 귀환할 경우 MAG과 송/수신을 수행한다. 이때 이동 에이전트는 MAG과의 안전한 통신을 위한 SA(Security Association) 데이타를 호스트에게 할당하며 이를 통하여 호스트는 MAG과 보안 통신을 수행할 수 있다. 이때 SA 데이타는 MAG에서 사용되는 프로토콜 및 암호화에 대한 정보를 포함하는 것으로 이동 에이전트의 고유 기능인 사용자 주문형태의 데이타를 수행할 수 있는 방안이다. 즉, 호스트의 이동 에이전트의 내부에 [그림 8]과 같은 내용을 포함하고 SA 데이타는 MAG에서 운영되는 데이타의 모듈을 탑재한다. 이 논문에서는 IPSec을 위한 IKE(Internet Key Exchange)를 탑재하였다[12][13].

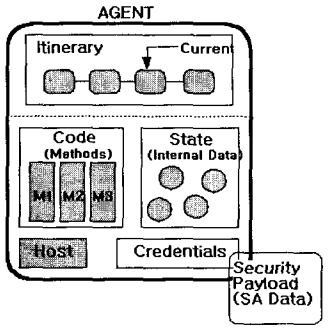

그림 8 재구성된 에이전트 구조

IKE는 두 개체간의 암/복호화를 위한 세션키를 공유하는 키 교환 프로토콜로 이를 통해 안전하게 호스트와 이동 에이전트 게이트웨이 사이의 키를 공유하고 호스트는 자신의 서명값을 전송하고 함께 IPSec을 위한 난수값을 함께 보낸다. IKE를 통한 보안협상을 통해 보안 채널(Tunnel)이 생성된 후 호스트는 결과 데이타를 암호화하여 MAG에게 전송하고 호스트는 필요한 데이타 즉, 에이전트가 MAG을 거쳤을 때 풀에 보관한 민감 데이타를 MAG으로부터 받는다[12]. 이는 상호 인증을 거친 후 보안정책상에 정해진 규칙에 따라 즉, AH (Authentication Header) 또는 ESP(Encapsulating Security Payload)를 수행한다[14][15]. 이때 MAG은 호스트가 필요로 하는 데이타 즉, 이동 에이전트 풀에 저장된 원본 데이타 중 민감한 데이타를 안전하게 전송한다. 전송된 데이타를 통해 호스트는 사용자의 원하는 작업을 종료할 수 있다. 호스트와 MAG 사이의 프로토콜은 [그림 9]와 같다. 상호 인증을 위해 서명값을 사용하고 암호화 프로토콜을 위해 난수를 생성하여 추가적으로 송/수신한다. [그림 10]에서 알 수 있듯이 네트웍을 통해 송신 또는 수신되는 이동 에이전트는 응용 단계에서 패킷 필터링에 의해 에이전트임을 감지하게 되면 보안정책 서비스에 따라 적용할 클래스를 적용 받고 여기에 IPSec을 위한 데이타를 첨가한 후 분할 또는 합병 등의 연산을 거치고 분할의 경우 원본 데이타는 에이전트 풀에 저장하여 차후 재생성 할 경우 이용한다

