情報保護學會論文誌
第11券 第3號, 2001. 6

이동 에이전트 실행 보호를 위한 암호학적 추적 방안

신 원*, 이 경 현**

Cryptographic Traces for the Protection of Mobile Agent Execution

Weon Shin*, Kyung Hyune Rhee**

요 약

본 논문에서는 이동 에이전트의 시큐리티 취약성을 살펴보고 이를 보호하기 위한 기존 연구를 분석한 후, 이동 에이전트 보호를 위한 새로운 방안을 제안하였다. 제안 방안은 이동 에이전트 수행 결과를 보호할 뿐만 아니라 이동 에이전트 실행 추적도 가능하다. 또한, 이동 에이전트 실행에 대한 부인 방지, 전방향 무결성, 삽입 또는 삭제 공격 방지 특성을 지닌다.

ABSTRACT

In this paper, we describe the vulnerabilities against mobile agent and analyze existing schemes to cope with them. Also we propose a new scheme which protects the executed information by mobile agent and simultaneously traces the executions of mobile agent. The proposed scheme provides non-repudiation between participations for the executions of mobile agent, forward integrity against modification and protection against code insertion or deletion attack.

keyword : *Mobile Agent Security, Cryptographic Trace*

Ⅰ. 서 론

최근 인터넷 및 네트워크 사용자의 폭발적인 증가에 따라 관련 시장 역시 대규모로 성장하고 있다. 특히, 전자상거래의 보급으로 인하여 인터넷서비스에 대한 사용자의 요구가 높아지고 있으며 이를 수용하기 위한 다양한 어플리케이션들이 등장하였다. 그 중 이동 코드 기반의 이동 에이전트(Mobile Agent)는 이를 해결하기 위한 패러다임으로 새로운 컴퓨팅 환경으로 개발된 최신 기술이다.

이동 에이전트는 "현재 객체의 상태(Status)와 실행 가능한 코드(Executable Code)를 포함한 객체로써, 분산 환경 하에서 특정 서비스를 제공하기 위하여 호스트 간 직접 이동을 통하여 실행되는 객체"로 정의된다. 이러한 이동 에이전트를 사용하는 시스템은 인터넷과 같이 다양하고 이질적인 시스템들을 손쉽게 통합할 수 있고, 수행되는 프로그램이 상황에 따라 동적으로 대처할 수 있는 유연성을 제공할 수 있다. 따라서, 이동 에이전트 시스템은 네트워크로 연결된 독립적인 다수의 에이전트가 하나의 유기적인 집합으로서의 기능을 수행하는 시스템으로써 현재의 클라이언트/서버 시스템에서 발생하는 문제점을 해결할 수 있는 새로운 분산처리 방안으로 인식되고 있다.

그러나 이동 에이전트에 있어 이동성(Mobility)은 시스템에 대한 통합을 쉽게 해주고 여러 시스템에 있어 융통성을 제공하는 반면에 새로운 보안 문제를 발생시키고 있으며 이동 에이전트 시스템 구현에 있어 큰 장애요소로 등장하고 있다. 여기서, 이동 에이전트의 가장 큰 특징인 이동성은 Java[1],

* 부경대학교 전자계산학과
** 부경대학교 전자컴퓨터정보통신공학부

ActiveX[2], Telescript[3], Agent-Tcl[4] 등으로 대표되는 이동코드 기술에 그 기반을 두고 있으므로 이동 에이전트의 보안 문제는 궁극적으로 이동 코드의 보안 문제로 귀결된다고 볼 수 있다. 따라서, 이동 에이전트 시스템을 중심으로 이동 코드 기술에 의해 새롭게 등장하는 보안 문제를 다루고, 안전한 에이전트 기반 시스템을 구현하기 위한 체계적인 연구가 필요하다.

2장에서는 에이전트 보호를 위한 기존 연구에 대해서 논하고, 3장에서는 이동 에이전트 수행 결과 보호 및 실행 추적을 위한 방안을 제안한다. 4장에서는 제안 방안의 부정 검출 및 안전성 분석에 대하여 논의하고, 마지막으로 5장에서는 결론을 유도하고 향후 연구과제에 대하여 기술한다.

II. 이동 에이전트 시큐리티

1.1 이동 에이전트 위협

이동 에이전트가 임의의 호스트로 유입되어 실행하기 위해서는 실행 코드, 데이터, 상태 정보 등이 반드시 함께 이전되어야 한다. 호스트는 자신의 환경에서 에이전트를 직접 실행하기 때문에 에이전트 내의 모든 정보에 접근할 수 있으며 이를 악용할 수도 있다. 악의적인 호스트가 이동 에이전트에 가할 수 있는 위협은 다음과 같다.

