단방향 해쉬 함수를 이용한 이동에이전트의 실행 결과의 보호

강도근, 공은배
충남대학교 컴퓨터공학과
{dkkang, keb}@comeng.chungnam.ac.kr

The Protection of Execution Result of Mobile Agent Result Using One-way Hash function

DoKeun Kang, EunBae Kong
Computer Engineering Department, Chungnam National Univ.

요약

이동 에이전트는 자의적인 이동 정보를 유지하며 네트워크를 순회하면서 목적을 성취하는 개체이므로, 이동에이전트의 목적을 성취하기 위해서 이동 에이전트는 자신의 실행을 이동에이전트가 방문하는 호스트에 그 실행을 요청하게 된다. 따라서 이동 에이전트 시스템에서는 보안 문제는 매우 중요한 문제이다. 이동 에이전트 시스템에서 발생할 수 있는 보안 문제는 크게 호스트를 보호하는 문제와 이동 에이전트 자체를 보호하는 문제로 나눌 수 있다. 이동 에이전트에서 발생할 수 있는 문제점 중 가장 중요하고 심각한 것은 이동 에이전트가 여러 호스트를 순회하면서 이동 에이전트의 코드를 실행한 결과를 수집하는데, 악의를 갖은 호스트가 이동 에이전트의 결과를 훔쳐볼 수도 있고, 그 결과를 조작할 수도 있다는 점이다.

본 논문에서는 이동 에이전트가 각 호스트를 순회하면서 실행한 결과를 악의적인 호스트가 이동 에이전트의 결과를 훔쳐볼 수 없게 하고, 조작할 수 없게 하는 방법을 단방향 해쉬 함수를 이용하여 보호하는 방법을 제안 한다.

1. 서론

이동 에이전트는 자의적인 이동 정보를 유지하며 네트워크를 순회하면서 목적을 성취하는 개체이다. 이동 에이전트는 스스로를 실행할 수 없으므로 에이전트 코드의 실행을 이동 에이전트가 방문한 호스트에 실행을 요청하게 된다. 또 이동 에이전트가 목적을 성취하기 위해서 여러 호스트를 순회할 수도 있다. 이동 에이전트가 여러 호스트를 순회할 경우, 이동 에이전트가 방문하는 호스트는 신뢰할 만한 호스트가 될 수도 있고, 그렇지 못할 수도 있다.

이런 이동 에이전트의 대표적인 예는 항공권 좌석 예약 에이전트를 들 수 있다. 사용자는 사용자가 여행하고자 하는 행선지와 출발 시간, 좌석 수와 가격을 명시한 이동 에이전트를 여러 비행기 회사의 호스트로 전송한다. 전송된 이동 에이전트는 사용자가 명시한 목적에 가장 적합한 항공권을 찾아 예약을 하고, 그 결과를 사용자에게 보고한다.

이러한 이동 에이전트는 Home Place(이하 HP)안에서 생성된다. HP는 사용자의 실행 의도나 목표를 입력으로 받아들여 이동 에이전트를 생성하고, 이동 에이전트를 공개된 네트워크로 전송하는 역할을 한다. HP 에서 생성된 이동 에이전트는 자의적인 이동 정보 혹은 HP 내에서 생성된 방문 리스트에 의해서 실행될 첫번째 호스트로 이동하게 된다. 이동한 호스트에서 이동 에이전트는 자신의 실행 목적을 호스트에게 제시한 후 실행을 요청하게 된다. 이동 에이전트를 받아들인 호스트는 이동 에이전트를 실행할 지 결정하고 실행을 시작한다. 이동 에이전트의 실행을 끝낸 호스트는 이동 에이전트에게 이동 에이전트의 실행 결과를 적재한 후, 이동 에이전트가 이동할 다음 호스트를 선정하거나 방문리스트에 따라서 다음 호스트로 전송한다.

이동 에이전트 시스템에서 보안 문제는 매우 중요한 문제이며, 크게 이동 에이전트를 실행하는 호스트를 보호하는 문제와 이동 에이전트 자체를 보호하는 문제로 나눌 수 있다. 현재까지 이동 에이전트를 보호하려는 연구가 진행되어 왔으나 아직 초기단계에 머무르고 있다.[2,5,6,7,8,9] 이동 에이전트에서 발생할 수 있는 문제점 중 가장 중요하고 심각한 것은 이동 에이전트가 여러 호스트를 순회하면서 이동 에이전트의 코드를 실행한 결과를 수집하는데, 악의를 갖은 호스트가 이동 에이전트의 결과를 훔쳐볼 수도 있고, 그 결과를 조작할 수도 있다는 점이다.

