이동에이전트의 클론 실행을 방지하는 프로토콜 설계

백주성* , 이동익, R.Ramakrishna
광주과학기술원 정보통신공학과
{bjstar,dilee,rama}@kjist.ac.kr

Design of a protocol for preventing mobile agent clone from execution

Jusung Baek, Dongik Lee and R.Ramakrishna
Info.&Comm. Dept., KwangJu Institute of Science & Technology

요 약

이동에이전트는 독립된 객체로서 자율성을 가지고 컴퓨터들을 이동하며 부여된 임무를 수행하는 프로그램이다. 이동에이전트는 코드와 데이터로 구성된 프로그램이므로 쉽게 복제될 수 있다 이렇게 복제된 이동에이전트를 이동에이전트 클론이라 한다. 복제된 클론은 원본과 구별이 불가능하다 이것은 에이전트의 인증을 불가능하게 만들고 예상되지 않은 에이전트의 중복 수행을 야기하며 에이전트의 내부정보 유출 공격을 위한 수단으로 사용된다 본 논문에서는 이동에이전트 클론에 의한 이러한 문제점을 고찰하고 온라인 상에서 클론의 존재를 탐지하고 실행을 방지하며 클론을 생성한 서버를 확인하는 프로토콜을 설계한다.

1. 서론

이동에이전트는 독립된 객체로서 자율성을 가지고 컴퓨터들을 이동하며 부여된 임무를 수행하는 프로그램이다[1]. 이동에이전트 개념은 이동코드 개념과 함께 분산 컴퓨팅의 새로운 패러다임으로 주목받고 있으며 워크플로우, 전자상거래, 정보검색 등의 영역에 사용되고 있다. 그러나 이동에이전트의 보안 취약성은 이동에이전트가 응용될 수 있는 영역을 제한하고 있다.

이동에이전트의 안전한 실행을 보장하기 위해서는 네트웍에서의 이동에이전트의 보호와 이동에이전트를 실행하는 서버로부터의 보호가 필수적이다. 네트웍에서 이동에이전트의 이동은 서버간 데이터의 전송과 동일하므로 기존 분산시스템에서 사용된 보안 과정을 적용하여 이동에이전트를 네트웍에서의 불법적인 도청, 복제, 변경에 대해 보호할 수 있다 서버로부터 에이전트를 보호하는 것은 매우 어려운 일로 알려지고 있으며[2], 최근에는 에이전트의 실행 과정을 추적하여 공격을 가한 서버를 확인함으로서 에이전트를 보호하는 연구[1,3]와 에이전트의 코드와 데이터가 에이전트의 실행에 어떤 영향을 미치는지 서버가 알 수 없도록 함으로서 의미 있는 에이전트 행동 변경 공격을 보호하는 연구[4]가 진행되고 있다. 그러나 에이전트를 보호하기 위해 반드시 해결되어야 하는 또 다른 문제 - 이동에이전트 클론의 문제는 아직 명확하게 연구된 바가 없다. 단지 [2]에서 클론을 구별하는 것은 불가능하다고 언급했을 뿐이다

이동에이전트는 코드와 데이터로 구성된 프로그램이므로 이동에이전트를 전송 받은 서버는 손쉽게 같은 이동에이전트를 복제할 수 있다. 이렇게 복제된 이동에이전트를 이동에이전트 클론이라 한다. 클론은 일단 생성되면 원본과 구별이 불가능하다 본 논문에서는 이동에이전트의 클론에 의해 야기되는 문제점들을 고찰하고 온라인 상에서 클론의 실행을 방지하며 클론을 생성한 서버를 확인하는 프로토콜을 설계한다.

본 논문은 다음과 같이 구성된다 2장에서는 이동에이전트 클론에 의해 야기되는 문제점들을 고찰한다. 3장에서는 클론 탐지 모델과 프로토콜을 설계하며 4장에서 결론과 향후 연구 방향을 제시한다

2. 이동에이전트 클론의 문제점

원본 에이전트와 클론 에이전트를 구분하기 위해서는 원본 에이전트에서 클론 에이전트로 복제할 때 원본 에이전트가 가지는 특정 정보를 클론이 가질 수 없도록 해야한다 그러나 원본 에이전트가 가지는 특정 정보 역시 에이전트를 복사할 때 클론에 제공되므로 이들 에이전트를 구분할 수 있는 방법은 없다[2]. 더구나 클론 에이전트는 실행될 때 자신이 원본 에이전트임을 주장하게 되며 에이전트를 생성한 에이전트 서버도 이를 구분할 수 없다. 적의를 가진 서버에 의한 이동에이전트의 클론 실행은 다음과 같은 문제를 발생시킨다.