- 도청(Eavesdropping) : 에이전트를 수행하는 호스트는 에이전트의 통신 및 에이전트 실행시 코드, 상태, 데이터 자체에도 액세스할 수 있으므로 에이전트가 가지는 중요 정보를 도청할 수 있다.
- 위장(Masquerade) : 호스트가 신뢰하는 목적지 호스트로 위장함으로써 에이전트를 속이는 경우이다. 에이전트는 안전한 장소로 생각하고 자신의 데이터 및 상태 정보를 악의적인 호스트에 모두 전달하게 된다.
- 변경(Alteration) : 도청보다 더 능동적인 공격으로 에이전트 내의 정보를 불법 수정하거나 다른 오동작을 유발하는 경우이다.
- 서비스 거부(DoS, Denial of Service) : 에이전트가 호스트에 도달하였을 때 에이전트의 요청에 의하여 호스트 자원을 제공해주어야 하지만, 악의적인 호스트는 이를 거부하고 에이전트의 정당한 수행을 방해하는 경우이다.

1.2 에이전트 보호를 위한 기존 연구

이동 에이전트의 수행은 다른 제 3자의 간섭을 받아서는 안되며, 이동 에이전트 내에 저장된 중요 상태 정보와 수행 코드 등이 이동 에이전트 외부로 누출되어서도 안된다. 그러나 이동 에이전트가 실행되는 서버는 실행환경을 제공하므로 만약 악의적인 목적을 갖는다면 에이전트 내부의 데이터 및 수행 코드까지도 접근할 수 있게 된다. 따라서 이에 대한 에이전트 보호 방안이 마련되어야 하는데 아직은 제한된 연구 만이 진행 중이다. 지금까지 수행되었던 연구는 보안 방지를 목적으로 하는 Blackbox Security와 Computing with Encrypted Functions 방안, 보안 검출을 목적으로 하는 Cryptographic Traces방안, 에이전트 실행 후 결과 보호를 위한 해쉬 체인을 이용한 수행 결과 보호 방안 등이 있다.

F.Hohl은 악의적인 호스트에 대해 이동 코드를 보호하기 위해 Blackbox Security 개념을 제안하였다[5]. 여기서 에이전트는 하나의 블랙박스로 취급되기 때문에 만약 어떤 시점에서 에이전트 코드가 공격을 받지 않는다면 그것의 입출력 동작만을 실제 공격자가 관찰할 수 있다는 것이다[6]. Hohl은 여러 변환 알고리즘을 통하여 실행 코드와 데이터 표현은 다르지만 같은 결과를 가지는 원래 에이전트에서 새로운 에이전트를 생성하도록 하였다. 그러나, Limited Blackbox Security 방법에 대해 해결해야할 두 가지 문제가 존재하는데, 첫 번째는 보호가 필요한 시간이 유용한 길이를 가져야 하는데 이것은 사용되는 어플리케이션의 시간에 의존한다. 두 번째는 변환 알고리즘으로부터 보호가 필요한 시간에 대한 하한값을 어떻게 결정하는가 하는 것이다.

T.Sander와 C.Tschudin은 이동 코드의 시큐리티 요구사항에 대한 일반적인 해결 방안으로 상호 동작이 불필요한 CEF(Computing with Encrypted Function)를 제안하였다. Sander와 Tschudin는 이를 "Mobile Cryptography"[6~7]라 하고 "네트워크에서 이동하는 실행 코드의 정보 보호 측면에 관련된 수학적인 기술의 연구"라고 정의하였다.

이 방법은 이동 에이전트의 함수를 암호화하여 실행하는 방법이다. 여기서, 암호화란 원래의 함수와 수학적으로 같은 의미를 가지지만 형태는 다른 함수를 찾아 원래 함수를 치환함으로써 원래 함수의 연산 과정을 보여 주지 않으면서 같은 결과를 얻는 것을 의미한다[6~7]. 그러나 이동 에이전트를 동작시키

기 위해 함수의 형태는 다르지만 동일한 결과를 생성하는 함수를 찾는 것은 사실상 힘들기 때문에 실제 구현에는 한계가 있다.

앞서 설명한 악의적인 호스트에 대한 에이전트의 정보 유출 방지에 목적을 두는 것과 반대로 G. Vigna는 이동 에이전트의 코드, 상태, 제어흐름에 대한 공격을 검출하는 것을 허용하는 실행 추적과 암호에 기반하는 메커니즘을 개발하였다[9]. 이 메커니즘은 어떤 가능한 에이전트 코드, 상태, 실행 흐름의 비합법적인 수정에 대한 검출을 목적으로 하고 있으며, Trace라 부르는 이동 코드의 실행 과정에서 수집되는 데이터 분석에 기반한다. 여기서, Trace는 코드 실행 확인을 위해 사용된다. 에이전트를 조작한 경우, 이 방법에서 에이전트의 소유자는 호스트가 주장하는 연산이 에이전트에 의해 수행될 수 없음을 증명할 수 있다.