본 논문에서는 우선 2 장에서 이동 에이전트에서 발생할 수 있는 보안 문제를 나열하고 현재까지 이를 보호하기 위해 고안되었던 방법을 간략히 기술하고, 3 장에서는 이동 에이전트의 결과를 보호하기 위하여 암호학적인 방법을 기술하고 4 장에서는 3 장에서 제시한 방법을 단방향 해쉬함수를 사용하여 보완하여 현실적으로 적용할 수 있는 방법을 기술하며, 마지막으로 5 장에서는 결론을 내기로 한다.

2. 이동에이전트의 보안문제

이동 에이전트는 자의적인 이동 정보를 유지하며 네트워크를 순회하면서 목적을 성취하는 개체이다. 이동 에이전트는 스스로를 실행할 수 없으므로 에이전트 코드의 실행을 이동 에이전트가 방문한 호스트에 실행을 요청하게 된다. 이동 에이전트가 여러 호스트를 순회할 경우, 이동 에이전트가 방문하는 호스트는 신뢰할 만한 호스트가 될 수도 있고, 그렇지 못할 수도 있다.

이동 에이전트에서 발생하는 보안 문제는 크게 이동 에이전트를 실행하는 호스트를 보호하는 문제와 이동 에이전트 자체

를 보호로 나눌 수 있다.

이동 에이전트를 실행하는 호스트는 이동 에이전트를 네트워크로부터 받아들여, 이동 에이전트가 요청한 서비스를 수행하고, 결과를 반환해야 한다. 하지만 호스트는 네트워크로부터 받아들인 이동에이전트의 특성과 행동을 수행하기 전까지는 알아 볼 수가 없다. 따라서 이동 에이전트가 Virus 나 Worm 과 같은 행동을 하는 경우에 호스트 자신을 보호해야 하고, 또 Denial of Service 와 같이 호스트의 컴퓨팅 파워를 무력화 시킬 수 있는 이동 에이전트로부터 보호해야 한다. 또한 이동 에이전트가 호스트의 리소스와 데이터를 고갈시키거나 마음대로 접근하는 것으로부터 보호해야 한다. 호스트를 보호하기 위해서는 이동 에이전트를 호스트상의 독립된 메모리영역으로 분리해서 각 이동 에이전트가 접근할 수 있는 메모리 영역을 제한하는 Sandbox[4] 모델을 사용할 수 있고, 호스트의 자원에 대한 명세나 정책을 적용함으로써 보완할 수 있다. 또한 호스트가 이러한 특징을 가지는 인터프리터를 사용한다면, 호스트를 이동 에이전트로부터 보호할 수 있고, 이러한 인터프리터로서는 Java[7]나 Secure TCL 등이 그 좋은 예이다.

이동 에이전트가 방문한 호스트가 악의적인 호스트일 경우, 호스트는 이동 에이전트의 코드를 마음대로 바꾸어 수행하여, 결과를 조작할 수 있고, 이동 에이전트가 수집한 데이터를 마음대로 조작할 수 있다. 따라서 이동 에이전트의 코드와 수집한 데이터나 수행 결과를 악의적인 호스트가 마음대로 변경시키지 못하게 해야 하며, 이동 에이전트가 수집한 데이터를 노출 시키지 말아야 한다. 호스트는 이동 에이전트의 모든 코드와 데이터에 접근할 수 있고, 조작할 수 있기 때문에 이러한 이동 에이전트를 보호하는 것은 매우 어려운 문제이다.[2] 이러한 이동 에이전트를 보호할 목적으로 제한한 모델들은 안전한 AgentTcl 언어(Safe-Tcl language), UC 버클리의 Towards Mobile Cryptography, Mole 시스템[6]에서 제안한 Time Limited Blackbox Security[3,5], Ara 시스템이 제안한 Passport model 등이 있다.

이동 에이전트가 수집한 결과값은 악의적인 호스트로부터 반드시 보호하여야 한다. 이동 에이전트의 결과를 보호하지 못하면, 이동 에이전트가 적재하여 가지고 다니는 결과에 악의적인 호스트는 다음의 두 가지의 해를 끼칠 수 있다.