◦ 에이전트의 인증 문제

어떤 객체가 특정 에이전트와 통신하고자 할 때 이 에이전트가

통신하고자 하는 에이전트인지 클론인지 확인할 수 없다.

◦ **에이전트의 중복실행**

클론과 원본에이전트의 동시실행으로 인해 두 개 이상의 예상되지 않은 실행결과가 발생한다. 만일 에이전트가 어떤 트랜잭션을 수행한다면 복수의 트랜잭션이 수행되는 것이다

◦ **클론의 병렬 수행을 통한 에이전트의 정보 노출**

이동에이전트가 서버로부터 자신의 코드와 데이터를 은폐할 수 있다고 가정해도 서버는 에이전트의 클론을 생성해서 각 클론에게 다른 입력 값을 주어 클론으로부터 나오는 결과로 에이전트의 행동이나 지식 등의 정보를 추출할 수 있다

이미 개발된 이동에이전트 시스템중의 하나인 Mole 시스템[5]에서 간단한 이동에이전트 application을 작성하고 클론의 생성에 의한 실행 결과를 확인해보았다. 클론 생성은 Mole 시스템의 에이전트 전송 프로토콜을 이용해 Mole 시스템의 수정없이 쉽게 만들 수 있었다. 클론의 생성으로 인해 원본 에이전트 확인이 불가능했으며 에이전트가 중복되어 실행된 것을 확인할 수 있었다.

3. 클론 탐지 프로토콜 설계

3.1 클론 탐지 프로토콜

그림 1. 클론 탐지 모델은 에이전트의 실행과정과 Coordinator의 관계에 대한 Petri-Net[6] 모델이다. 각 서버에 있는 플레이스들은 에이전트가 다음 트랜지션을 수행하기 위한 상태에 있음을 의미한다. 에이전트의 상태가 각 플레이스로 진이하면 서버는 *Coordinator*에게 다음 트랜지션에 해당하는 토큰을 요청한다. 트랜지션들은 에이전트의 상태가 트랜지션의 바로 이전 플레이스일 때 Coordinator로부터 토큰을 받으면 실행된다. 토큰들은 모든 트랜지션마다 다르며 트랜지션을 실행하고자 하는 서버의 요청에 따라 만들어진다 토큰은 오직 Coordinator만이 생성할 수 있도록 Coordinator에 의해 디지털 서명된다.

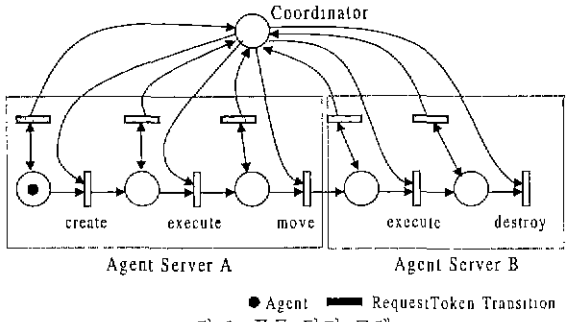

그림 1. 클론 탐지 모델

에이전트의 실행과정에 따른 에이전트 생성서버(C), 에이전트 실행서버(E), Coordinator(O)는 다음의 프로토콜을 수행한다. 프로토콜을 단순화하기 위해 모든 통신망은 안전하다고 가정한다. 즉 모든 메시지는 통신망 사이에서 변경되지 않는다.

C는 새로운 에이전트 ID를 만들고 RequestCreateToken 메시지를 O에게 보낸다.

◦ **RequestCreateToken (ID(Agent),ID(C))**

RequestCreateToken 메시지를 받은 O는 C에게 ID(Agent)에 해당하는 CreateToken을 이미 제공했는지 확인한다 이미 토큰이 제공되었다면 O는 C를 클론 생성자로 등록하고 토큰을 제공하지 않는다. 그렇지 않으면 CreateToken을 생성하고 토큰 테이블에 저장한 후 C에게 제공한다. Sig(A,M) 은 M이 A에 의해 서명된 것을 의미한다

◦ **CreateToken (ID(Agent),ID(C),Sig(O,ID(Agent),ID(C)))**

CreateToken을 제공받은 C는 에이전트를 생성하고 RequestExecuteToken 메시지를 O에게 보낸다.

