이동 에이전트의 상태 변경을 위한 검출 기법[1)]

강동현*, 최정환*, 장현수*, 김구수**, 엄영익*
*성균관대학교 정보통신공학부
**동양대학교 정보통신공학부
e-mail : {kkangsu, themars, jhs4071, gusukim, yieom}@ece.skku.ac.kr,
gusukim@dyu.ac.kr

Detection Scheme for State Modification of Mobile Agent

Dong Hyun Kang*, Jung Hwan Choi*, Hyunsu Jang*, Gu Su Kim**,
Young Ik Eom*
*School of Information and Communication Engineering,
Sungkyunkwan University
**School of Information and Communication,
Dongyang University

요 약

이동 에이전트는 다른 이동 에이전트와의 협업을 위해서 서로간의 통신을 통해 정보를 교환할 수 있다. 그러나, 정보 교환시 악의적인 이동 에이전트에 의해 정보가 변경될 수 있는 보안 위협이 있다. 본 논문에서는 자바의 성능 감시에 사용하는 JVMPI의 프로파일러 에이전트를 이용하여 이동 에이전트의 정보 변경을 감시하는 기법을 제안한다. 감시되는 정보는 성능 테이블로 저장되며, JVMPI의 프로파일러 에이전트는 저장된 성능 테이블을 기반으로 악의적인 이동 에이전트의 정당하지 않은 정보의 변경을 검출한다.

1. 서론

이동 에이전트는 생성된 목적을 수행하기 위하여 여러 플랫폼으로 이주하며 플랫폼을 통해 또는 직접적으로 다른 이동 에이전트와 통신을 하여 정보의 교환을 할 수 있다.

이동 에이전트는 코드, 상태, 그리고 데이터로 구성된다. 특히, 상태는 프로그램 카운터, 레지스터, 스택, 그리고 저장소로 구성된다[1].

자신의 목적을 수행하기 위하여 이동 에이전트는 실행을 중지하고 다른 플랫폼으로 이주하여 재실행할 수 있다. 이동 에이전트는 이런 이동성을 바탕으로 플랫폼에서 사용자의 업무를 대신하여 수행하거나 다른 이동 에이전트와 통신을 통한 정보 교환으로 업무를 수행할 수 있다[2].

그러나 악의적인 이동 에이전트와의 통신으로 인해 에이전트의 정보가 열람되거나 정당하지 않게 변경될 수 있는 보안 위협이 있다. 이러한 보안 위협을 위해서 이동 에이전트를 제어하고 감시하는 응용프로그램 개발에 대한 연구도 진행되고 있다[3]. 이에 따라 본 논문에서는 자바의 성능 감시에 사용하는 JVMPI의 프로파일러 에이전트를 이용하여 이동 에이전트의 정보 변경을 감시한다. 감시되는 정보는 성능 테이블로 저장된다.또한, JVMPI의 프로파일러 에이전트는 저장된 성능 테이블을 기반으로 악의적인 이동 에이전트의 정당하지 않은 정보의 변경을 검출하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 에이전트와 에이전트 사이의 통신을 통한 정보의 교환으로 인해 발생할 수 있는 보안 위협과 자바의 JVMPI에 대한 기존 연구에 대하여 설명한다. 3장에서는 본 논문에서 제안하는 시스템의 구성과 정당하지 않은 정보의 변경을 검출하는 방법을 설명하며, 4장에서는 3장의 제안 기법의 유효성을 검사한다. 마지막으로 5장에서는 결론 및 향후 연구 계획을 설명한다.

2. 관련연구

2.1 에이전트 상호간의 보안 위협

이동 에이전트는 플랫폼에서 다른 이동 에이전트와 직접적으로 또는 매개체를 이용하여 통신을 할 수 있다. 본 절에서는 이동 에이전트간의 통신에서 발생 할 수 있는 보안 위협에 대한 기존 연구에 대하여 설명한다[2, 4].

(1) 위장(Masquerade): 악의적인 에이전트가 신뢰할 수 있는 다른 에이전트의 민감한 정보를 추출하기 위하여 자신을 신뢰할 수 있는 제 삼자로 위장할 수 있다.