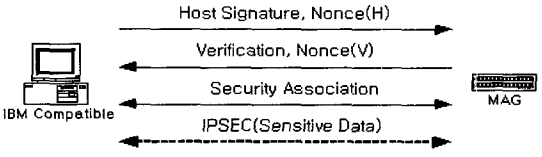

그림 9 호스트와 MAG사이의 암호화 프로토콜

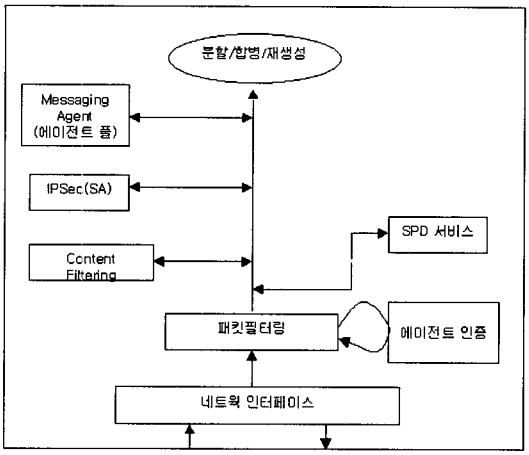

그림 10 이동 에이전트 보안기능 수행과정

[그림 9]에서 호스트의 서명(Signature)은 SSL과 같은 형태의 프로토콜에서 사용하는 전자서명 방법을 사용하기 위하여 인증서 기반을 사용한다. 인증서는 X.509에서 정의한 표준에 따르는 인증서를 사용한다. 난수는 각각 호스트 및 MAG에서 둘 사이의 세션키를 만들기 위한 매체로 사용된다. 다음으로 보안 협상과정은 SA를 설정하는 과정으로 즉, 사용할 알고리즘과 키의 크기 및 ESP 또는 AH와 같은 통신 프로토콜을 정의한다. 이 과정은 ISAKMP(Internet Security Association and Key Management Protocol에 따라 설계하였다[12]. 여기서 키교환에 대한 인증으로 인증서를 사용하였고 이는 IKE 프로토콜을 이용하여 사용하였다.

3.5 에이전트 작업 종료

위의 과정을 통해 호스트와 MAG간의 안전한 통신을 통해 이동 에이전트 고유의 목표를 달성하게 된다. 이때 호스트들에 대한 위험성 여부를 판단한다. 이는 원본 이동 에이전트의 리스트를 파악한 후 귀환되는 에이전트를 분석하여 얻을 수 있다. 즉, 정상적인 여정을 마친 이동 에이전트에는 해당 호스트의 서명이 포함되어 있고 만약 리스트를 다 여행하지 못했을 경우에는 마지막 호스트의 행위를 의심할 수 있다. 이를 통해 보안정책안의 호스트 리스트를 갱신하고 또한 이동 에이전트 풀에서 원본 데이타를 적재하여 이에 대한 응답을 클라이언트에게 안전하게 전송한다. 따라서 클라이언트는 이동 에이전트 본래의 목적에 부합되는 한 번의 전송을 취하여 안전하게 결과를 받을 수 있고 민감한 데이타를 포함하였을 경우에도 이동 에이전트에 대한 보안성을 유지할 수 있다.

3.6 분석 및 시험 결과

기존의 이동 에이전트에 대한 보안으로 마련된 분리되지 않은 서명의 방법과 이 논문에서 제안한 MAG상에서의 IPSec과의 보안성은 대동소이하다고 볼 수 있다. 하지만 기존의 방법은 이동 에이전트 내부에 항상 암호화 모듈을 추가하여 움직여야하는 단점을 가지고 있다. 하지만 MAG을 이용할 경우 한 번 방문한 호스트에 대해서 인증정보을 가지고 접근하므로 다음 방문 때는 ISAKMP에서 제안하는 세션키 생성 과정 만 수행하면 되고 따라서 부가적인 암호코드에 대한 부담을 줄일 수 있다. 또한 이동에이전트가 단순한 데이타를 처리하였을 경우에는 상관이 없지만 많은 양의 데이타 즉, 데이타베이스와 연관된 작업을 수행할 경우 에이전트 자체에 모든 데이타를 추가하여 전송하기란 거의 불가능하다. 따라서 이러한 관점에서 볼 때 제3의 매체인 MAG을 통하여 안전한 통로를 확보하고 이를 통해 에이전트 고유의 작업을 수행하는 방안을 제시하였다. 그리고 하드웨어를 사용하여 이동 에이전트의 보안을 수행하는 방법은 호스트들이 연계되어 하드웨어에 대한 정보를 공유해야하는 관리적 차원에서도 문제점이 발견되었다.

본 이동 에이전트 게이트웨이에 대한 평가 방법으로 SmartBit라는 사용 제품을 이용하여 테스트하였다. SmartBit는 패킷을 일정한 간격에 맞추어 네트워크 상에 연결된 장비에 전송하는 장비로 가상의 이동 에이전트를 유형 별로 생성하여 이동 에이전트 게이트웨이에 전송한다. 다음 그림은 이동 에이전트의 클래스 별로 SmartBit에서 초당 1000개의 패킷을 일정 간격으로 전송하였을 때의 결과를 그래프로 나타내고 있다.

