SELinux 정책템플릿 기술언어 설계

최민호*, 김정순**, 김민수***, 노봉남**
*전남대학교 정보보호협동과정
**전남대학교 전자컴퓨터정보통신공학부
***목포대학교 정보보호학과
e-mail:lunatine@lsrc.jnu.ac.kr

The Design of SELinux Policy Template Description Language

Min-ho Choi*, Jung-Sun Kim, Minsoo Kim, Bong-Nam Noh**
*Interdisciplinary Program of Information Security, Chonnam National University
**Division of Electronics Computer&Information Engineering, Chonnam National University
***Division of Infromation Security, Mokpo National University

요 약

현재 보안 운영체제로서의 SELinux를 이용하기 위해서는 전문적인 지식이 필요하며 정책을 설정함에 있어서도 시스템 콜 레벨의 설정이 필요하다는 문제점이 존재한다. 이러한 문제점을 개선하기위해 정책 정의를 SELinux의 TE모델보다 간략하게 표현 및 정의하고 사용자의 접근성을 높일 수 있는 기술언어를 본 연구에서 설계하였으며 이를 통해 보안 운영체제의 활용도를 높이도록 하였다.

1. 서론

인터넷이 발달하고 보안에 대한 중요성이 강조되면서 국내·외에서 운영체제 보안에 대한 개발 움직임이 활발하게 진행되고 있다. 이중에서 대표적인 국외 공개 보안 운영체제로는 미국 국가안전보장국(NSA: National Security Agency)의 SELinux, 독일 함부르크 대학의 Amon Ott가 개발한 RSBAC (Rule-Set Based Access Control) 등이 있다[1].

이중 SELinux(Security-Enhanced Linux)는 유타(Utah)대학의 Flas1)k(Flux Advanced Security Kernel) 구조를 리눅스에 적용하여 개발한 보안 운영체제로 TE(Type Enforcement), 역할기반 접근통제(RBAC: Role Based Access Control), 다중등급보안(MLS: Multi-Level Security) 등의 다양한 접근통제 정책을 집행하는 구조를 제공하고, 다양한 시스템내의 자원에 대한 접근통제를 수행한다. 또한, 구조적으로는 정책 결정 모듈과 정책집행 모듈을 분리시킴으로써 보안 정책에 유연성을 제공한다. SELinux에서는 리눅스에서 제공되는 시스템호출을 기반으로 규칙을 서술하고 사용자, 프로세스, 역할 등에 따라 규칙을 세분화하여 사용한다[3][6][7].

SELinux는 타입(type)이라는 이름으로 주체와 객체의 관계를 표현하고 타입의 전이를 통하여 관계가 변경되도록 되어 있어 오퍼레이션의 종류가 많고 복잡하여 전문지식을 갖고 있지 않은 일반 사용자가 이용하기에는 어려움이 많다. 또한, 규칙들 간의 복잡한 연관성 때문에 사용자가 쉽게 규칙을 변경하기 어렵게 되어 있으며, 새로운 소프트웨어나 프로그램이 추가되었을 때 그 것의 규칙을 시스템호출 수준에서 재 정의해야 하는 어려움이 존재한다.

현재 국내에서 이와 같은 어려움을 해결하기 위해서 정책템플릿(SELT: SELinux Template)이라는 보안환경 설정도구 연구가 진행되고 있다. 이 SELT가 사용자에게 SELinux가 제공하는 TE보다 쉽고 접근성이 높은 환경을 제공하기 위해서는 규칙을 정의하는 기술언어에 대한 연구가 필요하다[9].

본 논문에서는 TE보다 접근성이 높은 기술언어의 설계에 대해서 소개하고 TE와 복잡성 및 접근성에 대한 비교실험을 수행한다.

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

2. 정책템플릿 기술언어의 요건

정책템플릿 기술언어의 요건은 첫 번째로 정책템플릿 기술언어는 문법이 쉽고 단순해야 하며 사용자가 설정하고자 하는 보안정책을 명확하게 표현할 수 있어야 한다. 두 번째로 기술된 보안정책의 의미를 쉽게 파악할 수 있는 가독성을 갖추고 있어야 한다. 세 번째로 SELinux의 정책기술을 위한 타입간의 전이, 오퍼레이션 및 예외 상황과 같은 복잡한 설정과 te, fc 등의 파일에 분산되어 저장된 SELinux 정책 정보를 하나의 파일에서 관련성 있는 정보들의 흐름을 파악할 수 있도록 표현할 수 있어야 한다.