1) 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-(C1090-0801-0046))

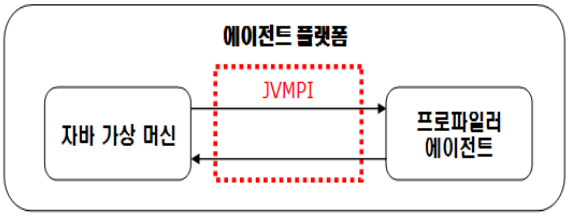
(2) 서비스 거부(Denial of Service): 악의적인 에이전트는 동일한 메시지를 다른 에이전트에 지속적으로 전송 또는, 무차별적인 에이전트의 메시지 전송처럼 다른 에이전트에게 서비스 공격을 할 수 있다.
(3) 부인(Repudiation): 에이전트가 트랜잭션 또는 통신에 참여한 후 악의적인 에이전트의 조작이나 삭제로 인해 참여에 대한 트랜잭션 또는 통신이 발생한 사실을 부정할 때 송수신 부인이 발생한다.
(4) 불법 접근(Unauthorized Access): 에이전트는 다른 에이전트와 버퍼 오버플로우 또는 상태의 리셋과 같은 일반적인 방법에 의해, 또는 에이전트의 데이터, 코드에 대한 접근 및 수정에 의해 직접 통신할 수 있다. 에이전트 코드의 수정은 신뢰받는 에이전트를 악의적으로 변경하거나, 도청으로 인한 에이전트의 활동 정보를 얻을 수 있다.

2.2 JVMPI

자바 2 SDK 1.6기반에서 사용하고 있는 JVMTI(Java Virtual Machine Tool Interface)와 SDK 1.5기반에서 사용되는 JVMPI(Java Virtual Machine Profiler Interface) 모두 양방향의 인터페이스를 지원하여 자바 성능 감시에 사용된다. JVMTI는 감시하기 위하여 바이트 코드로 변환하는 작업이 필요하다. 본 논문은 단순한 상태 정보의 변경 함수에 대해서 감시를 실시하기 때문에 변환 작업을 하지 않는 JVMPI을 이용하여 이동 에이전트 시스템에 접목하려고 한다[5].

JVMPI의 양방향 인터페이스 중 자바 가상 머신에서 프로파일러 에이전트의 방향은 자바 가상 머신에서 특정한 함수, 쓰레드의 시작과 같은 이벤트가 발생할 때 JVMPI 인터페이스에 의해 호출되고 프로파일러 에이전트는 해당 이벤트의 처리 코드를 가진 파일이다.

프로파일러 에이전트에서 자바 가상 머신의 방향은 이벤트의 활성화 또는 비활성화의 필요에 따라 제어하기 위해 호출한다. 그림 1은 JVMPI를 포함한 자바 가상 머신 과정을 나타낸다.

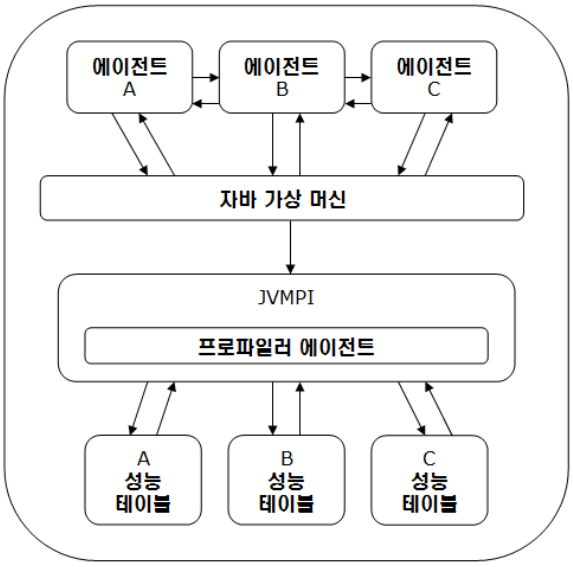


(그림 1) 자바 가상 머신 과정

3. 변경 검출 기법

3.1 시스템 구성

본 절에서는 제안하는 시스템 구성에 대하여 설명한다. 제안하는 플랫폼은 이동 에이전트, 자바 가상 머신, 성능 테이블, 그리고 검출 및 확인을 위한 JVMPI로 구성된다. 성능 테이블은 이동 에이전트의 정보 변경에 대한 감시 정보이며, 이동 에이전트를 식별하기 위하여 플랫폼은 각 이동 에이전트에 AID(Agent ID)을 부여한다. 그림 2는 제안하는 플랫폼의 시스템 구성도를 보인다.