- 호스트가 이동 에이전트의 결과를 엿보는 경우 : 호스트는 이동 에이전트가 수집한 결과를 엿볼 수 있으므로,이전 호스트의 결과에 비슷하지만 약간 차이가 나는 결과를 제시할 수 있다.
- 호스트가 이동 에이전트의 결과를 조작하는 경우 : 다른 호스트에서 수집한 결과 값을 삭제할 수 있고, 그 결과를 다른 결과값으로 조작할 수 있다.

따라서 방문하게 될 호스트에 대해 이동 에이전트가 수집한 결과를 보호하는 것은 아주 중요하다. 이동 에이전트가 수집한 데이터를 보호함으로써, 이동 에이전트가 수집한 데이터를 보다 신뢰할 수 있으며, 이동 에이전트를 보다 실용적으로 이용할 수 있다.

3. 이동 에이전트의 실행 결과 보호

앞에서 제시한 수집한 결과에 대해서 발생할 수 있는 두 가지 문제 중에서 첫번째 문제인 호스트가 이동 에이전트의 결과를 엿보는 문제는 현재의 암호학적 방법을 적용하여 쉽게 해결할 수 있다. 본 논문에서 언급하고 있는 실행 환경이 공개키 방식의 암호화 방식이 사용된다고 가정하면, 다음과 같은 방법으로 호스트가 이동 에이전트가 수집한 데이터를 엿보는 것을 쉽게 방지할 수 있다.

$$ER_n = E_{PK_{HP}}(R_n),\ SR_n = S_{SK_{H_n}}(R_n) \qquad ---(1)$$

ER_n : 호스트 n 에서 수집한 암호화된 결과
SR_n : 호스트 n 에서 수집한 결과에 대한 Digital Signature
$S_K(M)$: 메시지 M 을 비밀키 K 로 Digital Signature 를 붙임
$E_K(M)$: 메시지 M 을 공개키 K 로 암호화
R_n : 이동 에이전트가 호스트 n 에서 수집한 결과
PK_{HP} : HP 의 공개키
SK_H : H 의 비밀키

호스트가 이동 에이전트가 수집한 데이터를 이동 에이전트가 생성된 HP 의 공개키로 암호화하여, R_n 대신 ER_n 을 가지고 다니면, 다른 악의적인 호스트가 HP 의 비밀키를 알 수 없으므로 이 ER_n 를 엿볼 수 없다. 또한 이동 에이전트를 실행한 호스트의 Digital Signature 인 SR_n 를 붙임으로써, 호스트에서 얻은 결과 값이 위조되지 않았음을 보증할 수 있다. 그러나, HP 의 공개키로 암호화를 하고 Digital Signature 를 덧붙이더라도 앞에서 제시한 이동 에이전트의 결과를 조작하는 문제를 해결할 수는 없다. HP 의 공개키는 쉽게 얻을 수 있으므로, 다른 호스트의 실행 결과인 ER_n 값을 다른 값으로 대체 시킬 수 있고, 다른 호스트의 결과값에 해당하는 Digital Signature 를 악의적인 호스트의 Digital Signature 값으로 대치하는 경우, 사용자는 그 결과값의 위조여부를 판별할 수 없다.

따라서 위의 방법을 사용하면서 각각의 호스트에 수집한 결과값과 그 결과값의 유효성을 보증할 수 있는 방법이 필요하다. 본 논문에서는 단방향 해쉬 함수(one-way hash function)을 사용한 VL(Visit List)를 유지하여 이동 에이전트가 수집한 결과값을 보호하는 방법을 제시한다.

4. 단방향 해쉬 함수를 이용한 실행 결과의 보호

4.1 단방향 해쉬 함수

단방향 해쉬 함수는 그 실행속도가 매우 빠르며 다음과 같은 특성을 가지고 있고, 이 특성을 이용하여 호스트에 유일한 값을 사용하여 이동 에이전트를 보호할 수 있다.

$$h_{i+1} = H(h_i) \quad (H\text{: 단방향 해쉬 함수})$$

- 주어진 h_i에 대해서는, h_{i+1} 를 계산하기가 쉽다.
- 주어진 h_i에 대해서, $H(h_i)=h_{i+1}$ 가 되는 h 을 계산하기 어렵다.
- 주어진 h_i에 대해서, $H(h_i)=H(h_i^*)$와 같은 다른 메시지 h_i^* 를 찾기가 어렵다.

