ML/RBBAC(Multi-Level/Role-Behavior Based Access Conrtol) 접근통제

신욱*, 이동익*

*광주과학기술원 정보통신공학과

ML/RBBAC(Multi-Level/Role-Behavior Based Access Conrtol)

Wook Shin* and Dong-Ik Lee*

*Dept. of Info. and Comm., Kwang-Ju Institute of Science and Technology

요 약

본 논문은 현재 안전한 운영체제 개발에 사용되고 있는 강제 접근통제(MAC)와 역할 행위 기반 접근통제(RBBAC)의 혼합 기법인 다중 등급 역할 행위 기반 접근통제 (ML/RBBAC)를 제안하고, 접근통제 모델을 기술한다. ML/RBBAC은 안전한 운영체제가 MAC과 RBBAC을 동시 지원하기 위한 최적의 방법이며, 기존 MAC의 단점인 지나치게 엄격한 정보흐름 통제 및 보안 관리 부담의 증대 문제를 해결한다.

I. 서론

현재 안전한 운영체제의 설계를 위하여 사용하고 있는 접근통제 모델은 강제 접근통제(Mandatory Access Control: MAC)이다[1], [2], [3], [4], [5], [6], [7], [8]. MAC은 기존 운영체제에서 채용되던 임의 접근통제(Discretionary Access Control: DAC)와는 달리, 정보 흐름 통제를 통해 보다 강력한 안전성을 제공하며 중앙 집중적인 보안 관리를 가능하게 한다[9], [10], [11], [12]. 반면, 지나치게 엄격한 정보 흐름 통제로 인하여 시스템 가용성 및 유연성을 저하시키거나 보안 관리상의 부담을 증가시키기도 한다[13], [14]. 따라서, 이를 해결하기 위한 방법으로 안전한 운영체제 설계에 역할기반 접근통제(Role Based Access Control: RBAC)를 도입하고자 하는 시도가 있어왔다[4], [6], [8]. 그러나 이들 RBAC 적용 사례를 살펴보면, DAC 및 MAC과는 별도로 RBAC을 구현, 지원하고 있음을 알 수 있다. 즉, 시스템이 MAC 체계 하에서 작동하고 있는 도중, RBAC으로 전환하기 위해서는 재부팅이 필요함을 의미한다. 따라서, RBAC의 도입이 기존 MAC의 단점 보완을 위한 목적에서 이루어 지는 경우, 기존 연구 사례의 접근 방식은 궁극적인 해결책이 될 수 없다. 보다 적절한 방법은 상이한 접근통제 정책을 혼합하여 견고하게 정의된(well-defined) 하나의 접근통제 모델을 수립한 후 구현하는 것이다. 혼합 모델은 일관된 뷰(view)를 제공하고, 상이한 정책간의 충돌 가능성을 제거한다.

앞서 언급한 DAC 기반 MAC 적용 사례들 역시 혼합 모델을 통한 접근 방법에 해당한다. MAC의 속성인 ds-property[15] 내에 DAC을 포함하고 있기 때문에, 기존 DAC 기반 운영체제에 나머지 MAC의 속성인 ss-, *-property만을 구현하는 것은 결과적으로 혼합 모델 MAC을 무리 없이 적용하는 것과 같다.

그러나, MAC과 RBAC의 융합은 DAC과 MAC의 경우처럼 간단히 해결되지 않는다. 이는 두 기법이 서로 상이한 속성을 갖고있기 때문이며, 따라서 새로운 혼합 기법의 고안을 필요로 한다. 이에, 본 논문에서는 두 접근통제 기법의 합리적인 동시지원을 위하여 혼합 기법을 구성하고, 접근통제 모델을 기술한다. 단, 본 논문에서는 RBAC 대신 역할 행위 기반 접근통제 (Role Behavior Based Access Control: RBBAC)를 사용한다[16], [17]. RBBAC은 RBAC을 확장한 접근통제 방법으로 역할, 행위 개체의 도입을 통해 사용자와 시스템 자원의 추상화를 꾀하고 있다.

논문의 구성은 다음과 같다. 2장에서

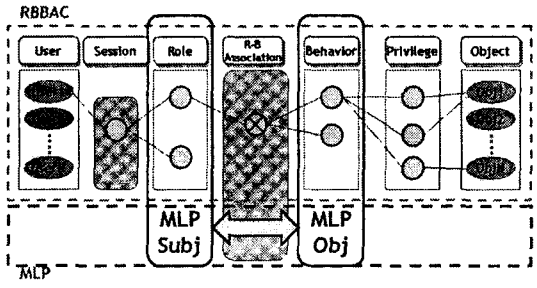

그림 1: ML-RBBAC 개념 다이어그램

ML-RBBAC의 개념에 대해 소개하고, 3장에서 ML-RBBAC 접근통제 모델에 대해 기술하며, 4장에서 결론을 맺는다.

