협업을 위한 역할기반 접근통제 모델

김형찬[0], 신 욱, 이동익
광주과학기술원 정보통신공학과
{kimhc[0], sunihill, dilee}@kjist.ac.kr

Role-Based Access Control Model for Cooperative Work

Hyung Chan Kim[0], Wook Shin, Dong Ik Lee
Dept. of Information and Communications,
Kwangju Institute of Science and Technology

요 약

컴퓨터 네트워크의 보편화는 시스템 간 협업을 가능하게 하였으며, 협업 시 보안문제가 대두되기 시작하였다. 본 논문에서는 이러한 협력적인 컴퓨팅 환경에서의 접근통제 고려사항을 제시한다. 제시한 고려사항에 대하여, 역할기반 접근통제를 적용한 접근통제 모델인 RBAC-IA를 제안하며, SELinux와 NFS를 기반으로 정책 적용을 하여 예제 시스템을 구성하였다.

1. 서 론

네트워크의 비약적 발전으로 인하여, 현재 많은 컴퓨팅 단위들이 상호작용을 하고 있다. 기존의 서버-클라이언트 모델로부터, 분산 OS, 미들웨어 기반, 워크플로우 시스템, 편재 컴퓨팅 등의 기반 위에서 공동의 협업을 위해 여러 가지 어플리케이션 및 시스템들이 구축되고 있다. 이러한 시스템들은 단일 보안 관리 체계를 가질 수도 있으나, 서로 배타적인 보안 관리 체계하에 상호작용을 해야 하는 경우도 있다. 예를 들어 서로 다른 은행간의 자금 이체를 하는 경우나, 편재 컴퓨팅 환경에서 외부 도메인의 디바이스가 다른 도메인의 Context에서 상호 작용하는 경우 등, 그 예는 다양하다.

기존의 역할기반 접근통제(Role-Based Access Control, RBAC)[1,2]는 기본적으로 단일 보안 관리 체계를 전제한다. 즉, 하나의 최고 보안 관리자 혹은 단일 보안 관리 계층을 기반으로 단일 보안 관할 도메인을 관리한다. 단일 보안 관리 계층은 하나의 최고 보안 관리자와 그 하부 관리자들로 구성되는데, 이 하부 관리자들은 최고 보안 관리자의 일부 보안 관리 권한을 위임 받는다. 따라서 기존 RBAC은 협업 컴퓨팅 환경을 위해, 배타적인 도메인을 위한 보안 관리를 지원하지 못한다.

본 논문에서는 기존의 역할기반 접근통제를 확장하여, 협업을 지원할 수 있는 접근통제 모델을 소개한다. 또한 제안된 접근통제 모델의 예제로써, SELinux와 NFS를 이용하여 시스템을 구성하고 실험한 결과를 소개한다.

2. 협업 컴퓨팅 환경을 위한 접근통제 고려사항

다수의 배타적인 보안 관리 도메인들이 존재하는 협업 컴퓨팅 환경을 지원하기 위해, 다음과 같은 접근 통제 요구 사항들이 존재 한다.

2.1 접근 통제 정책 적용의 유연성 및 확장성

협업 환경에서는 다수의 도메인이 존재할 수 있다. 또한, 하나의 도메인 내에도 많은 수의 주체 및 객체들이 존재할 수 있다. 이러한 환경에서 협업에 대한 접근 통제 정책은 도메인 및 도메인 내의 시스템들의 수에 관계 없이 유연하게 구성될 수 있어야 한다. 또한 여러 가지 협업을 동시에 지원할 수 있어야 한다. 한 조직은 다른 조직과 여러 가지 수의 협업을 할 수 있기 때문이다.

2.2 도메인 경계를 넘어선 접근의 공격에 대한 안정성

협업을 하기 위해서는 다수 도메인들간에 경계를 넘어서 권한 관계를 할당해야 한다. 이 때, 도메인간의 복잡하게 얽혀진 관계를 통한 은밀한 권한 상승(Covert Promotion)[그림1]이나, 자신과 관계가 없는 도메인 내에 있는 시스템의 권한을 획득하는 하는 공격(Infiltration)[그림1]에 취약하지 않아야 한다[3].