◦ **RequestExecuteToken (ID(Agent),ID(C),CreateToken)**

RequestExecuteToken을 받은 O는 메시지에 포함되어 있는 CreateToken과 자신이 마지막으로 제공한 토큰이 같으면 ExecuteToken을 생성하고 토큰테이블에 저장한 후 C에게 제공한다 T는 고유한 값으로서 C에 전달되는 ExecuteToken 과 MoveToken들을 구별하기 위한 값이다

◦ **ExecuteToken (ID(Agent),ID(C),T,Sig(O,ID(Agent),ID(C),T))**

ExecuteToken을 받은 C는 에이전트를 실행시켜 에이전트로부터 이동이나 소멸요청을 받는다. 이동요청일 경우 RequestMoveToken 메시지를 O에게 보낸다. E는 에이전트가 이동할 서버이다.

◦ **RequestMoveToken (ID(Agent),ID(C),ID(E),ExecuteToken)**

RequestMoveToken을 받은 O는 ExecuteToken이 자신이 마지막으로 보낸 토큰임을 확인하면 MoveToken을 생성하고 저장한 후 C에 제공한다

◦ **MoveToken (ID(Agent),ID(C),ID(E),T,**
Sig(O,ID(Agent),ID(C),ID(E),T))

C는 E에게 에이전트와 MoveToken을 보낸다.

◦ **MoveAgent (Agent,ID(C),MoveToken)**

E는 에이전트를 받으면 RequestExecuteToken 메시지를 O에게 보낸다.

◦ **RequestExecuteToken (ID(Agent),ID(E),MoveToken)**

O는 ID(E)가 MoveToken에 포함되어 있는 ID(E)와 같고 MoveToken이 자신이 마지막으로 전송한 토큰임을 확인하면 ExecuteToken을 생성하여 E에게 제공한다

◦ **ExecuteToken (ID(Agent),ID(E),T,Sig(O,ID(Agent),ID(E),T))**

E는 ExecuteToken을 받고 C가 수행한 과정을 그대로 수행한다 에이전트를 실행시켜 에이전트로부터 소멸요청을 받으면 RequestDestroyToken 메시지를 O에게 보낸다

◦ **RequestDestroyToken (ID(Agent),ID(E),ExecuteToken)**

O는 이 메시지를 받아 ExecuteToken이 마지막으로 전송된 토큰인지 확인되면 DestroyToken을 생성하여 E에게 제공한다.

◦ DestroyToken (ID(Agent),ID(E),Sig(O,ID(Agent),ID(E)))

E는 이 토큰을 받아 에이전트를 소멸시킨다. O에 저장된 에이전트의 토큰들은 일정시간 후 소멸된다.

3.2 클론 생성 모델

그림 2,3,4의 클론 생성 모델은 에이전트 서버가 클론을 생성하는 과정을 Petri-Net으로 모델링한 것으로서 클론 탐지 모델과 같이 클론 탐지 프로토콜을 검증하는데 사용된다. 클론의 생성은 아래의 그림들과 같이 세 가지 경우로 표현된다.

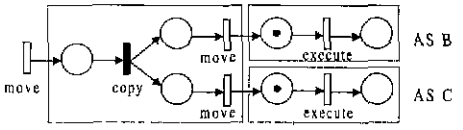

그림 2 에이전트를 받아 클론을 생성한 후 실행시키지 않고 다른 서버로 이동시키는 경우

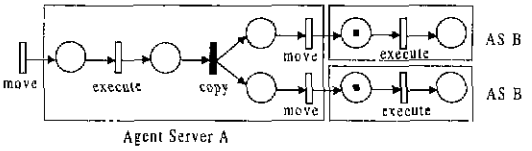

그림 3. 에이전트를 받아 실행시킨 후 클론을 생성시켜 다른 서버로 이동시키는 경우

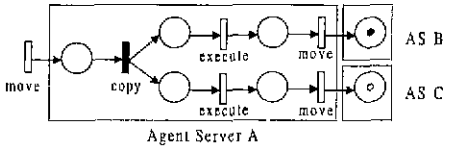

그림 4. 에이전트를 받아 클론을 생성시켜 각각 실행시킨 후 각각 서버로 이동시키는 경우

3.3 클론 탐지

클론 탐지 프로토콜은 클론 탐지 모델에 근거해 클론 생성 모델에 의해 발생된 클론들을 탐지하여 에이전트의 실행 보증 토큰을 제공하지 않음으로서 클론의 정상적인 실행을 방지하며 클론을 발생시킨 에이전트 서버를 찾아낸다.