II. ML-RBBAC

MAC을 시스템에 적용함으로써 발생하는 문제를 해결하기 위해서는 RBBAC의 추상화 개념을 도입해야 한다. 그러나, 앞서 언급한 바와 같이 상이한 두 정책의 속성을 동시에 지원하고자 할 경우, 두 정책의 특성을 융화하여 새로운 기법을 구성하는 것이 합리적이다. 그림 1은 ML-RBBAC의 개념을 도식화 한 것이다.

ML-RBBAC의 기본적인 아이디어는, RBBAC의 접근 주, 객체인 역할과 행위 개체를 등급화하여 접근통제를 행하도록 하는 것이다. 역할 및 행위 개체는 RBBAC에서 비롯된 개념이며, 접근주체와 개체를 등급과 부서로 분류하여 통제하는 것은 MAC에 기반을 둔 개념이다.

ML-RBBAC은 기존 안전한 운영체제에 적용된 MAC의 장점인 강력한 접근통제를 지원하는 동시에, 단점으로 지적되어 왔던[18]

- 지나치게 엄격한 정보 흐름 통제로 인한 불편
- 보안 관리상의 부담

을 해결한다. 엄격한 MAC의 제약을 완화하고 응용 프로그램 수행을 보다 수월하게 하기 위해서는 다음과 같은 방법을 사용한다. 등급과 부서로 분류된 역할 및 행위간 접근통제 과정에 예외를 명시할 경우, MAC에서는 허용되지 않았던 정보 흐름 역행을 허용한다. 즉, RBBAC의 제약에만 위배되지 않는다면, 보안관리자의 용인에 의해 MAC의 등급 및 부서 검사는 행하지 않는다.

또한, 보안 관리 상의 부담 감소를 위하여 역할과 행위라는 접근 주, 객체 추상화 도구를 이용한다. 개체의 추상화를 통한 보안관리 부담 감소는 이미 [18], [19]에서 언급된 바 있다.

III. RBBAC Model

접근통제 모델은 제안된 접근통제 정책을 정형화한 결과이다. 따라서, 모델은 접근통제 기법이 내포하고 있는 모호성을 제거하고 명확성을 부여하여 구현상의 편의를 제공한다. 이 장에서는 ML-RBBAC 모델을 구성한다.

ML-RBBAC 모델의 구성요소는 다음과 같다.

- U: 사용자의 집합, U의 원소 u_i는 시스템 내 사용자 식별자에 일대일 대응된다. 식별자는 사용자를 대표하기 위해 시스템이 사용자에게 부여한 것이며, 시스템 내에서 유일하고 상호 구별 가능하다.
- R: 역할의 집합. R의 원소 r_i는 (i, SUB-R(r_i))로 정의된다. 이 중, i는 역할 u_i의 식별자이며, SUB-R(r_i)는 R의 부분집합으로, r_i의 하위 역할들을 의미한다. 하나의 역할이 정의되는 시점에서 한 역할의 하위 역할들이 정의되며, 정의 후 변경하지 못한다.
- S: 세션의 집합. S의 원소 s_i는 (u_i , R_n)로 정의된다. 이 때, R_n는 R의 부분집합이다. 따라서, 하나의 세션은 한 사용자와 복수의 역할과의 관계이며, 실행 시점에서 사용자가 활성화한(activated) 역할에 관한 정보를 가진다. 사용자는 특정 작업의 수행을 위해 하나의 세션을 선택함으로써, 하나 이상의 역할을 취득한다(acquire)[16], [17].
- B: 행위의 집합. B의 원소 b_i는 (i, SUB-B(b_i), O)로 정의된다. SUB-B(b_i)는 B의 부분집합으로, b_i의 하위 행위들을 의미한다. 또한, O는 부분 순서(Partial Order) 정보이며, B에 속한 하위 행위들이 실행되어야 할 순서이다.
- P: 기본 권한의 집합. P의 원소 p_i는 (obj, mode)로 정의된다. obj는 시스템의 자원 객체이며, mode는 이에 접근하는 접근 모드로서 read, write, create, delete의 4가지 상태를 가진다. 시스템 자원 객체의 범위는 접근통제의 범위를 어디까지 설정하느냐에 따라 좌우된다. 파일 시스템, 네트워크 자원, 프로세스 스케줄, 프로세스간 통신 등은 일반적

인 시스템 자원이다.