2.3 조직 정보의 비밀성 보장

외부 도메인으로부터 내부 도메인에 대한 구성원, 실행 권한 등에 관한 접근통제 정보를 다른 도메인들로부터 보호할 수 있어야 한다. 일반적인 도메인간 접근통제를 위한 설정은 한 도메인의 사용자가 다른 도메인의 역할에 할당되거나, 한 도메인의 역할이 다른 도메인의 역할이나 권한에 대응된다. 이러한 직접적인 대응관계를 통해, 한 도메인은 다른 도메인의 역할계층 정보나 사용자 ID등의 정보를 접하게 된다. 따라서, 협업 접근 통제를 위하여 도메인간의 권한 관계를 설정하게 되면, 한 도메인은 다른 외부 도메인의 접근 통제 정보(ACI)를 접하게 된다. 이러한 정보는 사후에 악의적인 용도로 이용될 가능성이 있으므로 보호 되야 한다.

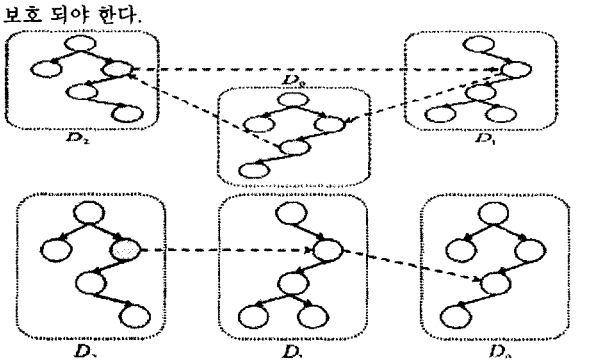

그림 1. Covert Promotion 과 Infiltration

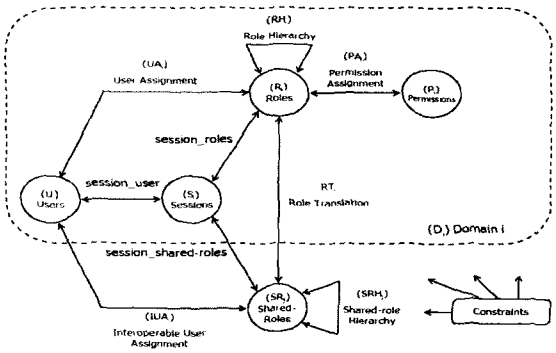

그림 2. RBAC-IA 2002 모델

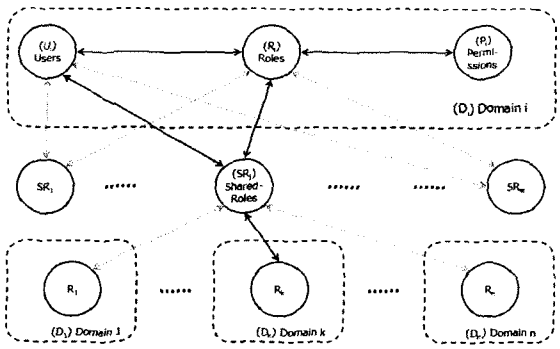

그림 3. RBAC-IA 모델에서의 도메인과 공유역할

3. RBAC-IA 2002 모델

이 장에서는 협업 환경을 지원하기 위한 역할기반 접근통제 모델인 RBAC-IA(Role-Based Access Control for Inter-domain Authorization) 모델[그림2,3]을 제안한다.

3.1 RBAC-IA Model의 주요 개념

RBAC-IA의 핵심은 공유 역할(Shared-Role)에 있다. 기존의 역할이 하나의 직무를 나타내는 것처럼, 공유 역할은 하나의 협업 작업에 대한 직무를 나타낸다. 협업에 관여하는 도메인들의 보안 관리자들은, 각 도메인의 내부 역할과 공유 역할과의 해석관계(Translation Relation)를 통하여 협업에 필요한 권한들을 구성한다. 이렇게 구성된 공유 역할은 도메인들 사이의 접근통제 시 가상 인터페이스로 작용한다. 이것은 사용자가 다른 도메인에 있는 역할에 직접 대응되지 않고, 중간 매개체로써 존재하는 공유 역할에 대응되기 때문이다. 이러한 공유 인터페이스 개념은 원천적으로 Domain-Crossing[3]을 허용하지 않기 때문에, 도메인 경계를 넘어선 불법 접근 공격에 강인하다. 또한, 이러한 간접성은 내부 조직의 접근 통제 정보를 외부에 노출 시키지 않아도 되는 장점이 있다.