또한, 사용자가 특정 주체(Subject)의 객체(Object)에 대한 접근권한을 설정하는데 있어서 그 의미를 사용자가 생각하는 대로 기술할 수 있도록 최대한 간략한 형태로 구성되어야 한다. 그리고 간략한 형태뿐만 아니라 정책템플릿 기술언어는 주체와 객체에 대한 선언과 접근권한에 대한 정의, 주체의 전이에 대한 정보를 표현할 수 있어야 한다.

3. 정책템플릿 기술언어 구문

정책템플릿 기술언어는 SELinux의 TE 모델이 정의하는 보안 정책을 모두 수용할 수 있어야 하므로 크게 다음과 같은 구조로 구성된다.

- 정책템플릿 정의 구문
- 주체 정의 구문
- 주체의 도메인 전이 정의 구문
- 객체 정의 구문
- 객체에 대한 주체의 접근 권한

3.1 정책템플릿 정의 구문

정책템플릿을 선언하는 구문이다. 하나의 정책템플릿은 주체에 대한 권한설정의 그룹화 의미를 갖는다.

```
Template template_name {
   SUBJECT TRANSITION OBJECT PERMISSION
}
```

(그림 1) 정책템플릿 정의 구문

그림 1에서 Template는 하나의 블록구문 형태로 정의 되고 template_name은 현재 정책템플릿에 대한 고유 식별자 이름으로 사용된다. 정책 템플릿은 주체(SUBJECT), 객체(OBJECT)에 대한 정의 섹션과 전이관계(TRANSITION) 및 권한설정(PERMISSION) 섹션을 포함한다.

3.2 주체 정의 구문

```
Subject:
        name    subject_name
        type    domain_type
```

(그림 2) 주체 정의 구문

그림 2는 주체가 되는 사용자를 정의하는 구문이다. name 키워드는 주체의 이름을 정의하며 type 키워드는 미리 정의된 프로그램의 유형으로 운영체제의 서버 어플리케이션(server), 네트워크 어플리케이션 (network), 일반 어플리케이션(common)등으로 정의한다. 예를 들어 Apache 웹서버 어플리케이션의 경우 사용자는 httpd, type은 common, server, network가 된다.

3.3 전이 정의 구문

```
Transition:
        domain { object_type …… }
```

(그림 3) 전이 정의 구문

그림 3은 주체 도메인으로의 전이 관계를 정의하는 구문이다. domain에는 주체의 도메인으로 전이되고자하는 도메인을 기술하고 주체 도메인으로 전이하기 위한 실행 객체(object_type)을 선언한다.

예를 들어 Apache 웹서버 어플리케이션의 경우 init 스크립트를 통해서 Apache 웹서버의 도메인으로 전이되기 위해서는 domain에 initrc, object_type에는 실행 파일 객체 이름 httpd_exec(객체 정의 구문에서 실제 실행 파일을 정의한 별칭이다)가 된다.

3.4 객체 정의 구문

정책템플릿에서 정책을 정의할 때 객체가 되는 대상을 정의한다. 각각의 대상은 객체 타입으로 그 형태를 지정하며 대상에 대한 경로 및 명칭을 선언하게 된다.

```
Object:
        object_name [option] { path_name | value }
        ……
```

(그림 4) 객체 정의 구문

그림 4에서 object_name에는 객체의 별칭(Alias)을 지정하는데 이는 object_name이 단일 파일, 디렉터리, 복수개의 파일 및 디렉터리 그리고 네트워크 값과 같이 여러 객체를 그룹화 하여 표현하기 때문이다. object_name으로 선언된 객체의 실제 경로 및 값에 대해서는 path_name, value를 통해서 정의하고 TE의 fc(file context)에 해당하는 정보를 정책템플릿의 객체 명칭으로 강제 레이블(label)할 것인지를 옵션(option)에서 지정하도록 한다. 예를 들어 Apache 웹서버 어플리케이션의 실행파일에 대한 객체를 선언할 경우 별칭이름은 httpd_exec가 되고 Apache 웹서버 고유의 파일이기 때문에 옵션에 'in'을 설정하여 강제 레이블을 표현하며 path_name에 /usr/sbin/httpd와 같은 서버 바이너리의 경로를 표기하게 된다.

