보안운영체제를 위한 보안정책모델 및 기술언어

조주봉*, 김정순**, 김민수***, 노봉남*

*전남대학교 정보보호협동과정, **전남대학교 전산학과,

***목포대학교 정보보호학과

Security policy Model and description language for Secure Operating Systems

Joo-Bong Cho*, Jung-Sun Kim**, Minsoo Kim***, Bong-Nam Noh*

*Interdisciplinary Program of Information Security, Chonnam Univ.,
**Division of Computer Science, Chonnam Univ.,
***Division of Information Security, Mokpo Univ.

요 약

SELinux와 같은 보안 운영체제들은 정교한 접근통제 기능을 제공하고 있으나 타입(Type)과 매크로 등과 같은 정책 설정 구성요소가 많아지게 됨으로서, 운영체제에 대한 지식이 부족한 정책관리자가 정책을 설정하고 적용하기 어렵다. 이러한 정책기술 및 적용의 복잡성은 SELinux와 같은 훌륭한 보안운영체제의 범용화의 걸림돌이 되고 있다. 따라서 본 논문에서는 이러한 정책기술 및 정책 설정의 문제점을 해결하기 위해 정책관리자들이 사용하기 쉬운 보안 정책모델과 기술언어를 제안하고 SELinux의 정책과 성능을 비교하여 제안한 방법이 개선되었음을 보인다.

I. 서론

최근 인터넷 환경의 발전에 따라 전 세계에 연결된 개인 컴퓨터와 네트워크에 접근하여 정보를 이용할 수 있게 되었다. 이러한 환경에 의해 인가되지 않은 다수의 사용자에게 민감한 데이터가 노출되거나 악의적인 공격이 빈번이 발생한다. 이러한 문제를 해결하기 위해서는 보안 운영체제와 같은 새로운 보안 기술이 필요하다. 보안 운영체제는 컴퓨터 운영체제상에 내재된 보안상의 결함으로 인하여 발생 가능한 각종 해킹으로부터 시스템을 보호하기 위하여 기존의 커널 내에 보안 기능을 추가한 것이다. 보안운영체제의 접근통제를 위한 보안 구조에 대한 연구는 참조모니터를 중심으로 오래전부터 연구되어 왔으며 많은 보안 운영체제에 적용되었다. 최근에 연구된 보안 구조로는 SELinux의 Flask 보안구조, RSBAC의 GFAC등이 있다[1]. 그러나 이러한 연구에서는 정책 설정의 복잡성으로 인해 하나의 응용프로그램을 설치하여 통제하려면 그에 대한 수백여 개의 정책이 필요하다. 지금까지 연구된 보안 운영체제의 접근통제 모델은 다양한 접근통제 모델들을 표현하는데 한계가 있으며, 정책기술의 복잡성으로 인하여 시스템의 응용들에 적합한 정책을 생성하기가 어려워 일반 사용자들이 사용하기 어렵다[6]. SELinux의 경우는 섬세한 접근통제 기능을 제공하지만 그 표현의 복잡성으로 인하여 너무 많은 설정 요소들이 필요하며 표현된 정책이 복잡하여 이해하기 매우 어렵다. 설정된 엄격한 정책(strict policy)의 경우, 1200개 이상의 타입과 39개의 커널 객체 클래스 및 197개의 퍼미션 등의 설정 요소가 구성되고, 100개 이상의 매크로와 40,000개 이상의 규칙이 설정되어 있으며, 실제 리눅스 커널로 적재되어 적용되는 규칙은 360,000개 이상이 된다.

본 논문에서는 기존에 제시된 보안정책을 통한 정책 복잡성의 단점을 보완하여 효율적으로 정책을 관리하며 보안 운영체제의 복잡도 및 접근통제 정책의 관리 어려움을 감소시키기 위한 보안통제정책 모델을 제시한다.

II. 관련 연구

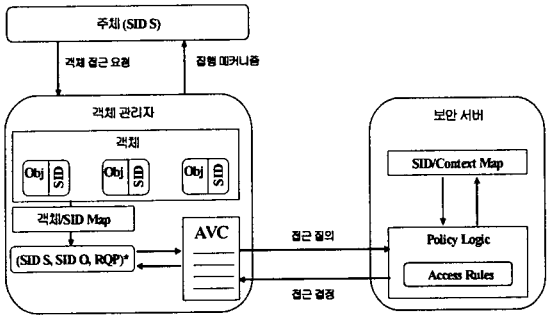
1. SELinux

SELinux는 NSA에서 유타 대학과 SCC(Secure Computing Corporation)의 도움으로 설계한 Flask 구조를 보안 구조로 사용한다.