3.2 Core RBAC-IA Model의 정의

핵심 모델은 RBAC-IA 모델의 기본적인 컴포넌트들을 정의한 것으로, 도메인, 역할, 공유 역할, 권한과 그 관계들을 나타낸다. 도메인 경계를 넘어선 접근을 하는 사용자는 공유 역할에 할당된다(Interoperable User Assignment). 공유 역할에 할당된 사용자는 협업을 위해, 해당 공유 역할에 할당된 권한을 사용할 수 있다. 핵심모델의 정의는 다음과 같다.

- D_i : 하나의 보안 관리 (계층)을 가진 도메인. $(1 \le i \le n)$
- U_i, R_i, P_i : 도메인 D_i 에서 정의되는 사용자, 역할 및 권한.
- SR_j : 공유 역할. $(1 \le j \le m)$
- $UA_i \subseteq U_i \times R_i$: 사용자와 역할의 할당 관계.
- $IUA_i \subseteq U_i \times SR_j$: 사용자와 공유역할의 할당 관계.
- $PA_i \subseteq P_i \times R_i$: 역할에 권한을 할당하는 관계.
- $RT_i \subseteq SR_j \times R_i$: 공유역할과 역할의 해석 관계.
- S_i : 도메인 D_i 에서의 세션.
- $assigned\_perms\_on\_role(r:R_i) \rightarrow 2^{P_i}$: 역할에 할당된 권한.

$$assigned\_perms\_on\_role(r) = \{p \in P_i \mid (p,r) \in PA_i\}$$

- $assigned\_perms\_on\_shared\_role(sr:SR_j) \rightarrow 2^{\bigcup_{k=1}^{n} P_k}$: 공유역할에 할당된 권한.

$$assigned\_perms\_on\_shared\_role(sr) = \{p \in \bigcup_{k=1}^{n} P_k \mid (sr,r) \in \bigcup_{k=1}^{n} RT_k, (p,r) \in \bigcup_{k=1}^{n} PA_k\}$$

- $sessions\_user(s:S_i) \rightarrow U_i$: 세션 s에 대한 사용자.
- $session\_roles(s:S_i) \rightarrow 2^{R_i}$: 세션 s에 대한 역할. 즉,

$$session\_roles(s_i) \subseteq \{r \in R_i \mid (session\_user(s_i), r) \in UA_i\}$$

- $session\_shared\_roles(s:S_i) \rightarrow 2^{\bigcup_{k=1}^{m} SR_k}$: 세션 s에 대한 공유역할.

$$session\_shared\_roles(s_i) \subseteq \{sr \in \bigcup_{k=1}^{m} SR_k \mid (session\_user(s_i), sr) \in \bigcup_{k=1}^{n} IUA_k\}$$

- $avail\_session\_perms(s:S_i) \rightarrow 2^{\bigcup_{k=1}^{n} P_k}$: 세션 s에 대한 사용 가능한 권한으로 다음과 같은 권한을 가진다.

$$\bigcup_{r \in session\_roles(s_i)} assigned\_perms\_on\_roles(r) \cup \bigcup_{sr \in session\_shared\_roles(s_i)} assigned\_perms\_on\_shared\_roles(sr)$$

3.3 Hierarchical Model의 정의

계층모델은 핵심모델 위에서 상속관계를 추가한다. 상속관계는 각 내부 도메인의 역할과 공유 역할에 대한 partial ordering을 정의한다. 하위(공유)역할의 권한을 상위 계층의 역할이 사용할 수 있도록 정의한다. 하지만, 공유 역할과 내부 도메인의 역할간의 partial ordering에 대한 해석관계는 아니다.

- $authorized\_users\_on\_role(r:R_i) \rightarrow 2^{U_i}$:역할 r에 인가된 사용자.

$$authorized\_users\_on\_role(r) = \{u \in U_i \mid r' \ge r, (u,r') \in UA_i\}$$

- $authorized\_users\_on\_shared\_role(sr:SR_j) \rightarrow 2^{\bigcup_{k=1}^{n} U_k}$: 공유역할 sr에 대한 인가된 사용자.

$$authorized\_users\_on\_shared\_role(sr) = \{u \in \bigcup_{k=1}^{n} U_k \mid sr' \ge sr, (u,sr') \in \bigcup_{k=1}^{n} IUA_k\}$$

- $authorized\_perms\_on\_role(r:R_i) \rightarrow 2^{P_i}$:역할 r에 인가된 권한.

$$authorized\_perms\_on\_role(r) = \{p \in P_i \mid r' \le r, (p,r') \in PA_i\}$$