또한, 에이전트 수행 후 그 결과를 보호하고자 하는 연구도 함께 수행되어 왔다. 이 방안은 에이전트를 안전하게 수행하는 것은 불가능하다는 전제에서 출발하여 호스트에서 수행을 마친 이동 에이전트에 저장된 데이터가 다른 호스트에 의해 위·변조되는 것을 막기 위한 방향으로 진행되어 왔다. "해쉬 체인을 이용한 방안[10]"은 이동 에이전트가 여러 서버를 지날 때마다 각 수행 결과에 해쉬함수를 적용하여 이동 경로 상에서 서버가 수행한 작업 결과들 사이에 연관성을 부여하는 방법으로, 각 수행 결과는 해쉬 체인을 이루므로 변조가 되었을 경우 해쉬 체인을 실제 수행하여 어떤 서버에서 변조되었는지 확인이 가능하다.

Ⅲ. 이동 에이전트 수행 결과 보호 및 실행 추적 방안

3.1 제안 방안

제안 방안에서 이동 에이전트는 에이전트 소유자 S_0로부터 출발하여 $S_i(1\le i\le n)$를 여행하는 것을 가정한다. 각 호스트에서 이동 에이전트는 소유자가 지시한 작업을 수행하기 위하여 p를 실행하는데, 여기서 p는 초기 상태 $p=\{c_0, s_0, t_0\}$에서 시작하여 각 호스트 S_i에서 m개의 단계로 이루어진 코드 $p=\{c_j, s_j, t_j\}(1\le j\le m)$를 의미한다. S_n까지의 여행을 마친 에이전트는 최종적으로 이동 에이전트 소유자인 S_0로 돌아가고, 부정이 발생한 것으로 생각될 때 수행 결과 보호 및 실행 추적을 위한 프로토콜을 동작한다.

각 호스트는 암호화를 위한 서로의 공개키를 알고 있고 서명을 위한 자신의 비밀키를 안전하게 소유하고 있다. 프로토콜 진행 방해를 위한 서비스 거부 공격은 없으나, 각 호스트는 에이전트 수행 결과 또는 코드 변경을 통한 공격은 할 수 있다고 가정한다. 본 논문에서 사용되는 용어들은 다음과 같다.

S_0 : 이동 에이전트 소유자
S_i : 이동 에이전트 실행 서버, $1\le i\le n$
ID_A : A의 ID
$E_K(m)$: 관용키 암호 시스템에서 K를 이용하여 m을 암호화
$S_A(m)$: 공개키 암호 시스템에서 메시지 m에 대한 사용자 A의 전자서명
$P_A(m)$: 공개키 암호 시스템에서 사용자 A의 공개키로 메시지 m을 암호화
$H(m)$: 메시지 m에 대한 해쉬값
$M_K(m)$: K를 이용한 m에 대한 메시지 인증값

제안 방안은 다음과 같다.

(1.1) 에이전트가 소유자 S_0로부터 첫 번째 호스트 S_1으로 이동

① $S_0 \rightarrow S_1 : S_{S_0}(ID_{S_0}, ID_{S_1}, AT), E_{K_{S_1}}(p, s_{S_0})$

에이전트 소유자 S_0는 자신의 신분 ID_{S_0}, 첫 번째 호스트 신분 ID_{S_1}과 함께 에이전트 토큰 $AT=(S_0, ID_{Agnet}, timestamp)$에 서명하고, 에이전트 코드 p와 초기 상태 s_{S_0}를 임의의 키 K_{S_0}로 암호화하여 전송한다. 여기서, ID_{Agnet}는 에이전트의 신분을, $timestamp$는 에이전트가 생성된 시각이다.

② $S_1 \rightarrow S_0 : S_{S_1}(ID_{S_1}, ID_{S_0}, ID_{Agent}, H(*), ack)$

호스트 S_1은 서명값을 확인하고 자신의 신분 ID_{S_1}, 에이전트 소유자 신분 ID_{S_0}, 에이전트 토큰 속의 에이전트 신분 ID_{Agnet}, 그리고 ①에 대한 해쉬값 $H(*)$과 함께 계속 프로토콜을 진행할지에 대한 응답 ack를 서명하여 보낸다. ack가 부정의 응답이라면 프로토콜은 중단된다.