SELinux의 보안구조는 그림1처럼 보안서버

* 본 연구는 정보통신부 대학 IT 연구센터 육성, 지원사업의 연구결과로 수행되었습니다.

(Security Server), 객체관리자(Object Manager), AVC(Access Vector Cache)로 구성되어 있다. 보안서버는 선택된 보안 정책모델에 따른 보안정책을 결정한다[8]. 여기서 보안 정책은 보안서버의 코드와 보안 데이터베이스를 말한다. 이 데이터베이스를 변경함으로서 MAC 보안 모델과 다른 RBAC과 같은 다른 보안 모델로도 변경이 가능한 유연성을 제공한다[7].



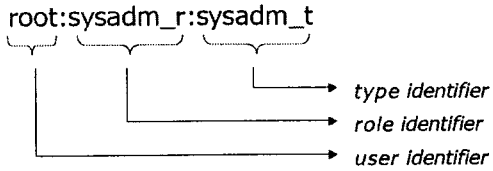
[그림 1] SELinux 구조

객체관리자는 보안서버에게 주체(클라이언트)로부터 온 요청된 객체에 대한 허용여부를 질의하는 역할을 한다. 또한 객체에 대한 SID를 매핑으로 얻어서 보안서버에 보내고 보안서버로부터 보안 정책을 받는다. 그런 후에 새로운 객체에 대한 레이블을 얻고 객체에 대한 접근 결과를 얻게 된다[3].

SELinux는 LSM의 내부 커널 객체에 관한 접근통제 방식을 이용하여 내부 커널 객체와 시스템 호출에 대한 접근통제를 수행하고, Flask구조를 리눅스로 이식하여 보안정책과 정책적용 메커니즘을 분리하였다. 또한 보안모델이 정책의 적용에 대하여 독립적인 구조이므로 TE 모델을 통한 여러 가지 정책 모델의 기술이 가능하다[4].

2. TE 모델과 보안정책

Flask에는 보안 레이블을 위한 정책과는 별도로 두 가지의 데이터 타입인 보안 콘텍스트와 보안 아이디(SID)가 있다. 보안 콘텍스트는 보안 레이블을 나타내는 가변 길이 스트링이고 SID는 보안 서버가 보안 콘텍스트에 맵핑한 정수이다. SID는 실제 콘텍스트에 대한 간단한 핸들로서 시스템을 지원하며 오직 보안 서버만이 그것을 해석할 수 있다. Flask는 객체 관리자라 불리는 구성체를 통해 실제 시스템을 바인딩 하고 이것들은 SID와 보안 콘텍스트를 불투명하게 처리한다. 그러나 보안 콘텍스트의 속성을 간섭하지는 않으며 포맷이 변경되어도 객체 관리자는 변경될 필요가 없다[2]. 보안 콘텍스트내의 보안 속성에는 Identity, Domain, Type, Role 그리고 Transition이 존재한다. Identity는 어떤 도메인에 접근 가능한지 나타내는 속성으로서 유닉스 시스템에서 말하는 UID와는 약간 다른 개념이다. setuid나 su 명령으로는 사용자 Identity를 바꾸지 않는다. Domain은 주체(프로세스 또는 프로그램)에 할당되며 도메인 안에서 동작한다. 같은 도메인에 속해 있는 프로세스들은 같은 객체들의 집합에 같은 퍼미션들을 가진다. 이는 유닉스 시스템의 UID와 유사한 개념이라고 할 수 있다. Type은 객체(파일, 디렉터리, 장치 등)에 할당되며 누가 객체에 접근할 수 있는지 결정하게 된다. 이는 도메인과 비슷한 개념으로 볼 수 있다. Role은 오직 프로세스와 연관되며 도메인을 사용할 수 있는지 결정한다. role이 도메인에 들어가는 것이 인증되지 않았다면 접근을 거부하게 된다. Transition은 프로세스가 자신의 도메인을 바꿀 때 일어나며 요청된 operation에 어떠한 보안콘텍스트를 적용할 것인지를 결정하는 것이다[5].



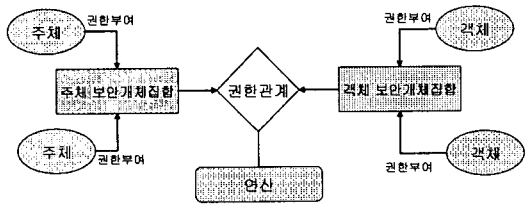
[그림 2] 주체와 객체에 할당되는 보안 컨텍스트

그림 2는 주체와 객체에 할당되는 보안 컨텍스트이며, 이는 type, role, user 식별자로 구성되어 있다[9].