4.2 단방향 해쉬 함수를 이용한 VL 의 계산

이동 에이전트가 HP 에서 출발할 때, HP 는 VL(Visit List)를 다음과 같이 초기화 하여 생성한다. 이 VL 은 각각의 호스트에 유일한 단방향 해쉬 함수의 값의 체인 형태로 연결되어 있다.

$VL_0 = H(MA\_ID, T^*, N_0, HP\_ID)$ MA_ID : HP 에서 생성된 이동 에이전트의 ID
T_* : VL0 값을 생성할 당시의 시간값
N_0 : HP 가 생성한 random Nonce
HP_ID : 이동 에이전트를 생성한 HP 의 ID

VL 의 초기값을 계산한 후, 이동 에이전트가 n 번째 이동한 호스트에서 계산하는 VL 값은 다음과 같이 계산된다.

$$VL_n = H(VL_{n-1}, SRn) \quad \text{--- (2)}$$

VL_{n-1} : $n-1$ 번째 호스트에서 생성된 VL 값
SRn : n 번째 호스트에서 얻은 결과의 Digital Signature 값

VL 값만을 사용하면, 이동 에이전트의 결과값을 파기시킬 목적으로 호스트가 임의의 VL 값을 변조시키거나 삭제할 수 있다. 이를 보호하지 못하면, 이동 에이전트의 결과 값이 쉽게 그 정당함을 HP 로부터 인정받을 수 없게된다. 따라서 이 VL 값의 Checksum 을 유지하여, VL 값이 변조되거나 삭제되었을 때, VL 값을 변조하거나 삭제한 호스트를 찾아내어 VL 을 보호할 수 있다.

$$VLCn = H(VL0, VL1, \ldots, VLn) \quad \text{--- (3)}$$

위의 VL 리스트 값을 계산한 후, 앞에서 언급한 암호학 방법에 덧붙여 호스트에서의 결과값을 보호한다. 따라서, (1), (2), (3)을 하나로 정리하면 다음과 같다.

$$ER_n = E_{PK_{HP}}(R_n, VLC_n), SR_n = S_{SK_{H_n}}(R_n)$$
$$VL_n = H(VL_{n-1}, SR_n)$$
$$VLC_n = H(VL_0, VL_1, \ldots, VL_n)$$

4.3 VL 값을 이용한 이동 에이전트의 보호

단방향 해쉬 함수를 이용하여 이동 에이전트가 수집한 결과값을 보호할 수 있다. 방문한 호스트에서 각각의 호스트에 유일한 해당하는 단방향 해쉬 함수의 값을 계산한 후, 이 계산한 값을 체인 형태로 연결함으로써 특정의 호스트가 임의의 호스트에서 계산된 결과를 함부로 변경하지 못하도록 할 수 있다. 이 체인을 VL 로 표현하며, 각각의 호스트에 유일한 단방향 해쉬 함수의 값인 체인형태로 연결되어 있다. 이 VL 값을 이동 에이전트가 갖는 결과 값과 같이 암호화 하여 적재된다. 이동 에이전트가 HP 에 도착하였을 때, 사용자는 이동 에이전트가 가지고 있는 결과 값을 복호화 하며, 이 VL 의 각각의 값과 전체 VL 값이 유효한지 검사함으로써 쉽게 결과 값의 조작여부를 판단할 수 있다.

VL 을 이용하면 다음과 같은 기준에 맞게 이동 에이전트를 효율적으로 보호할 수 있으며, 그 변조와 삭제, 임의의 삽입으로부터 이동 에어전트를 보호할 수 있다.