③ $S_0 \rightarrow S_1 : S_{S_0}(ID_{S_0}, ID_{S_1}, H(p, s_{S_0}), P_{S_1}(K_{S_0}))$

에이전트 소유자 S_0는 참여자들의 신분 ID_{S_0}과 ID_{S_1}, p와 s_{S_0}의 해쉬값, 키 K_{S_0}를 S_1의 공개키로 암호화하여 전송한다. 호스트 S_1는 키 K_{S_0}를 사용하여 에이전트 코드 p와 초기 상태 s_{S_0}를 복호화할 수 있으며, 해쉬값을 통하여 무결성을 확인한 후 에이전트를 실행할 수 있다.

④ $S_1 \rightarrow S_0 : S_{S_1}(ID_{S_1}, ID_{S_0}, ID_{Agent}, H(*))$

호스트 S_1는 에이전트를 제대로 전송받아 실행할 수 있다는 응답으로 참여자들의 신분, 에이전트의 신분, ③에 대한 해쉬값을 전송한다.

(1.2) 호스트 S_1에서 이동 에이전트 실행

이동 에이전트가 여행할 각 호스트 S_i에서 p는 초기 상태 $\{c_{S_i}, s_{S_i}, t_{S_i}\}$에서 시작하여 m개의 실행, 상태, 추적값을 가지는 $\{c_j, s_j, t_j\}(1 \le j \le m)$의 단계를 거치면서 실행된다. 여기서, c_j는 에이전트 코드가 수행해야할 j번째 문장 또는 j번째 함수로 정의할 수 있다. s_j는 c_j를 수행하기 전 상태값으로 정의할 수 있고, t_j는 c_j를 수행한 후 부정 검출을 위한 추적값이다.

① $T_{S_1}(0) = \{c_{S_1}, s_{S_1}, t_{S_1}\}, M_{H_1}(\{c_{S_1}, s_{S_1}, t_{S_1}\})$

호스트 S_1에서 에이전트의 초기 상태 $\{c_{S_1}, s_{S_1}, t_{S_1}\}$와 H_1을 사용하여 메시지 인증값을 저장한다.

② $T_{S_1}(j) = \{c_j, s_j, t_j\}, M_{H_1}(\{c_j, s_j, t_j\}, T_{S_1}(j-1)), 1 \le j \le m$

S_1는 에이전트를 계속 실행하면서 각 상태 $\{c_j, s_j, t_j\}$와 H_1을 사용하여 메시지 인증값을 계속해서 저장하면서, m번 반복한다. 여기서, H_i는

$$H_i = \begin{cases} H(K_{S_0}) & \text{if } i = 1 \\ H(T_{S_{i-1}}) & \text{if } 1 < i \le n \end{cases}$$

와 같이 정의된다.

③ $T_{S_1} = \{T_{S_1}(j) \mid 0 \le j \le m\}$

모든 에이전트 코드를 실행하고 난 후 S_1는 T_{S_1}을 저장한다.

(2.1) 에이전트가 첫 번째 호스트 S_1으로부터 두 번째 호스트 S_2로 이동

참여자만 S_1, S_2로 다를 뿐 (1.1)의 과정과 거의 비슷하다.

① $S_1 \rightarrow S_2 : S_{S_1}(ID_{S_1}, ID_{S_2}, AT), E_{K_S}(p, s_{S_1})$

S_1은 에이전트 암호화 전송을 위해 임의의 키 K_{S_1}를 사용한다.

② $S_2 \rightarrow S_1 : S_{S_2}(ID_{S_2}, ID_{S_1}, ID_{Agent}, H(*), ack)$

(1.1)과 마찬가지로 ack가 부정의 응답이라면 프로토콜은 중단된다.

③ $S_1 \rightarrow S_2 : S_{S_1}(ID_{S_1}, ID_{S_2}, H(p, s_{S_1}), P_{S_2}(H(T_{S_1}), K_{S_1}))$

S_1은 에이전트 코드 p와 초기 상태 s_{S_1}을 복호화할 수 있도록 K_{S_1}을 전송하는데, 부가적으로 S_1에서 에이전트가 동작한 후의 T_{S_1}에 대한 해쉬값도 함께 공개키로 암호화하여 전송한다. 이 값은 S_2에서 에이전트 실행시 메시지 인증값을 생성하는 키로써 사용된다.

④ $S_2 \rightarrow S_1 : S_{S_2}(ID_{S_2}, ID_{S_1}, ID_{Agent}, H(*))$

호스트 S_2는 에이전트를 제대로 전송받아 실행할 수 있다는 응답으로 참여자들의 신분, 에이전트의 신분, ③에 대한 해쉬값을 전송한다.

(2.2) 호스트 S_2에서 이동 에이전트 실행

① $T_{S_2}(0) = \{c_{S_2}, s_{S_2}, t_{S_2}\}, M_{H_2}(\{c_{S_2}, s_{S_2}, t_{S_2}\})$

② $T_{S_2}(j) = \{c_j, s_j, t_j\}, M_{H_2}(\{c_j, s_j, t_j\}, T_{S_2}(j-1))$

③ $T_{S_2} = \{T_{S_2}(j) \mid 0 \le j \le m\}$

모든 에이전트 코드를 실행하고 난 후 S_2는 T_{S_2}를 저장한다.