(그림 2) 플랫폼 시스템 구성도

프로파일러 에이전트는 이동 에이전트에서 정보 변경을 위한 함수 호출 시 이동 에이전트에 대한 상태 정보, 또는 실행 정보를 감시하여 성능 테이블에 저장하는 기능을 한다. 또한, 프로파일러 에이전트는 성능 테이블에 저장된 정보를 기반으로 정당하지 않은 정보의 변경을 검출한다. 성능 테이블의 각 엔트리는 표 1과 같은 필드들로 구성된다.

<표 1> 성능 테이블 엔트리의 필드

필드명	설 명
AID	플랫폼에서 부여한 이동 에이전트의 ID 저장
Update	정보의 변경 시점 저장
APC	프로그램 카운터
Title	변경되는 정보의 변수 명
Data	변경되는 정보의 변수 값

AID는 이동 에이전트가 플랫폼의 외부로부터 이주하여 올 때 플랫폼으로부터 부여 받는 식별자이며, Update는 정보의 변경으로 인한 함수의 호출 간격을 시점으로 저장한다. APC(Agent Program Counter)는 정당한 정보의 변경이 이루어졌을 때, 프로그램의 카운터이다. Title은 정보의 변경 시 변경되는 변수의 명을 저장하며, Data는 변경되는 변수의 값이다.

프로파일러 에이전트에서 각 이동 에이전트의 성능 테이블을 작성하고 정당하지 않은 정보의 변경을 검출하기

위한 함수를 가지고 있다. 표 2는 JVMPI에서 사용하는 함수 및 필드를 보인다.

<표 2> JVMPI에서 사용하는 함수 및 필드

필드명	설 명
$Agent_i$	이동 에이전트의 전체 정보
AID_i	정보 변경 에이전트
$Update_i$	정보의 변경 시점
APC_i	프로그램 카운터
Notification	이동 에이전트에게 통보
$Title_i$	변경되는 정보의 이름
$Data_i$	성능 테이블의 Title에 해당하는 최근 변수의 값
$Data_j$	새로운 값이 들어갈 변수 값

성능 테이블은 이동 에이전트가 플랫폼의 외부로부터 이주하여 올 때 플랫폼으로부터 부여받은 AID와 이동 에이전트 정보를 초기 값으로 사용하는 성능 테이블이며 그림 3의 (1)번과 같은 동작으로 생성된다.

APA(Agent Performance Authenticator)는 정보 변경을 위한 함수 호출시 정당하지 않은 검출을 위하여 에이전트에서 생성된다. 생성된 APA는 이동 에이전트와 성능 테이블 매핑에 사용하며 그림 3의 (2)번과 같은 동작으로 생성된다.

K_G는 플랫폼의 공유키이다. AID를 K_G로 암호화함으로써 악의적인 에이전트에 의해 에이전트의 AID가 유출, 또는 변경되는 것을 방지할 수 있다[6].

3.2 정당하지 않은 정보의 변경 검출

본 절에서는 악의적인 이동 에이전트로부터 정당하지 않은 정보의 변경을 검출하는 기법에 대하여 설명한다.

본 논문에서는 이동 에이전트가 플랫폼의 외부로부터 이주한 후 이동 에이전트의 정보 변경은 $Update_i(1 \le i \le n)$번 수행을 가정한다.

프로파일러 에이전트에서 이동 에이전트 정보 변경 검출 절차는 그림 3과 같다.