데이터의 기밀성	Public Key 암호화 scheme 이 안전하고 믿을 만하다면, 공격자가 쉽게 암호화된 *ERn* 값을 쉽게 복호화 낼 수 없다.
Signautre 의 부인 불가	Digital Signature scheme 이 안전하고 믿을 만 하다면, 호스트는 자신이 행한 Digital Signature 에 대해 부정할 수 없다.
강력한 단방향 무결성	공격자가 결과값은 그대로 두고, Signature 값을 변경하는 경우,, Digital Signature 을 검증하는 과정에서 쉽게 찾아낼 수 있으며, Signature 값을 가지고 생성되는 VL 값과 VL 값의 Checksum 인 VLC 이 Signature 값과 일치하는 값인지 검증함으로써, 공격자는 Signature 값을 변경시킬 수 없다.
공개적으로 검증 가능한 단방향 무결성	VL 체인은 이동 에이전트를 받아들인 호스트는 *SRn* 값과 *VLn* 값을 조합하여 점검함으로써 빠르고 쉽게 점검할 수 있다.
임의의 VL 값의 삽입으로부터의 보호	VLC 값이 항상 마지막 단계에서 삽입한 체인의 Checksum 이므로 호스트가 새로운 값을 삽입하기 위해서는 VL 체인의 가장 마지막에 삽입을 해야 한다. 공격자가 임의로 VL 체인의 중간에 결과값을 삽입할 경우 VLC 의 값이 바뀜으로써 공격자는 항상 VL 체인의 가장 마지막에 결과값을 추가하여야 한다.
임의의 VL 값의 삭제로 부터의 보호	악의적인 호스트는 VL 체인의 특정 값이 삭제할 수 있지만, VLC 값을 암호화 해서 간직하고 있으므로, VL 체인의 삭제를 감지해 낼 수 있다.
체인값 위조 감지	VL 체인 값뿐 아니라 각각의 단계에서 VLC 값을 유지하고 있으므로, 공격자가 특정 위치의 VL 값을 변조함으로써 다른 특정한 호스트를 공격하려 한다면, VL 체인과 VLC 값의 검증으로 공격한 호스트의 위치를 감지해 낼 수 있다.

5. 결론 및 향후과제

VL 을 사용함으로써 이동 에이전트가 호스트에서 수집한 데이터를 체인형태로 표시하여 묶음으로써 악의를 가진 호스트가 함부로 이동 에이전트가 수집한 데이터를 변경, 위조 시키지 못하게 할 수 있다. 또한 VLC 의 값을 사용함으로써, 사용자나 HP 는 VL 리스트의 값이 위조한 호스트의 위치를 감지해 낼 수 있으므로, 보다 안전하고 정확한 실행 결과를 보증할 수 있다.

본 논문에서는 일방향 해쉬 함수를 사용하여 호스트의 결과에 대한 보호하는 법을 다뤘다. 그러나 악의적인 호스트는 이동 에이전트의 코드의 실행 순서를 마음대로 실행할 수 있으며 코드를 변조할 수 있고, 이동 에이전트의 실행은 호스트에 100% 의존하므로 이동 에이전트 코드의 변조를 완벽하게 보호할 수 있는 방법은 없지만, 이동 에이전트의 코드를 적정한 수준에서 보장할 수 있는 방법이 연구되어져야 할 것이다.

6. 참고 문헌

[1] Holger Peine: Security Concepts and Implementation in the Ara Mobile Agent System. 7th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 17-19th, Stanford University, USA.

[2] W. Farmer, J. Gurrman, and V.Swarup. Security for mobile agents: Authentication and state appraisal. In to appear in the proceedings of the European Symposium on Research in Computer Security(ESORICS), Lecture Notes in Computer Science, September 1996

[3] Baumann, J., Hohl, F., Rothermel, K., and Strasser, M. (1998) Mole - Concepts of a mobile Agent System, The World Wide Web Journal, speciall issue on Software Agents.

[4] Sun Microsystems. Java Security ,http://java.sun.com/security/whitepaper.ps

[5] Hohl, F.(1998) Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts, Springer Lecture Notes in Computer Science. http://www.informatik.uni-tuttgart.de/ipvr/vs/projekte/mole/vignabuch.ps.gz

[6] T. and Tschudin, Chr.: Towards Mobile Cryptography; the 1998 IEEE Symposium on Security and Privacy. http://www.icsi.berkeley.edu/~sander/publications/

[7] Sander, Tomas; Tschudin, Christian F.: Protecting Mobile Agents Against Malicious Hosts, in: Giovanni Vigna (Ed.): Mobile Agents and Security. pp 44-60. Springer-Verlag, 1998.

[8] Riordan, James; Schneier, Bruce: Environmental Key Generation Towards Clueless Agents, in: Giovanni Vigna (Ed.): Mobile Agents and Security. pp 15-24. Springer-Verlag, 1998

[9] Hohl, Fritz: An approach to solve the problem of malicious hosts. Universität Stuttgart, Fakultät Informatik, Fakultätsbericht Nr. 1997/03