⋮

(n.1) 에이전트가 $n-1$번째 호스트 S_{n-1}로부터 n번째 호스트 S_n으로 이동

(n.2) 호스트 S_n에서 이동 에이전트 실행

(n.3) 에이전트가 n 번째 호스트 S_n으로부터 에이전트 소유자 S_0로 이동

① $S_n \rightarrow S_0$: $S_{S_n}(ID_{S_n}, ID_{S_0}, AT), E_{K_S}(p, s_{S_n})$

S_n은 에이전트 암호화 전송을 위해 임의의 키 K_{S_n}를 사용한다.

② $S_0 \rightarrow S_n$: $S_{S_0}(ID_{S_0}, ID_{S_n}, ID_{Agent}, H(*))$

S_0는 에이전트 토큰을 확인하고 응답으로 참여자, 에이전트 신분, ①에 대한 해쉬값을 서명하여 전송한다.

③ $S_n \rightarrow S_0$: $S_{S_n}(ID_{S_n}, ID_{S_0}, ID_{Agent}, P_{S_0}(H(T_{S_n}), K_{S_n}), H(*))$

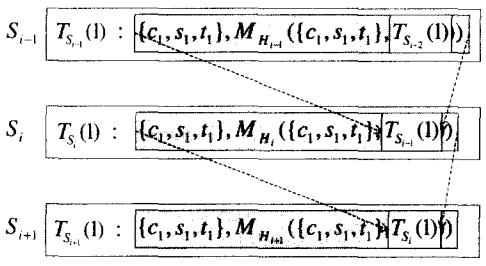
S_n에서 에이전트가 동작한 후의 T_{S_n}에 대한 해쉬값과 K_{S_n}을 공개키로 암호화하고, ②에 대한 해쉬값을 서명하여 보낸다.

3.2 제안 방안의 분석

제안 방안은 실행 추적을 위한 G.Vigna[9]의 방안을 확장하여 사용하고 있는데, 실행 결과 보호 및 실행 추적의 기능을 모두 제공한다. 여기서, 이동 에이전트는 에이전트 소유자 S_0를 출발하여 원하는 정보를 얻기 위해 호스트 $S_i(1 \le i \le n)$ 사이를 여행한다.

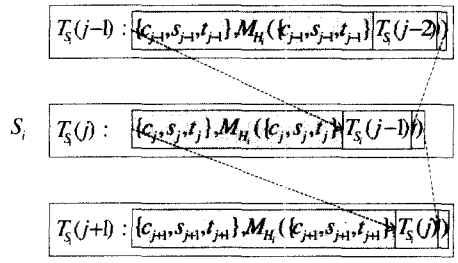
(1) $m=1$인 경우

먼저 에이전트 코드 $p=\{c_j, s_j, t_j\}(1 \le j \le m)$에서 수행할 단계 $m=1$이라면, 각 S_i에서 이동 에이전트를 수행한 결과와 추적값은 $T_{S_i}=\{T_{S_i}(j) \mid 0 \le j \le 1\}$이 된다. 여기서 $T_{S_i}(0)$는 이전 호스트 S_{i-1}에서 수행한 $T_{S_{i-1}}(1)$이 되며, $T_{S_i}(1)$은 현재 호스트 S_i에서 수행한 결과이고 동시에 다음 호스트 $T_{S_{i+1}}(0)$에서의 초기값이 된다. 즉, 각 호스트에서의 초기값과 최종값 사이에는 다음 〈그림 1〉과 같은 사슬 관계를 이루게 된다.



(2) $m>1$인 경우

각 T_{S_i}에서 이전 호스트의 최종값은 다음 호스트의 초기값이 되므로 위 (1)의 경우와 마찬가지로 사슬 관계를 형성한다. 또한, 각 호스트 S_i에서 수행되는 $T_{S_i}(j)(1 \le j \le m)$들도 역시 다음 〈그림 2〉와 같은 사슬 관계를 형성한다.



〔그림 2〕 호스트 S_i에서의 사슬 관계

3.3 수정 방안

제안 방안은 그 목적에 따라 다음과 같이 수정하여 사용할 수 있다.