SELinux의 경우는 정교한 접근통제 기능을 제공하지만 그 표현의 복잡성으로 인하여 너무 많은 설정 요소들이 필요하며 표현된 정책이 복잡하여 이해하기 매우 어렵다. 설정된 엄격한 정책(strict policy)의 경우, 1200개 이상의 타입과 39개의 커널 객체 클래스 및 197개의 퍼미션 등의 설정 요소가 구성되고, 100개 이상의 매크로와 40,000개 이상의 규칙이 설정되어 있으며, 실제 리눅스 커널로 적재되어 적용되는 규칙은 360,000개 이상이 된다.

III. 제안한 보안정책모델의 기본구조 및 구성요소

그림 3은 제안된 모델의 기본구조를 보여준다. 보안개체에 해당하는 주체와 객체, 보안개체집합인 주체 보안개체집합과 객체 보안개체집합, 권한 관계, 연산 등과 권한부여로 구성된다.



[그림 3] 제안된 모델의 기본 구조

제안된 접근통제 모델은 개체, 관계, 속성을 접근통제에 적합하게 보안개체와 보안개체 집합들의 권한관계로 표현한다. 접근 통제는 주체와 객체 그리고 주체가 객체에 행하는 행위(권한 또는 연산)로 나타낼 수 있다.

제안된 모델에서는 실세계의 객체 중에서 접근통제와 관련된 보호대상과 보호주체를 보안개체라 정의한다. 보안개체는 주체와 객체로 구분된다. 주체는 접근행위를 행하는 행위자에 해당하는 보안개체이며 객체는 접근행위의 대상이 되는 보안개체들이다. 주체의 객체에 대한 허가된 행위를 권한관계로 해석하여 보안개체 집합들 사이의 권한관계로 나타내고, 주체를 주체 보안개체 집합에 포함시키고, 객체를 객체 보안개체 집합에 포함시켜서 주체와 객체간의 관계를 주체 보안개체 집합과 객체 보안개체 집합간의 권한 관계로 표현하여 접근통제가 수행되게 한다.

1. 주체(Subject)

주체는 객체에 대한 접근을 요청하거나 허가된 행위를 수행하는 행위자에 해당하는 보안개체이다. 현실세계에서는 일반적으로 사용자가 주체가 된다. 마찬가지로 컴퓨팅환경에서 주체는 해당 시스템에서 정당한 권한을 갖는 사용자(user)가 되며 현실세계와는 다르게 프로세스(process)가 주체가 될 수 있다. 사용자와 프로세스는 밀접하게 연관되어 있으며, 유닉스 계열 시스템 환경에서는 사용자 정보를 프로세스의 자료구조에 포함하여 프로세스 기반으로 동작한다. 지금까지의 대부분의 보안 운영체제는 프로세스를 컴퓨팅환경에서 객체에 대해 실제행위를 수행하는 주체로 사용되어 왔다.

2. 객체(Object)

객체는 주체가 실제 행위를 수행하는 대상이 되는 보안개체로서 가용한 모든 자원들이 객체가 된다. 경우에 따라서는 주체도 객체의 일종으로 사용되기도 한다. 유닉스계열 시스템과 같은 컴퓨팅 환경에서의 객체들은 대부분 파일 형태로 표현하여 관리한다. 그러나 접근통제 서비스를 위해 모든 객체들을 동일한 파일형태로 관리하게 되면 객체의 특성에 따른 접근통제가 어렵고 불필요한 자료구조들이 포함되는 등 구현 및 관리가 복잡하게 되므로 대부분 시스템의 객체들을 유사한 특성을 갖는 클래스별로 분류하여 관리한다.

3. 연산(Operation)

연산은 주체가 객체에 대해 행하는 일련의 구체적이거나 논리적인 행동으로 객체에 허용된 연산들이다. 연산이 객체에 할당되었을 경우 객체는 권한이 부여된 객체가 되어 주체가 접근통제 행우를 수행할 수 있게 되며, 연산이 부여되지 않은 객체는 권한이 설정되어 있지 않으므로 주체가 접근통제 행위를 수행할 수 없다. 본 모델에서는 주체 보안개체 집합과 객체 보안개체 집합간의 권한관계에 연산을 부여한다. 일반적으로 사용되는 연산은 시스템 콜을 사용하여 정의하는데 예를 들면 파일 객체에 허용된 오퍼레이션들은 읽기(read), 쓰기(write) 등으로 정의된다.