- $authorized\_perms\_on\_shared\_role(sr:SR_j) \rightarrow 2^{\bigcup_{k=1}^{n} P_k}$:공유역할 sr에 인가된 권한.

$$authorized\_perms\_on\_shared\_role(sr) = \{p \in \bigcup_{k=1}^{n} P_k \mid sr' \le sr, (sr',r) \in \bigcup_{k=1}^{n} RT_k, (p,r) \in \bigcup_{k=1}^{n} PA_k\}$$

- $RH_i \subseteq R_i \times R_i : R_i$ 에 대한 상속관계(Inheritance Relation)로 다음과 같은 partial ordering 이 성립한다.

$$r_1, r_2 \in R_i,\ r_1 \ge r_2 \Rightarrow authorized\_perms\_on\_role(r_2) \subseteq authorized\_perms\_on\_role(r_1) \wedge$$

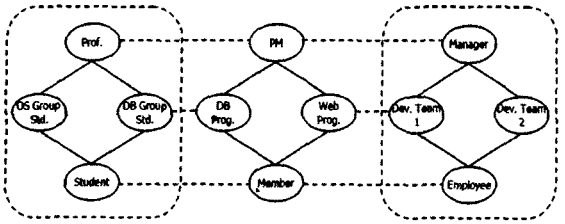

그림 4. RBAC-IA 의 예제 (산학 프로젝트)

$$authorized\_users\_on\_role(r_1) \subseteq authorized\_users\_on\_role(r_2)$$

■ $SRH_j \subseteq SR_j \times SR_j : SR_j$ 에 대한 상속관계로 다음과 같은 partial ordering 이 성립한다.

$$sr_1, sr_2 \in SR_j, sr_1 \geq sr_2 \Rightarrow authorized\_perms\_on\_shared\_role(sr_2) \subseteq authorized\_perms\_on\_shared\_role(sr_1) \wedge authorized\_users\_on\_shared\_role(sr_1) \subseteq authorized\_users\_on\_shared\_role(sr_2)$$

4. 실험

실험을 위하여 NSA(National Security Agency)에서 개발한 보안 운영체제 커널인 SELinux[4]와 분산 파일 시스템인 Network File System (NFS)[5]을 사용하였다. SELinux는 Flask구조로 RBAC과 Type Enforcement를 지원하며, 특히 실험적이지만 RBAC의 역할계층을 지원한다.

실험에서 DAC옵션을 끄기 위해 모든 파일 권한은 777로 주고, umask 역시 000으로 설정하였다. 그리고 파일 생성시 소유자 및 그룹의 PID를 두 시스템 모두 동일하게 맞추었다. 접근통제 실험을 위하여, 두 시스템을 NFS로 연결하고, 각 공유 역할에 대한 디렉토리를 만들어 해당 역할만 접근할 수 있도록 RBAC권한을 설정하였다. 하지만, SELinux가 NFS에 대한 직접적인 영속(Persistent) Labeling을 지원하지 못하기 때문에, NFS client쪽에서는 디렉토리의 깊이를 1단계 더 두어 local에서 labeling된 디렉토리의 접근통제를 받게 하였다.

RBAC-IA모델의 적용 예제로, 대학의 연구실과 회사의 개발팀이 산학 프로젝트를 하는데, 공유 Repository를 사용하는 예를 든다[그림4]. 어떤 연구실의 DB그룹(dbg_r)이 DB 프로그래밍을 담당하고, 어떤 회사의 개발 1팀(dev1_r)이 Web 프로그래밍을 맡는다. 연구실의 교수(prof_r)와 회사의 매니저(mgr_r)는 공동 프로젝트의 매니저에 해당된다. [그림 5]는 [그림4]의 SELinux 상의 역할계층 설정이다.

실험결과[그림6,7]에서 avc_enforcing 명령의 결과인 enforcing은 현재 SELinux 커널에 의한 접근통제가 활성화 되었다는 것을 알려준다. [그림6]에서, 연구실 도메인의 교수(prof_r)는 프로젝트 매니저(pm_r)의 공유 역할을 통하여, 공유 역할의 웹 프로그래밍 담당 역할(web_prg_r)의 권한을 얻는다. web_prg_r은 공유 역할의 해석관계의 의하여, 회사 도메인의 개발 1팀(dev1_r)의 역할에 해당하는 사용자들이 파일을 읽고 쓸 수 있는 권한이 있다. 따라서 교수(prof_r)는 회사 도메인의 개발1팀(dev1_r)의 사용자가 기록한 파일에 접근할 수 있다.