첫째, 실행에 대한 추적보다는 각 호스트에서 코드를 실행한 후 그 결과만을 안전하게 유지해야하는 경우(예를 들어, 가격 비교를 위해 각 호스트에서 자신의 가격을 제안하는 경우)를 가정한다. 각 호스트에서 수행할 코드는 $m=1$로 두고 실행할 코드 $p=\{c_j, s_j, t_j\}(1 \le j \le m)$를 수정하여 $p=\{c_j, s_j, r_j, t_j\}$ $(1 \le j \le m)$로 둘 수 있다(여기서, r은 호스트에서 수행 결과). 이렇게 수정하고 각 호스트에서 동작을 수행하면 각 호스트끼리 수행 결과에 대한 사슬 관계가 유지된다. 따라서, 수행 결과에 무결성을 유지할 수 있으며, 이후에 임의의 호스트가 자신이 수행했던 결과를 다시 수정할 수도 없다.

둘째, 에이전트 수행의 각 단계를 좀 더 안전하게 수행하기 위해서는 다음과 같이 수정이 가능하다. 각 호스트는 에이전트를 계속 실행하면서 각 상태 $\{c_j, s_j, t_j\}$와 H_i을 사용하여 메시지 인증값을 계속해서 저장하면서, m번 반복한다. 여기서, H_i는

$$H_i = \begin{cases} H(K_{S_0}) & \text{if } i=1 \\ H(T_{S_{i-1}}) & \text{if } 1<i \le n \end{cases}$$

와 같이 정의되는데, 호스트 S_i는 이전 호스트에서 작성된 $T_{S_{i-1}}$를 해쉬하여 메시지 인증값의 비밀키로

사용한다. 이를 수정하여 한 호스트에서 각 수행에 대한 안전성을 더 강화하기 위해서 기존의 $M_{H_i}(\{c_j, s_j, t_j\}, T_{S_i}(j-1)), 1 \le j \le m$ 부분을 $M_{H_i(T_{S_i}(j-1))}(\{c_j, s_j, t_j\}, T_{S_i}(j-1))$로 수정하면, 각 실행에 대한 인증값의 키들이 사슬 관계를 생성한다. 따라서, 전체 실행들이 이전 실행의 추적값에 의존하여 한 호스트가 단일 키로 인증값을 생성하는 것을 막을 수 있다.

셋째, 제안 방안은 각 호스트에서 실행한 에이전트 코드에 대한 시뮬레이션을 수행하여 부정한 호스트를 검출할 수 있다. 그러나, 부정한 호스트를 검출하는 것과 더불어 어떠한 실행을 수행할 때 부정이 개입되었는지 추적하기 위해서는 제안 방안의 수정이 필수적이다. 각 호스트들은 모든 에이전트 코드를 실행하고 난 후 $T_{S_i} = \{T_{S_i}(j) \mid 0 \le j \le m\}$을 저장하고 그 해쉬값만 다음 호스트로 전달한다. 이를 수정하여 T_{S_i}를 다음 호스트에 에이전트 코드와 함께 전달한다면 이전 호스트에 대한 실행을 확인해 볼 수 있고, 최종적으로 에이전트 소유자가 추적해 볼 수 있는 장점이 있다. 그러나 전달해야 하는 메시지가 에이전트 이동하는 동안 계속해서 늘어나므로 필수적인 응용에 대해서만 사용하도록 한다.

Ⅳ. 부정 검출 및 안전성 분석

4.1 부정 검출

소유자 S_0가 이동 에이전트가 여행을 끝내고 다시 돌아온 후 부정이 있다고 생각되면 검증 절차를 시작한다. S_0는 (1.1), (2.1), …, (n.1)의 ②에서 각 호스트에서 서명된 *ack*를 보고 프로토콜 참여 여부를 알 수 있으며 서명 기법이 안전하다면 각 호스트는 그 사실을 부인할 수 없다. 여기서, 이동 에이전트 코드 $p = \{c_j, s_j, t_j\}(1 \le j \le m)$에서 $m = 1$인 경우는 해쉬 체인을 이용한 이동 에이전트 수행 결과 보호와 유사하게 동작함으로써 원하는 정보를 해당 호스트에서 얻은 후 에이전트가 소유자에게 전달할 수 있다. $p = \{c_j, s_j, t_j\}(1 \le j \le m)$의 일반적인 경우에는 임의의 호스트가 부정한 동작을 수행했는지를 정직한 서버의 도움을 통하여 확인할 수 있다. 즉, S_x가 부정 서버라면 정직한 서버 S_{x-1}의 에이전트 수행 후 최종값 $T_{S_{x-1}}$를 얻을 수 있고, 이를 기반으로 초기 상태 $s_{S_{x-1}}$에서 에이전트 코드 p를 수행하여 시뮬레이션한 T_{S_x}'와 S_x에서 수행 후 결과인 T_{S_x}를 서로 비교하여 같지 않으면 S_x가 부정 서버라는 것을 검출할 수 있게 된다. 만약 S_0이 이동 에이전트가 여행한 모든 부정을 검출하기 원한다면 S_1에서 S_n으로부터 저장된 $T_{S_i}(1 \le i \le n)$를 제공받아 전체를 시뮬레이션하면 하나 이상의 부정에 대해서도 검출이 가능하다.