3.5 권한 정의 구문

정책템플릿에 선언한 주체와 객체 사이의 권한을 설정하는 구문이다. 각각의 권한은 주체가 객체에 대해서 어떠한 권한을 줄 것인가의 형태로 설정하게 된다.

```
Permission:
        object_name object_type { operations …… }
```

(그림 5) 객체 정의 구문

그림 5에서 object_type은 객체 정의 구문에서 선언한 객체의 형태를 정의하고 이 객체에 대한 operations을 허가하도록 정의하게 된다. operations는 실제 TE에서 사용하는 권한을 사용자가 알기 쉬운 형태로 read, write, execute 형태로 제공되며 네트워크 및 특수 객체에 대해서는 추가적인 액션을 제공한다. 예를 들어 Apache 웹서버 어플리케이션의 httpd_exec 객체의 권한을 설정할 경우 object_name은 httpd_exec, object_type은 file, operations에는 read, write, execute 등이 주어진다.

4. 정책템플릿 기술언어 사용 및 TE와의 비교

앞서 정의한 기술언어를 이용하여 실제 정책을 정의하고 작성된 정책을 TE와 비교하여 편의성과 정책의 복잡성 개선부분을 확인해 본다.

4.1 정책템플릿 기술언어의 실제 사용 예

정책템플릿 기술언어의 정책의 예로써 서버 어플리케이션인 Apache 웹서버의 정책을 정의하면 그림 6과 같다.

Apache 웹서버에 대한 정책템플릿 명칭은 httpd_selt로 정의하였으며 주체의 이름을 httpd로 정의하였다. Apache 웹서버의 경우는 서버 어플리케이션이며 네트워크 서비스를 수행하는 어플리케이션이기 때문에 Subject 섹션에 daemon과 network를 정의하였다.

```
Template httpd_selt {
  Subject:
    name        httpd
    type        common ,daemon ,network

  Transition:
    unconfined    { httpd_exec }
    unconfined    { httpd_run_script }
    initrc        { httpd_exec }
    initrc        { httpd_run_script }

  Object:
    httpd_exec          in  { /usr/sbin/httpd, /usr/sbin/httpd.worker }
    httpd_run_script    in  { /etc/init.d/httpd }
    httpd_conf          in  { /etc/httpd/conf.d(/.*)?,
                              /etc/httpd/conf/httpd.conf }
    httpd_ssl               { /etc/pki/tls(/.*)? }
    httpd_krb5              { /etc/krb5.conf }
    httpd_run_general       { /var/run(/.*)? }
    httpd_log           in  { /etc/httpd/logs/ }
    httpd_module        in  { /etc/httpd/modules(/.*)? }
    httpd_run               { /var/run/httpd.pid(/.*)?  }
    httpd_build             { /usr/lib/httpd/build(/.*)?  }
    httpd_htdocs            { /var/www(/.*)?  }
    httpd_cgi               { /var/www/cgi-bin(/.*)?  }
    httpd_user_home         { /home(/.*)? }
    httpd_port              { 80, 443 }

  Permission:
    httpd_exec          file  { all, read, execute }
    httpd_run_script    file  { read, execute }
    httpd_conf          dir   { read }
    httpd_conf          file  { read }
    httpd_ssl           dir   { access, view }
    httpd_ssl           file  { read }
    httpd_krb5          file  { all }
    httpd_run_general   dir   { access, view }
    httpd_run_general   file  { read }
    httpd_module        dir   { access, view }
    httpd_module        file  { read, execute }
    httpd_run           dir   { access, create, remove, view  }
    httpd_run           file  { read, write, remove }
    httpd_build         dir   { access, view }
    httpd_build         file  { read  }
    httpd_htdocs        dir   { create, remove, access, view  }
    httpd_htdocs        file  { read, write, remove  }
    httpd_cgi           file  { read, execute  }
    httpd_user_home     dir   { access, view  }
    httpd_port          tcp_socket  { server, client }
    httpd_port          udp_socket  { server, client }
    httpd_port          raw_socket  { allow }
}
```

(그림 6) Apache 웹서버의 정책템플릿

Apache 웹서버의 도메인 전이는 TE의 기본상태인 unconfined나 시스템의 init스크립트인 initrc 도메인에서 서버 데몬 파일을 수행하여 httpd로 전이

가 발생하는 것으로 정의하였고 Apache 웹서버가 접근하는 객체에 대해서 Object 섹션에 그룹화 하여 정의하였다.