(1)N : C_table(AID_i, $Title_i$, $Data_j$, APC_i, $Update_i$)
(2) : $APA_i = E_{K_G}(AID_i)$

(3)B → A: $PT_i = E_{K_{RS}}(APA_i)$
(4) : $Data_i$ = GetData(PT_i, $Title_i$)
(5) : $Result_i$ = Compare($Data_i$, $Data_j$)

(6) : APC_i = GetPC(PT_i, $Update_i$)
(7) : Commit(PT_i, $Title_i$, $Data_j$, APC_i, $Update_i$)

(8)B → A: Datai = GetData(PT_i, $Title_i$)
(9) : Resulti = Compare($Data_i$, $Data_j$)

(10) : APC_i = GetPC(PT_i, $Update_i$)
(11) : Commit(PT_i, $Title_i$, $Data_j$, APC_i, $Update_i$)

(12) : Resulti = Compare($Agent_1$, $Agent_n$)
(13)D : D_table(PT_i)

(14) : Notification($Agent_i$)

(그림 3) 정보변경 검출 절차

이동 에이전트에서 정보 변경을 위한 함수의 호출시 JVMPI의 프로파일러 에이전트는 그림 3의 (3), (4), (5)번과 같은 동작을 수행한다.

프로파일러 에이전트는 수행된 결과를 자바 가상 머신으로 전달한다.

자바 가상 머신에 전달된 결과 True인 경우: 현재의 상태 정보, 실행 정보를 변경하기 위하여 저장한다. 정보 변경을 위한 함수의 포인터를 반환 시, JVMPI의 프로파일러 에이전트는 성능 테이블을 기록하여 다음 검출에 사용하며 그림 3의 (6), (7)번과 같은 동작을 수행한다.

자바 가상 머신에 전달된 결과 False인 경우: 현재의 상태 정보, 실행 정보가 정당하지 않은 변경으로 인해 손실 되었으므로 저장하지 않는다. 정보 변경을 위한 함수의 포인터를 반환 시, JMVPI의 프로파일러 에이전트는 이동 에이전트의 홈 플랫폼에게 악의적인 변경으로 인해 손실 되었음을 통보하기 위하여 그림 3의 (14)번과 같은 동작을 수행한다.

현재의 플랫폼에서 다른 플랫폼으로 이동 에이전트가 이주할 때, JVMPI의 프로파일러 에이전트는 현재의 플랫폼에서 정당하지 않은 정보의 변경을 확인하기 위해 성능 테이블의 초기 정보의 값과 현재의 정보 값을 비교한다. 초기 정보의 값과 현재의 정보 값을 비교하여 플랫폼에서 사용하지 않은 정보들에 대한 정당하지 않은 정보의 변경이 있는지 검출하며 그림 3의 (12)번과 같은 동작을 수행한다.

자바 가상 머신에 전달된 결과 True인 경우: 성능 테이블을 삭제하여 플랫폼의 자원을 반납하며 그림 3의 (13)번과 같은 동작을 수행한다.

자바 가상 머신에 전달된 결과 False인 경우: 플랫폼에서 사용하지 않은 정보에 정당하지 않은 변경으로 인해 손실 되었으므로 JMVPI의 프로파일러 에이전트는 이동 에이전트의 홈 플랫폼에게 악의적인 변경으로 인해 손실 되었음을 통보하기 위하여 그림 3의 (14)번과 같은 동작을 수행한다.

4. 제안 기법 검증

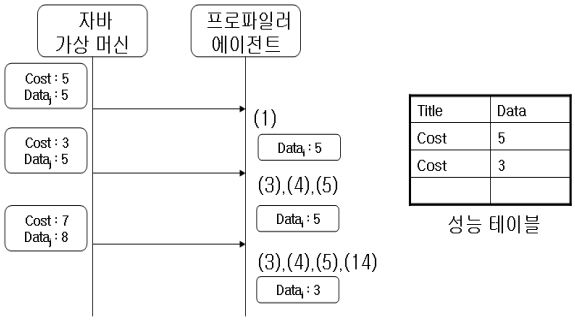
본 절에서는 3장에서 설명한 에이전트의 정당하지 않은 정보의 변경 검출에 대한 유효성을 검증한다.

만약 항공권을 조사하는 이동 에이전트 기반 시스템이 존재 한다면. 사용자는 각각의 항공사를 방문 또는 홈페이지를 통한 항공권의 가격을 조사하고 비교하는 작업을 이동 에이전트에게 대행시킬 수 있다. 위의 가정을 가지고 아래와 같은 시나리오를 구성한다.