4. 보안개체집합

보안개체 집합은 어떤 조직의 구성요소들에게 부여될 수 있는 하나의 직책이나 직무를 위한 소속부서 및 직책에 적합한 업무 기능을 표현하기 위해 정의한 추상적인 개념이다. 보안개체 집합은 접근통제의 구성요소인 주체와 객체들의 집합으로 표현될 수 있다. 주체들의 집합을 주체보안개체 집합, 객체들의 집합을 객체 보안개체 집합이라 정의한다. 보안개체 집합들 사이에는 상속관계가 존재하며, 보안개체 집합들이 모여서 새로운 보안개체 집합을 구성할 수 있다.

5. 보안개체집합 부여

보안개체집합 부여는 주체 보안개체 집합과 객체 보안개체 집합간의 설정된 권한관계를 실제 접근통제행위의 행위자인 주체와 접근통제 행위의 대상인 객체에게 각각 할당한다. 주체 보안개체 집합을 주체에 부여함으로써 해당 주체는 주체 보안개체 집합과 권한관계가 설정된 객체 보안개체 집합을 부여받은 객체에 대해 허용된 연산을 수행한다.

6. 권한관계

보안개체 집합과 보안개체 집합간의 권한 관계는 다대다 관계이며, 권한관계 설정을 통해 주체 보안개체 집합에 속한 주체들이 객체 보안개체 집합에 속한 객체에 대한 권한을 부여받게 된다. 조직의 업무수행에 필요한 사용자와 객체의 역할들을 보안개체 집합들로 정의하고, 주체 보안개체 집합과 객체 보안개체 집합을 각각 사용자와 해당 객체에 부여하여 객체에 대해 권한이 필요한 경우 해당 주체 보안개체 집합과 해당 객체 보안개체 집합의 권한 관계를 설정하여 사용자가 해당 객체데 대한 연산을 수행하도록 한다. 이러한 방법은 사용자와 정보객체의 수가 많은 환경에서 접근통제를 용이하게 수행할 수 있는 장점을 제공한다.

IV. 정책 기술언어

제안한 모델의 구성요소인 주체, 객체 및 보안개체 집합 등의 기본 문법을 정의한다. 보안개체 집합을 부여 받지 못한 주체와 객체는 기본적으로 모든 권한이 거부된 상태이므로 접근이 허용되지 않는다. 또한, 주체 보안개체 집합과 객체 보안개체 집합을 부여 받았더라도 이들 보안개체 집합간의 권한 관계가 설정되지 않았을 경우 해당 주체와 객체는 모든 접근 행위가 거부된다.

1. 보안개체 집합의 선언

주체 보안개체 집합과 객체 보안개체 집합에 해당하는 보안개체 집합을 Domain 예약어를 이용하여 선언한다.

```
○ 보안개체 집합 선언 문법
  Domain Domain_Name

○ 사용 예
  Domain Apache_s
  Domain Apache_o
```

2. 보안개체집합 부여

주체나 객체에 보안개체 집합을 grant 예약어를 사용하여 부여한다. 주체 보안개체 집합을 부여할 때는 subject 예약어를 사용하고 객체 보안개체 집합을 부여할 때는 object 예약어를 사용한다.

```
○ 보안개체 집합 선언 문법
  Grant [subject|object|] [user|object]
  {
        Domain_list
  }

○ 사용 예
  Grant subject Apache {
        apache_s
  };
  Grant object /usr/sbin/httpd {
        apache_o
  };
```

3. 권한관계 설정

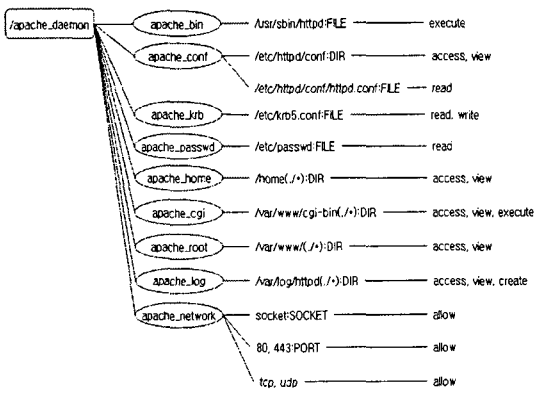
authorize 예약어를 사용하여 주체 보안개체 집합과 객체 보안개체 집합의 권한관계를 설정한다.

```
○ 보안개체 집합 선언 문법
  authorize S O { Type: operations };

○ 사용 예
  authorize apache_s apache_o {
     FILE : read, write, execute
     DIR  : write
  };
```

4. Apache 정책의 기술 예제

Apache는 기본적으로 웹 서비스를 제공하는 서버 어플리케이션으로 다양한 모듈들을 지원하기 때문에 정책을 모두 표현하는 규칙을 만들기는 매우 어렵다.