- O: 부분 순서 집합으로, O의 원소 o_i와 o_j간에 '<='의 순서가 'o_i <= o_j'와 같이 정의되어 있을 경우, o_j가 o_i보다 먼저 수행되어야 함을 의미한다.
- DSOD, SSOD: 정적의무분리 원칙(Static Separation of Duties) 및 동적 의무분리 원칙(Dynamic Separation of Duties)의 집합이다. DSOD는 (Ri, Rj)로 정의되며, 역할집합 Ri와 Rj가 한 사용자에게 동시에 배정 허가(authorized)될 수 없음을 의미한다. SSOD는 (Ri, Rj)로 정의되며, 한 세션 내에 Ri와 Rj가 동시에 속할 수 없음을 의미한다.
- A: ML-RBBAC을 위해 사용되는 정보표현 테이블.
- F: ML-RBBAC을 위해 사용되는 함수의 집합.

역할과 행위는 ML-RBBAC을 대표하는 개체이다. 역할은 다음과 같은 의미를 가진다.

- 시스템 내 접근 주체인 사용자를 추상화 한다.
- 실세계 조직에서 사용자가 차지하고 있는 지위를 접근통제 체계 내로 반영한다.
- 사용자의 등급 및 부서를 위임 받아 접근통제에 참여함으로써, MAC 논리를 수용한다.
- 정적, 동적 의무분리를 지원한다.

행위는 다음과 같은 의미를 갖는 개체이다.

- 접근 객체인 기본 권한의 추상화 도구이다.
- 자원의 등급 및 부서 반영을 통해 접근통제에 참여함으로써, MAC체계를 수용한다.
- 기본 권한의 집합을 포함하고, 그들 사이의 부분 순서를 정의할 수 있어, 시스템 오퍼레이션의 순차적 실행을 지원한다.

ML-RBBAC 접근통제의 시행을 위하여 사용하는 중요 자료구조는 다음과 같다.

- AT_User-Role: 사용자와 역할 간의 연관관계 정보를 저장하기 위한 테이블.
- AT_Session-Role: 세션과 역할 간의 연관관계 정보를 저장하기 위한 테이블.
- AT_Behavior-Privilege: 행위와 기본 권한간의 연관관계 정보를 저장하기 위한 테이블.
- AT_Role-Behavior: 역할과 행위간의 연관관계 정보를 저장하기 위한 테이블. 전통적인 접근통제 시스템에서 기술하는 주체-객체간 접근 규칙과 의미상 동일한 정보를 담고 있다.
- AT_DSOD: 동적 의무분리 관계에 있는 역할들에 관한 정보 저장 테이블.
- AT_SSOD: 정적 의무분리 관계에 있는 역할들에 관한 정보 저장 테이블.
- AT_LevelCategory: 각 사용자와 객체의 등급과 부서정보를 저장하기 위한 테이블.
- TBL_AccessHistory: 발생한 접근에 관한 정보를 저장하는 저장소. 저장된 정보는 행위 내 기본 권한이 순서에 맞게 실행되었는지를 검사하기 위해 쓰이며, 감사 서비스 지원을 위해 다른 보안 서비스 모듈로 이동할 수 있다.

다음은 ML-RBBAC에서 사용되는 함수들이다.

(단, L과 C는 각각 등급(Level) 및 부서(Category)를 의미하며, X^p는 X의 Power set을 의미한다.)

☐ f_RelRoles() : $S \rightarrow R^P$

- AT_Session-Role을 참조하여 세션에 연관된 역할의 집합을 반환한다.

☐ f_AuthRoles(): $U \rightarrow R^P$

- AT_User-Role을 참조하여 사용자에게 허용가능 한 역할의 집합을 반환한다.

☐ f_SelSession(): (U, S)→{TRUE, FALSE}

- 다음을 만족하면 TRUE를 반환한다.
  - 사용자 u가 하나의 세션 s를 선택
  - f_RelRoles(s)의 반환값이 f_AuthRoles(u)의 부분집합

☐ f_SessionLevel(): $S \rightarrow L$

- 세션을 선택한 사용자의 등급을 반환한다.

☐ f_SessionCategory(): $S \rightarrow C^P$

- 세션을 선택한 사용자의 부서를 반환한다.

☐ f_AcqRoles(): (U, S)→R^P

- f_RelRoles(s)를 호출하여 사용자가 선택한 세션에 연결된 역할집합을 반환한다. 즉, 사용자가 현재 취득하고 있는 역할을

반환한다. 따라서, f_SelSession(u, s)의 반환 값이 TRUE여야 한다.