4.2 안전성 분석

제안 방안은 다음과 같은 안전성을 가진다.

첫째, 프로토콜 참여에 대한 부인방지를 제공한다. 이는 각 호스트가 서명된 *ack*에 의해 프로토콜 참여가 결정이 되며, 부정의 응답이라면 프로토콜은 즉시 중단된다. 즉, 서명 방안이 안전하다면 차후에 프로토콜 참여에 대한 부인 방지를 제공하고, 동시에 암호화된 에이전트 코드만 전송되므로 암호 시스템이 안전하다면 암호화 키 없이는 에이전트에 대한 어떠한 정보도 얻을 수 없다.

둘째, 각 수행에 대한 전방향 무결성(Forward Integrity)을 제공한다. 각 호스트는 에이전트 코드 실행에 대한 책임을 지고 이전 호스트의 추적값을 해쉬한 후 이를 키로 적용하여 에이전트 코드 수행에 대한 메시지 인증값을 생성한다. 호스트에서의 각 실행 및 연속하는 호스트 사이에는 사슬 관계를 형성함으로써 한 번 수행 후에는 자신이 작성한 값이라 할지라도 수정할 수 없도록 하는 특징을 가진다.

셋째, 각 호스트는 이전 호스트 수행이 올바른지 확인할 수 있는 방법을 제공한다. 임의의 호스트에서 이전 호스트에서 수행된 결과에 대해 의심이 간다면 이전 호스트에서의 코드 초기값을 전송받아 직접 시뮬레이션함으로써 이전 호스트에서의 에이전트 실행에 대한 검증이 가능하다.

넷째, 코드 삽입 또는 삭제 공격을 막을 수 있다. 앞에서 살펴본 바와 같이 모든 이동 에이전트 실행은 각 실행 간 또는 각 호스트 간 메시지 인증값으로써 사슬 관계를 맺고 있으므로 임의의 코드를 삽입 또는 삭제하여 수행하더라도 이에 대한 검출이 가능하다.

부가적으로 추적값 t_j에 대하여 검토하여 보면, 실제 에이전트 코드에서 실행되는 문장은 "White Statement"와 "Black Statement"로 나눌 수 있는데[9], t_j는 각 문장에 따라 다른 처리를 수행해야 한다. White Statement는 에이전트 내부 변수의

값을 기반으로 에이전트의 실행 상태가 수정되는 연산이고, Black Statement는 외부 실행 환경으로부터 받은 정보를 사용하여 에이전트의 상태가 변하는 연산이다. $z=x+y$와 같이 내부 변수의 연산으로 수행되는 White Statement는 수행시 그 때의 스택에 대한 해쉬값 또는 서명값으로 t_i를 정의할 수 있으며, $read(x)$와 같은 Black Statement는 입력값 x에 대한 실행 호스트의 서명으로 t_i를 정의할 수 있다.

4.3 기존 방안과의 비교

다음 〈표 1〉은 이동 에이전트 보호 중 실행 추적과 수행 결과 보호를 위한 기존 방안들과 제안 방안을 비교하였다.

〔표 1〕 이동 에이전트 보호 방안들의 비교

		특징	안전성	효율성
Vigna〔9〕의 방안		암호 프로토콜을 이용한 부정 검출 실행 단계별 추적 불가	전자서명 및 해쉬 함수의 안전성에 기반 무결성, 부인 방지 제공	호스트 수 및 실행 단계에 비례하여 효율성 감소 추적 정보 저장을 위한 별도의 공간 필요
Karjoth 등〔10〕의 방안		해쉬 체인을 이용한 수행 결과 보호	해쉬 함수의 안전성에 기반 비밀성, 무결성 제공	호스트 수에 비례하여 효율성 감소
제안 방안	실행 추적 측면	해쉬 체인을 이용한 부정 검출 및 수행 결과 보호 실행 단계별 추적 가능	전자서명 및 해쉬 함수의 안전성에 기반 무결성 제공	실행 단계에 비례하여 효율성 감소 추적 정보 저장을 위한 별도의 공간 필요
	수행 결과 보호 측면	수행 결과 보호만을 위한 방안으로 수정	해쉬 함수의 안전성에 기반 비밀성, 무결성, 부인 방지 제공	호스트 수에 비례하여 효율성 감소