[그림 4] Apache가 사용하는 객체들과 연산들

그림 4은 Apache가 사용 중인 객체와 연산관계를 나타내고 있다. 이 그림을 토대로 접근통제 정책을 기술해 보면 다음과 같다.

```
// 보안 개체 집합 선언
domain apache_s_manager;
domain apache_o_bin;
doamin apache_o_conf, apache_o_krb;
doamin apache_o_passwd, apache_o_home;
// 보안 개체 부여
grant subject  webmaster {apache_s_manager};
grant object  /usr/sbin/httpd {apache_o_bin};
// 권한 관계 설정
authorize apache_s_manager apache_o_bin
 { FILE : {execute} };
authorize apache_s_manager apache_o_conf
 { FILE : {read} };
```

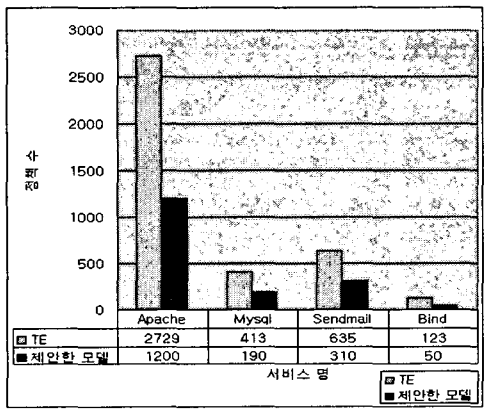
V. 실험

표 1는 SELinux의 각 서비스별 type의 개수와 rule의 개수를 "targeted policy"의 컴파일을 통해 나타낸 것이다.

[표 1] SELinux의 각 서비스별 정책 수

	apache	mysql	named	sendmail
types	30	8	13	4
rules	2729	413	635	123

위 개수는 190개의 type과 1712개의 rule의 기본 정책 type, rule을 제외한 수이다. 가장 많은 정책을 가지는 apache를 보면 rule의 수가 2729개를 사용하고 있는 것을 알 수 있다. 이는 상당히 많은 정책을 사용하고 있음을 알려준다.



[그림 5] TE 모델과 제안한 모델의 비교

그림 5는 TE와 제안한 모델의 정책 복잡성을 비교한 차트이다. TE의 경우 상당수의 정책이 필요하며 제안한 모델의 경우 최소 정책만으로도 접근통제를 할 수 있었다. SELinux의 경우 "targeted policy"이며, 이는 매크로를 포함한 결과의 수치이다. 제안한 모델의 경우도 마찬가지로 최소한의 접근통제 정책만을 표현한 수치이다.

이처럼 정책 수가 상당한 차이를 보이는 것은 SELinux의 경우 많은 오퍼레이션과 많은 구문을 사용함으로 인해 제안한 모델과 차이가 난다. 이는 보안 관리자의 정책 관리를 힘들게 하며, 실제 적용하기가 힘들다.

VI. 결론

본 논문에서는 보안 운영체제의 복잡도 및 접근통제 정책의 관리 어려움을 감소시키고 개선하기 위한 보안 정책모델을 제안하였다. 위의 실험에서 실험한 결과를 통해 제안한 접근 통제 모델이 Apache 43%, Mysql 46%, Sendmail 48%, Bind 40% 줄어들어 정책의 복잡성을 획기적으로 개선하였음을 확인하였다.

향후에는 다른 정책 모델과의 유연성을 위해 MAC, RBAC을 수용할 수 있는 변환 모듈과 정책 설정의 어려움을 없애고자 정책 자동 설정 모듈에 대한 연구가 진행되어야 한다.

[참고문헌]

[1] "유연한 접근통제를 제공하는 보안 운영체제를 위한 접근통제보안구조", 정보보호학회 논문지
[2] Implementing SELinux as a Linux Security Module, Technical report, NAI Labs, May 2002.
[3] The Flask Security Architecture: System Support for Diverse Security Policies
[4] SELinux, http://www.nsa.gov/selinux/index.cfm
[5] http://www.nsa.gov/selinux/
[6] "리눅스 보안 운영체제를 위한 접근통제 프레임워크", 한국정보과학회 학회지
[7] A Security Architecture for adapting multiple Access Control Models to Operating Systems
[8] Configuring the SELinux Policy, Technical report
[9] Implementing SELinux as a Linux Security Module, Technical report