첫 번째 클론생성모델에서 서버 A는 에이전트를 복제한 후 이동시키기 위한 토큰을 Coordinator로부터 얻을 수 없다 A가 RequestMoveToken을 Coordinator에게 보내게 되면 Coordinator는 최근 마지막으로 제공한 토큰이 MoveToken이므로 토큰의 제공을 거부하고 A를 클론 생성자로 확인하기 때문이다. 또한 A가 이전 서버로부터 에이전트를 전송 받을 때 얻은 MoveToken을 사용해서 서버 B와 C에 에이전트를 전송하면 B와 C가 Coordinator에게 ExecuteToken을 요구할 때 Coordinator가 잘못된 MoveToken임을 확인하게 되고 A가 클론 생성자임을 확인한다.

두 번째 클론생성모델에서는 서버 A는 B에게 에이전트를 이동시키기 위한 토큰을 하나밖에 받을 수 없다 따라서 서버 B에게 전송되는 에이전트들중 먼저 도착하는 에이전트만 제대로 실행된다. 서버 B는 A가 같은 에이전트를 두 번 보냈다는 사실을 확인할 수 있으므로 Coordinator에 의해 두 번째 에이전트에 대한 ExecuteToken이 거절당했을 때 A가 클론생성자임을 확인시킬 수 있다.

세 번째 클론생성모델은 서버 A에서 에이전트가 두 번 실행된 후 각자 다른 서버로 이동을 요청하는 경우다. 이 경우 서버 A는 두 개의 MoveToken을 얻을 수 없으므로 둘 중 하나는 잘못된 MoveToken을 가지고 이동하게 된다. 따라서 서버 B나 C에서 A가 클론 생성자임이 확인된다. 또한 서버 A에서 일어난 두 번의 실행은 실행 토큰이 같기 때문에 서버 A는 에이전트에 의해 두 번의 트랜잭션이 일어났다고 주장할 수 없다.

4. 결론 및 향후 연구

이동에이전트의 보안 연구에서 지금까지 거의 다루어지지 않은 클론의 문제점을 조사하고 기존 이동에이전트 시스템에서 클론이 쉽게 만들어 질 수 있다는 사실을 확인하였다. 클론의 문제점을 해결하기 위해 클론 탐지 프로토콜을 설계하였고 클론 생성 모델을 통해 클론 탐지 프로토콜이 성공적으로 수행됨을 증명했다

클론 탐지 프로토콜은 기존 이동에이전트 시스템에 추가적인 요소를 필요로 한다. 따라서 클론의 실행이 문제가 되는 응용부분의 경우에만 이 프로토콜을 적용하는 것이 바람직하다고 여겨진다

클론 탐지 프로토콜의 증명은 클론 생성 모델을 통해 정형화 되지 않은 방식으로 기술되었다. 정형화된 기법을 이용한 프로토콜의 완전성 증명이 앞으로의 연구 과제이다. 설계된 클론 탐지 프로토콜은 X-MAS[7]에 적용되어 구현될 예정이다. 그리고 추후 클론 실행의 제약이 약한 시스템에서 시스템의 성능 저하를 줄이면서 클론 탐지를 수행하는 프로토콜에 대한 연구를 진행하고자 한다.

참고문헌

[1] 백주성,이동익, "디지탈 서명과 감시도구를 이용한 이동에이전트의 보호", 한국정보과학회 '97가을학술발표논문집(III) 제24권 2호 p419-422 1997.10.

[2] William M. Farmer, Joshua D. Guttman. and Vipin Swarup. "Security for Mobile Agents: Issues and Requirements," In Proceedings of the 19th NISSC, pages 591-597, Baltimore, Md., October 1996

[3] Giovani Vigna, "Protecting Mobile Agents through Tracing," in Proceedings of the Third ECOOP Workshop on Operationg System support for Mobile Object Systems, 1997

[4] Fritz Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts," LNCS on "Mobile Agent Security", 1997

[5] Markus Straser, Joachim Baumann, and Fritz Hohl. "Mole -- a Java based mobile agent system." In 2nd ECOOP Workshop on Mobile Object Systems, pages 28-35. Linz, Austria, July 1996.

[6] Wolfgang Reisig, "A Primer in Petri Net Design". Springer-Verlag, 1992.

[7] 유정준,백주성,박종열,이동익, "Java-based Mobile Agent System: X-MAS", KICS'98, May 1998 , K-JIST