V. 결 론

본 논문에서는 이동 에이전트의 이동성으로 인해 발생하는 시큐리티 취약성에 대하여 설명하고 이를 해결하기 위한 기존 연구로써 보안 방지를 목적으로 하는 Blackbox Security와 Computing with Encrypted Functions, 보안 검출을 목적으로 하는 Cryptographic Traces, 해쉬 체인을 이용한 수행 결과 보호에 대하여 살펴본 후, 이동 에이전트 실행 결과 보호 및 실행 추적을 동시에 수행할 수 있는 새로운 방안을 제안하였다. 본 제안 방안은 부인 방지, Forward Integrity, 삽입 또는 삭제 공격 방지 특성을 지닌다. 다수의 부정한 호스트에 대해서도 각 호스트에 대한 이동 에이전트 실행시 부정 검출이 가능하도록 되어 있으며, 각 호스트에서 이전 호스트 실행에 대한 부정 검출도 역시 가능하도록 설계되었다.

이동 에이전트 시스템은 다양한 사용자의 요구에 부합하여 많은 컴퓨팅 분야에서 적용되고 있으며, 이를 위한 상업적인 제품들도 등장하고 있는 실정이다. 최근, 실생활의 여러 서비스가 네트워크 상으로 이전하고 전자상거래를 기반으로 하는 다양한 응용 분야가 등장하고 있다. 이러한 상황에서 새로운 사이버 공격 기법의 등장, 조직화된 시스템 해킹 등으로 인해 액세스 제어 기반의 호스트 보호뿐만 아니라 사용자의 권한을 대행하는 이동 에이전트 자체의 보호를 위한 패러다임의 전환이 요구되고 있다. 따라서, 기존의 악의적인 에이전트에 대한 호스트 보호뿐만 아니라 악의적인 호스트 및 에이전트에 대한 이동 에이전트의 보호도 필수적으로 연구되어야 하는 분야이다.

참 고 문 헌

〔1〕 http://java.sun.com

〔2〕 http://www.microsoft.com

〔3〕 J.E.White, "Telescript Technology: The Foundation for the Electronic Marketplace," General Magic, Inc., Sunnyvale, CA, 1994

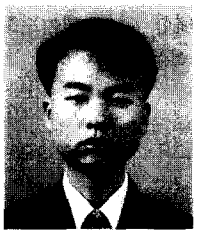
〔4〕 R.S.Gray, "Agent Tcl: A Transportable Agent System," Proceedings of the CIKM Workshop on Intelligent Information Agents, Baltimore, MA, 1995

〔5〕 F.Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts," In: G.Vigna (Ed.), Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science 1419, pp. 92-113, 1998

〔6〕 F.Hohl, and K.Rothermel, "A Protocol

Preventing Blackbox Tests of Mobile Agents," In: R.Steinmetz (Ed.), Kommunikation in verteilten Systemen, Springer-Verlag, pp. 170-181., 1999
[7] T.Sander and C.Tschudin, "Towards Mobile Cryptography," International Computer Security Institute (ICSI), TR-97-049, 1997
[8] T.Sander and C.Tschudin, "Protecting Mobile Agents Against Malicious Hosts," In: G.Vigna (Ed.), Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science 1419, pp. 44-60, 1998
[9] G.Vigna, "Cryptographic Traces for Mobile Agents," In: G.Vigna (Ed.), Mobile Agents and Security, Springer-Verlag, Lecture Notes in Computer Science 1419, pp. 137-153, 1998
[10] G.Karjoth, N.Asokan, C.Gulcu, "Protecting the Computation Results of Free-roaming Agents," Proceedings of the Second International Workshop, MA'98, pp. 195-207, 1998
[11] S.Oaks, Java Security, O'Reilly & Associates, Sebastopol, CA, 1998
[12] 박종렬, 신욱, 이동익, "이동코드와 보안문제", 정보처리학회지 vol.7 no.2, pp. 115-122, 2000
[13] A.Menezes, P.van Oorschot, and S.Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL, 1996

〈著者紹介〉



신 원 (Shin, Weon) 준회원
1996년 2월 : 부경대학교 전자계산학과 졸업
1998년 3월 : 부경대학교 전자계산학과 석사
1998년~현재 : 부경대학교 전자계산학과 박사과정
〈관심분야〉 정보보호, 이동에이전트, 멀티미디어 통신, 암호이론, 네트워크 보안

이 경 현 (Rhee, Kyung Hyune) 정회원
1982년 2월 : 경북대학교 사범대학 수학교육과 졸업
1985년 2월 : 한국과학기술원 응용수학과 석사
1992년 2월 : 한국과학기술원 수학과 박사
1985년 2월~1993년 2월 : 한국 전자 통신 연구소 연구원, 선임 연구원
1993년 3월~현재 : 부경대학교 전임강사, 조교수, 부교수
〈관심분야〉 정보보호, 암호학, 멀티미디어 정보보호, 네트워크성능 평가, 재시도 대기체계론