☐ f_DSODRoles(): $R \rightarrow R^P$

■ 한 역할과 동적 의무분리 관계에 있는 역할을 반환한다.

☐ f_SSODRoles(): $R \rightarrow R^P$

■ 한 역할과 정적 의무분리 관계에 있는 역할을 반환한다.

☐ f_ConfirmDSOD(): U→{TRUE, FALSE}

■ 위의 함수들을 이용, 현재 시점에서 사용자가 취득한 역할들이 서로 동적의무 분리관계에 있는지를 확인한 후, 모두 동적 분리되어 있는 경우 TRUE를 반환한다.

☐ f_ConfirmSSOD(): U→{TRUE, FALSE}

■ 위의 함수들을 이용, 사용자에게 허용된 역할들이 서로 정적으로 분리되어 있는지를 확인한 후, 정적으로 분리되어 있을 경우, TRUE를 반환한다.

☐ f_RelPrivileges(): $B \rightarrow P^P$

■ AT_Behavior-Privileges를 참조하여, 특정 행위에 연관된 기본 권한의 집합을 반환한다.

☐ f_ConfirmOrder(): B→{TRUE, FALSE}

■ 어떤 행위 내의 기본 권한들이 순서에 맞게 실행되었을 경우 TRUE 값을 반환한다

ML-RBBAC 모델에서 사용하는 접근통제 규칙의 형태는 그림 2와 같이 표현할 수 있다.

f_accessConfirm(x)는 접근행위를 허용할 지의 여부를 결정하기 위하여 사용되는 함수이다. x는 (r, b, cond)의 triple로 정의되며, r은 역할의 원소, b는 B의 원소이다. 매개변수 cond가 가질 수 있는 값은 MLS 또는 NULL 값이다. 이 함수는 주어진 역할 r과 행위 b사이의 연관 관계가 접근통제 규칙에 설정되어 있는지를 검사하며, cond 값이 MLS일 경우, 역할이 접근 주체인 사용자로부터 상속받은 등급 및 부서 정보와 행위가 자신에게 속한 기본 권한으로부터 얻은 등급 및 부서 정보를 바탕으로 등급 및 부서 비교를 행할 수 있다. 따라서, cond는 다단계 접근통제의 엄격한 등급 비교를 완화할 수 있는 통로가 된다. 그림은 cond 값에 따른 f_AccessConfirm() 함수의 접근 결정 논리를 보여주는 그림이다.

시스템 내 모든 접근이 합법적일 경우 시스템은 안전하다고 할 수 있다. 또한, 사용자 u와 기본권한 p에 대하여, 접근 access(u, p)가 다음 조건을 모두 만족할 경우 그 접근을 합법적이라 한다.

■ 조건 1: $\exists s \in S$인 s에 대하여, f_SelSession(u, s) = TRUE이다. 즉, 사용자 u가 세션 s를 선택하였음을 의미한다.

■ 조건 2: $\exists x \in X$ such that x = (r, b, cond)를 만족한다. 즉, 역할 r과 행위 b의 연관 관계가 접근통제 규칙에 명시되어 있다.

■ 조건 3: $r \in$ f_RelRoles(s)를 만족한다. 즉, 역할 r은 세션 s에 연결되어 있다.

또한 다음과 같은 조건들이 만족되어야 한다.

■ 조건 4: $\exists p \in P$ such that $p \in$ f_RelPrivilege(b)를 만족한다. 이때의 p는 행위 b와 연관되어 있다.

■ 조건 5: f_ConfirmDSOD(u)가 TRUE를 반환한다. 현재, 사용자 u에게 배정된 역할들이 서로 동적 의무분리 관계에 있어서는 안된다.

■ 조건 6: f_ConfirmSSOD(u) 가 TRUE를 반환한다. 사용자 u에게 배정될 수 있는 역할들이 서로 정적 의무분리 관계에 있어서는 안된다.

■ 조건 7: f_ConfirmOrder(b) 가 TRUE를 반환한다. 행위 b내의 기본 권한들이 명시된 바에 따라 순서에 맞게 실행되어야 한다.