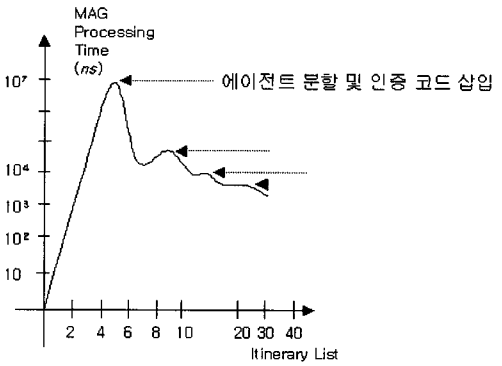

그림 11 여정 리스트의 포함수에 따른 MAG의 처리 속도

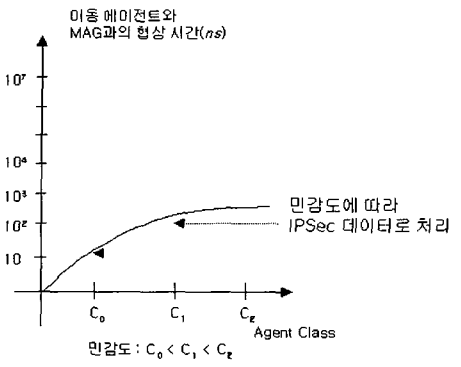

그림 12 이동 에이전트 민감도에 따른 통신 속도

위와 같은 결과에서 알 수 있듯이 [그림 11]처럼 이동 에이전트 내부에 많은 수의 여정 리스트를 가질수록 초기에는 에이전트 분할과정에 따른 시간이 많이 걸리는 것을 알 수 있다. 하지만, 시간이 지남에 따라 데이타베이스에 여정 리스트에 대한 위험성 여부가 적재되고 이를 이용하여 여정 리스트가 증가하여도 처리속도는 오히려 감소하는 것을 알 수 있다. 또한 [그림 12]에서 알 수 있듯이 이동 에이전트의 클래스의 민감도에 따라 속도가 증가하는 것이 아니라 호스트에 안착한 이동 에이전트와 MAG 간에 IPSec을 통한 안전한 채널이 생성될 경우 민감한 데이타의 양과는 관계없이 거의 일정한 속도를 내는 것을 알 수 있다.

4. 결 론

기존의 분산환경에서 대두된 이동 에이전트는 불법적인 호스트들에 대한 위험성 때문에 실제 응용에 적용하는데 많은 문제점이 있었다. 본 논문에서는 안전한 제3의 개체인 MAG을 통해 이러한 이동 에이전트의 보안상의 문제점을 해결하였고 그에 따른 부담을 클라이언트에게는 투명하게 유지하였다. 따라서, 불법적인 호스트로부터 클라이언트의 이동 에이전트를 안전하게 유지할 뿐만 아니라 코드를 재생성함으로 인해 보다 민감한 데이타가 유출되는 것을 막을 수 있다. 이는 현재 가장 많이 네트워크 상에서 사용되는 VPN 장비 등을 이용하여 구축함으로 인해 시설 추가에 따른 부담을 줄일 수 있다. 하지만, 제3의 개체를 더 경유해야 하는 추가 부담으로 인한 오버헤드는 여전히 단점으로 남고 있다. 이를 좀 더 개선하기 위한 방안이 필요하겠다. 또한 보안정책에 대한 점진적이고 적응적인 구축이 더 연구되어야 하며 제3의 매체인 MAG의 처리능력에 따라 이동 에이전트 시스템 뿐만아니라 이를 게이트웨이로 이용하는 내부망 전체 시스템에도 막대한 영향을 초래할 수 있다는 점에서 MAG의 성능 개선은 필수 사항이라 하겠다. 또한 제3의 개체를 경유하는 경우 오버헤드에 관하여 정량적인 분석이 향후 연구를 통해 추가 되어야 하겠다. 마지막으로 결과에서도 알 수 있듯이 상호인증을 위한 인증서 기반의 서명 검증 등의 공개키 연산이 필수적이여서 MAG 시스템에 공개키 가속기 등을 설치하는 방안도 검토해야 할 것이다. 하지만, 무엇보다도 이러한 이동 에이전트 자체에 대한 보안성에 대한 이론적 정립이 미흡하고 해결방안이나 테스트에 대한 인지도가 낮다는 현실도 간과해서는 안될 것이다.