[그림7]에서는 반대로 개발1팀의 한 사용자가 연구실의 교수나 회사의 매니저가 해석되어 있는 프로젝트 매니저(pm_r)의 디렉토리에 접근하려 하지만, 실패하는 결과이다.

5. 결론 및 향후 계획

협업을 위하여 RBAC기반의 새로운 접근통제 모델을 제시하고, SELinux기반에서의 실험을 제시하였다. 추후에 인증서를 이용하여 공유 역할을 분배하는 RBAC-IA 접근 관리 메니저를 만들 계획이다.

```
# Role Hierachy
dominance { role mgr_r { role dev1_r { role empl_r; }
            role dev2_r { role empl_r; } } }
# Shared-Role Hierachy
dominance { role pm_r { role db_prg_r { role mem_r; }
            role web_prg_r { role mem_r; } } }
```

```
# Role Hierachy
dominance { role prof_r { role osg_r { role stud_r; }
            role dbg_r { role stud_r; } } }
# Shared-Role Hierachy
dominance { role pm_r { role db_prg_r { role mem_r; }
            role web_prg_r { role mem_r; } } }
```

그림 5. 예제 시스템들의 역할 계층 설정

```
[dilee_u@secure /]$ avc_enforcing
enforcing
[dilee_u@secure /]$ id
uid=521(dilee_u) gid=521(dilee_u) groups=521(dilee_u) context=dilee_u:prof_r:pro
f_t sid=566
[dilee_u@secure /]$ newrole -r pm_r
Authenticating dilee_u.
Password:
[dilee_u@secure /]$ id
uid=521(dilee_u) gid=521(dilee_u) groups=521(dilee_u) context=dilee_u:pm_r:pm_t
sid=593
[dilee_u@secure /]$ newrole -r web_prg_r
Authenticating dilee_u.
Password:
[dilee_u@secure /]$ id
uid=521(dilee_u) gid=521(dilee_u) groups=521(dilee_u) context=dilee_u:web_prg_r:
web_prg_t sid=621
[dilee_u@secure /]$ more /rbac_exp/shared_jobs/web_prg/web_prg/web_plan
This is the plan for web dev. (only access for web_prg_r)
[dilee_u@secure /]$
```

그림 6. prof_r 사용자의 web_prg_r권한의 파일 읽기

```
[jglee_u@air /]$ avc_enforcing
enforcing
[jglee_u@air /]$ id
uid=518(jglee_u) gid=518(jglee_u) groups=518(jglee_u) context=jglee_u:dev1_r:dev
1_t sid=543
[jglee_u@air /]$ newrole -r pm_r
Authenticating jglee_u.
Password:
jglee_u:pm_r:pm_t is not a valid context
[jglee_u@air /]$ newrole -r web_prg_r
Authenticating jglee_u.
Password:
[jglee_u@air /]$ id
uid=518(jglee_u) gid=518(jglee_u) groups=518(jglee_u) context=jglee_u:web_prg_r:
web_prg_t sid=565
[jglee_u@air /]$ cd /rbac_exp/shared_jobs/
[jglee_u@air shared_jobs]$ ls

[jglee_u@air shared_jobs]$ cd pm
bash: cd: pm: 허가 거부됨
[jglee_u@air shared_jobs]$ ls --context
ls: pm: 허가 거부됨
ls: db_prg: 허가 거부됨
ls: mem: 허가 거부됨
drwxrwxrwx  root     root     system_u:object_r:web_prg_objs_t
[jglee_u@air shared_jobs]$
```

그림 7. dev1_r 사용자의 pm_r 디렉토리 접근

6. 참고문헌

[1] D.F. Ferraiolo, J. Cugini, D.R. Kuhn "Role Based Access Control: Features and Motivations", Proc. of Annual *Computer Security Applications Conference*, IEEE Computer Sociery Press, 1995.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Younman, "Role-Based Access Control Models", IEEE Computers, Vol. 29, No. 2, pp. 38-47, Feb. 1996.

[3] Apu Kapadia, Jalal Al-Muhtadi, R. Campbell, D. Mickunas, "IRBAC 2000: Secure Interoperability Using Dynamic Role Translation", The 1st International Conference on Internet Computing, Jun. 2000.

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

[5] Linux-NFS How To, http://www.linuxdoc.org