■ 조건 8: f_ConfirmAccess(x) 가 TRUE를 반환한다. 만약 x의 cond가 MLS를 따르도록 지시할 경우, 해당 접근은 MLS의 제약조건을 위배할 수 없다.

```
/* The body of f_AccessConfirm( x ) */
/* parameter is x=( R, B, cond ) */
/*  MLS Access condition */
/*  X( L, C ): L  the level of X, C  the category of X */
/*  ( L, C ) <= ( L, C ): L is lower than L, and C is a
subset of C */

  If (cond = MLS) {
    case (mode of privilege in B) {
       WRITE: if(R( L, C ) <= B( L, C )) then return TRUE
       READ: if(R( L, C ) >= B( L, C ) ) then return TRUE
            .
            .
    }
    return FALSE;
}
```

그림 2: ML/RBBAC 내의 접근통제 규칙

IV. 결론

본 논문에서 제안한 접근통제 기법인 ML/RBBAC은 MAC과 RBBAC을 정책간의 충돌 없이 동시에 지원하는 안전한 운영체제를 설계하도록 하며, 다음과 같은 장점을 제공한다.

첫째, 유연한 보안 서비스를 가능하게 한다. 이는, MAC의 엄격한 정보 흐름 통제를 완화함으로써 얻어진다. 보안 관리자가 명시할 경우, MAC의 규칙을 우회할 수 있다. 그러나, 이 경우에도 RBBAC의 규칙에 어긋나는 접근 행위는 허용되지 않으므로, 안전한 접근을 보장할 수 있다.

둘째, RBBAC을 도입함으로써, 보안 관리의 부담을 감소시킨다. 또한, 계층적 구조로 설계된 시스템에서의 자원 계층 확장성을 제공하는 RBBAC의 장점을 통해 시스템 자원 관리를 효율적으로 할 수 있게 된다.

또한, 논문에서 제안한 접근통제 기법을 모델화함으로써, 구현하기 용이하도록 하였다.

제안된 접근통제 기법은 현재 광주과학기술원에서 개발된 강제 접근통제 기반 안전한 운영체제인 CSRL 시스템에 적용 중이다.

참고문헌

[1] http://www.tis.com/docs/research/

[2] http://www.cs.utah.edu/~sds/synergy/

[3] http://www.cs.utah.edu/flux/fluke/html/flask.html

[4] http://www.nsa.gov/selinux/

[5] "UNICOS Multilevel Security (MLS) Feature User's Guide", SG-2111 10.0, Cray Research, Inc, 1990. http://rcs21.urz.tu-dresden.de/ebt-bin/nph-dweb/dynaweb/unicos_mk/2111_10.0/@Generic__BookTextView

[6] http://www.rsbac.de/

[7] C. W. Lee, H. K. Kim, and T. G. Park, "Implementation of L4 Linux-MLS with Security Servers", *Proc. of Joint Workshop on Information Security and Cryptology 2000, Okinawa*, Japan, Jan. 2000.

[8] J. G. Ko et al. "Design and Implementing for Secure OS based on Linux", *Proc. of WISA 2000*. pp.175-182, Nov. 2000.

[9] R. Sandhu and P. Samarati, "Access Control: Principles and Practice", *IEEE Communications*, Vol. 32, No. 9, September 1994

[10] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-Based Access Control Models", *IEEE Computer*, pp. 38-47, Vol. 29, No. 2, Feb. 1996.

[11] DOD 5200.28-STD, "Trusted Computer System Evaluation Criteria", Department of Defense, Dec. 1985.

[12] W. E. Boebert, and C. T. Ferguson, "A Partial Solution to the Discretionary Trojan Horse Problem", 9th Security Conference, DoD/NBS, pp 141-144, September 1985.

[13] E. G. Amoroso, "Fundamentals of Computer Security Technology", ISBN 0-13-108929-3, Prentice-Hall PTR., 1994.

[14] M. V. Joyce, "Access control and Applications on Trusted Systems", Computer Security Applications Conference, 1992. Proceedings, pp. 160-167, 8th Annual , 1992

[15] D. Gollmann, "Computer Security", ISBN 0-471-97844-2, John Wiley and Sons Ltd., 1999.

[16] W. Shin and D. I. Lee, "Role-Behavior Based Access Control (RBBAC) with security attribute enforcement rules", *Proc. of the 12th Workshop of Information Security and Cryptology WISC 2000*, pp. 219-232, Sep. 2000.

[17] W. Shin, D. I. Lee, and S. H. Yoon, "Role-Behavior Access Control on Mobile Agent System for Workflow Management System", *Journal of The Korean Institute of Information Security and Cryptology*, Vol. 10, No. 2, pp. 11-27, Sep. 2000.

[18] R. Sandhu and P. Samarati, "Access Control: Principles and Practice", *IEEE Communications*, Vol. 32, No. 9, September 1994

[19] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-Based Access Control Models", *IEEE Computer*, pp. 38-47, Vol. 29, No. 2, Feb. 1996.