4.2 정책템플릿 기술언어와 TE의 비교

이 장에서는 기존의 TE 모델을 통해 작성된 SELinux 정책인 targeted 정책과 정책템플릿 기술언어를 통해 작성된 정책을 비교하여 정책템플릿 기술언어를 통해 SELinux의 정책 복잡성의 감소 결과를 한다. 비교 대상이 된 정책은 Fedora Core 5의 targeted 정책이며, 규칙의 수는 매크로를 포함한다.

targeted 정책과 정책템플릿 기술언어를 비교한 결과는 표 1과 같으며 이를 통해서 TE 모델을 통해 작성된 정책에 비해 정책템플릿 기술언어를 통해 작성된 정책에서의 타입수와 규칙의 수가 타입은 24%, 규칙은 91% 정도 감소하였음을 확인할 수 있다.

<표 1> TE와의 정책복잡성 비교

구 분	정책템플릿 기술언어		Type Enforcement	
	타입 수	규칙 수	타입 수	규칙 수
Apache	14	22	30	285
Mysql	7	8	8	78
Named	9	11	13	145
Sendmail	13	15	4	80
전 체	43(-24%)	56(-91%)	55	588

또한, TE 모델을 기반으로 한 SELinux의 정책은 te 파일과 fc 파일이 다른 디렉터리에 따로 존재하여 사용자가 보안 정책을 한눈에 보고 이해하기가 쉽지 않다. 이에 반하여 정책템플릿은 단일 파일을 통해서 정의된다. 그리고 사용자가 정책을 정의할 때 주체와 객체의 관계에 기초하여 정의하기 때문에 각각의 주체, 객체, 전이, 권한의 섹션으로 블록화 하는 문법을 제공하는 정책템플릿 기술언어가 한눈에 이해하기도 쉬우며 수정 또한 용이하다.

5. 결론

보안 운영체제로서의 SELinux는 앞으로 보다 많은 사용자가 사용하게 될 것이다. 그러나 현재까지의 SELinux는 전문 지식을 갖지 않은 일반 사용자들에게는 사용하고 싶어도 사용할 수 없는 상태로 되어 있다.

본 연구에서 설계된 정책템플릿 기술언어는 태스크, 도메인, 관계 등의 구성요소와 연산자 및 제약사항을 정의한 모델을 바탕으로 기술언어를 정의하고 문법을 기술하였다. 본 연구를 통해서 SELinux의 가장 큰 문제점으로 지적되고 있는 보안정책 설정의 복잡성에 대해서 TE 모델 기반의 보안 정책과 비교하여 규칙 수를 평균 90% 정도 감소시켰다. 또한, 정책을 섹션화하여 가독성을 높였으며 현재까지 사용함에 있어서 복잡성과 난해함을 갖고 있던 저수준의 보안환경 설정 규칙을 간소화 하였다. 보안 운영체제의 정책 설정에 있어서 복잡성을 줄이고 접근성을 높임으로써 보안 운영체제의 활용도 향상을 기대할 수 있게 되었다. 향후에는 본 연구를 통해 작성된 정책템플릿 기술언어가 신뢰할 수 있는 보안정책 기술언어가 되기 위하여 시스템에 적용됨에 있어서 정확성과 신뢰성에 대한 검증이 연구 되어야 할 것이다.

참고문헌

[1] A. Ott, "The Rule Set Based Access Control (RSBAC) Linux Kernel Security Extension," 8th Int. Linux Kongress, Enschede 2001.
[2] James M., Stephen S., Greg K.H., "Linux Security Modules: General Security Support for the Linux Kernel," USENIX Security Symposium, 2002.
[3] Lee B., Daniel F.S., David L.S., Kenneth M.W., and Sheila A.H., "A Domain and Type Enforcement UNIX Prototype," fifth USENIX Unix Security Symposium, 1995.
[4] Peter L. and Stephen S., "Integrating Flexible Support for Security Policies into the Linux Operating System," NSA Technical Report, February 2001.
[5] Stephen S., Configuring the SELinux Policy, NAI Labs Rep. 02-007, 2003. http://www.nsa.gov/selinux/policy2-abs.html
[6] UNICOS Multilevel Security(MLS) Features User's Guide, SG-21111 10.0
[7] Flask, http://www.csutah.edu/flux/flask
[8] SELinux, http://www.nsa.gov/selinux/
[9] SELT, http://selt.jnu.ac.kr