◉ 이동 에이전트 A는 항공권의 가격을 조사와 비교를 사용자를 대신하여 수행하며, S항공사와 U항공사의 정보를 다른 에이전트로부터 받아 저장하고 있다. 이동 에이전트 A는 플랫폼의 외부로부터 이주해 왔으며 플랫폼은 AID와 이동 에이전트의 정보를 초기 값으로

성능 테이블을 생성한다. 이동 에이전트 A는 항공권 판매 대행 이동 에이전트 B와 통신을 통한 정보의 교환으로 A항공사, B항공사의 항공권 가격 정보를 받고 홈 플랫폼으로 돌아간다.

Case 1. 통신 중인 에이전트 혹은 동일 플랫폼상의 이동 에이전트로부터 정당하지 않은 정보의 변경으로 손실된 정보가 플랫폼에서 사용하는 경우는 그림 4와 같으며, 그림 3의 (5)번에 의해 결과가 정당하지 않게 변경되었다는 것을 검출하고 이동 에이전트의 홈 플랫폼에 통보하기 위해 그림 3의 (14)번과 같은 동작을 수행한다.



(그림 4) Case 1의 검출 과정

Case 2. 통신 중인 에이전트 혹은 동일 플랫폼상의 이동 에이전트로부터 정당하지 않은 정보의 변경으로 손실된 정보가 플랫폼에서 사용하지 않는 경우는 그림 5와 같으며, (12)에 의해 정보가 정당하지 않게 변경되었다는 것을 검출하고 이동 에이전트의 홈 플랫폼에 통보하기 위해 그림 3의 (14)번과 같은 동작을 수행한다.

Title	Data
Name	A
Home Platform	123.125.67.12
Member	7
Goal	Survey Cost
S Airport	40000
Seat	50
U Airport	70000
Seat	100

(a) 최초 성능 테이블 정보

Title	Data
Name	A
Home Platform	123.125.67.12
Member	7
Goal	Survey Cost
S Airport	90000
Seat	0
U Airport	70000
Seat	100

(b) 최종 성능 테이블 정보

(그림 5) Case 2의 검출 예

4. 결론

본 논문은 자바의 성능 감시에 사용하는 JVMPI의 프로파일러 에이전트를 이용하여 이동 에이전트의 정보 변경을 감시하며, 감시되는 정보는 성능 테이블로 저장했다. 또한, JVMPI의 프로파일러 에이전트는 저장된 성능 테이블을 기반으로 악의적인 이동 에이전트의 정당하지 않은 정보의 변경을 검출하는 기법을 제안했다.

플랫폼은 각 이동 에이전트를 식별하기 위한 AID을 부여한다. 부여된 AID의 유출 및 변경을 보호하기 위하여 플랫폼의 공유키로 암호화하고 개인키를 이용하여 복호화한다.

향후 연구로는 본 논문에서 제안한 기법을 구현하고 검출에 대한 복구과정에 대하여 연구할 예정이다.

참고문헌

[1] M. Farmer, Joshua D. Guttman, and Vipin Swarup, "Authentication and State Appraisal," Proc. of the 4th European Symposium on Research in Computer Security, ESORICS, pp. 118-130, Sep. 1996.

[2] W. Jansen and T. Karygiannis, "Mobile Agent Security," NIST Special Publication 800-19, Aug. 2003.

[3] P. Bellavista, A. Corradi and C. Stefanelli, "Monitor and Control of Mobile Agent Applications," ACM OOPSLA Workshop on Experiences with Autonomous Mobile Objects and Agent Based Systems, Minneapolis, USA, Oct. 2000.

[4] W. Jansen, "Countermeasures for Mobile Agent Security," Computer Communications, Elsevier, pp. 1667-1676 Oct. 2000.

[5] Sun, http://java.sun.com

[6] 김재곤, 김구수, 엄영익, "홈 네트워크 환경에서 다중 도메인을 지원하는 공유키 및 공개키 기반의 이동 에이전트 인증 기법," 정보보호학회논문지, Vol.14 No.5, pp. 109-119, Oct. 2004.