참 고 문 헌

[1] D.C, Chess, C. Harrison, A. Kershenbaum, "Mobile Agents:Are They a Good Idea?," IBM Research Report, 1995.

[2] R. S. Gray, "Agent Tcl: A flexible and secure mobile-agent system," Proceedings of Fourth Annual Usenix Tcl", Workshop, pp.9-23, 1996.

[3] Stefan Funfrocken, "Protecting Mobile Web-Commerce Agents with Smartcard," Mobile Security Agent, LNCS 1419, 1999, springer, pp.44-64, Giovanni Vigna (Ed.).

[4] Crystaliz Inc., General Magic Inc., GMD Fokus, IBM Corp., "Mobile Agent Facility Specipication," Joint Submission Sup- ported by the Open Group, OMG TC Document, November 1997.

[5] F. Hohl, "A model of Attacks of Malicious Hosts Against Mobile Agents," presented at 4th Workshop on Mobile Object Systems Secure Internet Mobile Computations, France, 1998.

[6] Jae-Kyoung Park, Yoo-Hun Won, "Protecting Mobile Agent with VPN," 2001. 6, 情報保護學會論文誌 第 11券 第3號.

[7] Antonio Corradi, Rebecca Montanari, Cesare Stefanelli, "Security Issues In Mobile Agent Technology," http://www.lia.deis.unibo.it/Software/SOMA

[8] Jian Tang, Jari Veijalainen, "Using Agents to Improve Security and Convenience in mobile E-Commerce," Proceedings of the 34th Hawaii International Conference on System Science-2001, 2001 IEEE.

[9] Niranjan Suri, "NOMADS:Toward a Strong and Safe Mobile Agent System," Agent 2000 Barcelona Spain, pp.163-164, ACM 2000.

[10] Hartmut Vogler, Thomas Kunkelmann, Marie-Louise Moschgath, "An Approach for Mobile Agent Security and Fault Tolerance using

Distributed Transact- ions," Proceedings of the 1997 Inter- national Conference on Parallel and Distributed Systems, pp.268-274,1997 IEEE.

[11] Panayiotis Kotzanikolaou, "Secure Stansactions with Mobile Agents in Hostile Environments," pp.193-202, 1999 IEEE.

[12] D. Maughan, M. Schertler, M. Schneider, J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," November 1998, RFC 2408, available at http://www.cis.ohio-state.edu/Service/ rfc/rfc-text/rfc2408.txt

[13] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)," November 1998, RFC 2409, available at http://www.cis.ohio-state.edu/Service/rfc/rfc-text/rfc2409.txt

[14] S. Kent, R. Atkinson, "IP Authentication Header," November 1998, RFC 2402, available at http://www.cis.ohiostate.edu/Service/rfc/rfc-text/rfc2402.txt

[15] S. Kent, R. Atkinson, "IP Encapsulating Security Payload(ESP)," November 1998, RFC 2406, available at http://www.cis.ohio-state.edu/Service/rfc/rfc-text/rfc2406.txt

[16] Casey Wilson, Peter Doak, "Creating and Implementing Virtual Private Network," pp.379-418, 2000, CORIOLIS (Ed.).

박 재 경

1994년 2월 동국대학교 컴퓨터공학과 졸업. 1996년 2월 홍익대학교 전자계산학과 석사. 1996년 3월 ~ 현재 홍익대학교 전자계산학과 박사과정. 1997년 1월 ~ 현재 (주)퓨처시스템 정보통신 연구소 선임연구원. 관심분야는 네트워크 보안, Mobile Agent Security, VPN

원 유 헌

1972년 2월 성균관대학교 수학과(B.S). 1975년 2월 한국과학기술원(KAIST). 1975년 3월 한국과학기술 연구소 연구원. 1985년 2월 고려대학교 대학원 컴퓨터과학과(Ph.D). 1986년 3월 미국 RPI대학 교환 교수. 1976년 ~ 현재 홍익대학교 컴퓨터공학과 교수. 관심분야는 프로그래밍 언어, 객체지향 언어, 실시간 언어, 실시간 시스템, 멀티미디어 시스템, 정보보호