SELinux 정책 복잡성 개선을 위한 보안정책 템플릿

정종민*, 김정순**, 김민수***, 정성인****, 노봉남*

*전남대학교 정보보호협동과정, **전남대학교 전산학과,

***목포대학교 정보보호학과, ****한국 전자통신 연구원

Security Policy Template to Reduce the Complexity of SELinux Security Policy

Jong-Min Jung*, Jung-Sun Kim**, Minsoo Kim***, Seong-in Jung****, Bong-Nam Noh*

*Interdisciplinary Program of Information Security, Chonnam National Univ., **Division of Computer Science, Chonnam Univ., ***Division of Information Security, Mokpo Univ., ****Electronics and Telecommunications Research Institute.

요 약

보안을 위협하는 요소들에 대하여 기존의 보안기술들은 응용계층 기술의 한계를 드러내고 있다. 이를 극복하기 위한 방법으로 보안 운영체제에 대한 연구가 활발히 진행되고 있지만, 보안정책 설정의 복잡성 때문에 일반 사용자들이 보안정책을 설정하여 적용하기가 어렵다. 본 논문에서는 대표적인 보안 운영체제인 SELinux의 보안모델과 이와 관련된 연구들을 살펴보고, SELinux의 정책 복잡성 개선을 위한 SELinux 보안정책 템플릿을 제안한다.

I. 서론

인터넷이 환경의 발전은 정보이용의 편의성을 제공함과 동시에 악의적인 공격에 언제나 노출되어 있다. 이에 따라서, 안전한 정보의 공유 및 이용을 위해 암호화, 방화벽 및 침입탐지시스템 등의 응용계층 보안기술들이 개발되어 네트워크나 서버의 정보를 보호하고 있다. 하지만 이러한 응용계층의 보안기술들은 자체적인 취약점을 가지고 있을 뿐 아니라, 내부자의 침입, 권한의 오남용, 시스템 해킹을 통한 공격에 대응하기가 어렵다. 이러한 문제점을 해결하고 신뢰할 수 있는 전산환경(TCB: Trusted Computing Base)을 구현하기 위해 보안 운영체제에 대한 연구가 활발히 진행 되고 있으며, 대표적인 보안 운영체제로 SELinux(Security Enhanced Linux)를 들 수 있다.[1]

SELinux는 NSA에서 Flask(Flux Advanced Security Kernel) 구조를 리눅스에 적용하여 개발한 보안 운영체제로 TE(Type Enforcement), 역할기반 접근통제(Role Based Access Control), 다중등급보안(MLS: Multi-Level Security) 등의 다양한 접근통제 정책을 집행하는 구조를 제공한다. 그리고 파일 및 장치파일 뿐 아니라 프로세스, 시그널, 메모리 등의 다양한 시스템 내의 자원에 대한 접근통제를 수행한다. 또한, 최소권한 할당을 통해 피해 범위를 최소화하고 악의 적인 코드의 실행을 방지한다. 구조적으로는 정책결정과 정책집행 모듈을 분리시킴으로써 보안정책에 유연성을 제공한다[2],[9].

이와 같은 SELinux의 특징은 보다 섬세한 접근통제를 가능하게 하지만, 보안정책의 복잡성을 증가 시킨다. 그래서 일반 사용자들이 보안정책을 원하는 목적에 맞게 설정하는 일이 매우 어렵기 때문에 본 논문에서는 SELinux의 복잡한 보안정책이 가지는 단점과 이를 개선하기 위한 관련 연구를 살펴보고 SELinux의 보안정책의 복잡성을 개선하기 위한 SELinux 보안정책 템플릿(SELT: SELinux Template)을 제시한다.

본 논문의 구성은 2장에서 SELinux의 TE 모델과 보안정책의 복잡성에 대해 기술하고, 3장에서는 이를 개선하기 위한 보안환경 설정도구들에 대해서 살펴본다. 4장에서 SELinux의 보안정책 복잡성 개선을 위한 SELT를 제시하고, 5장에서 연구결과를 살펴보며, 6장에서는 결론과 향후 연구내용에 대해서 기술한다.

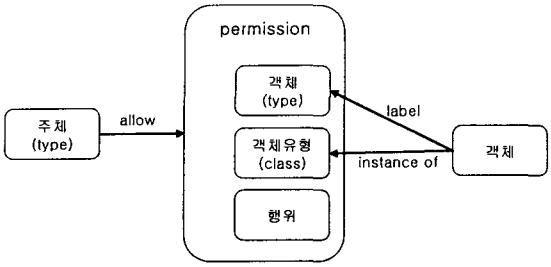
II. SELinux의 보안 모델

1. TE 모델과 보안정책

SELinux의 보안정책은 시스템 내의 모든 프로세스가 제한된 도메인에서만 실행되도록 샌드박스 개념을 구현한 것이다[6]. 즉 각 프로세스는 자신의 도메인에서 자신에게 주어진 최소한의 권한 내에서만 동작하고 다른 프로세스 도메인을 간섭할 수 없다[1],[3]. 이러한 보안정책 설정을 위해 SELinux의 보안

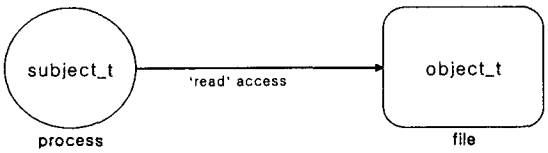
* 본 연구는 정보통신부 대학 IT 연구센터 육성, 지원사업의 연구결과로 수행되었습니다.

정책은 TE 모델로 구성되어 있으며, 그림 1은 TE 모델의 구조를 나타낸다.



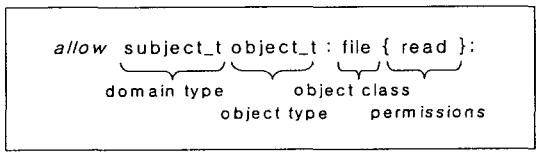
[그림 1] TE 모델의 구조

SELinux에서 시스템내의 모든 주체와 객체들은 동일한 보안 속성을 가진 그룹을 의미하는 '타입'을 할당 받는데 특별히 주체에 부여되는 '타입'을 '도메인'이라 한다. 그리고 특정한 '타입'을 갖는 주체가 객체를 생성하거나 특정한 '타입'을 갖는 객체를 실행하면 정책에 정의된 타입으로 전이가 발생한다[1].



[그림 2] 주체와 객체와 권한 관계

그림 2는 'subject_t'를 도메인으로 갖는 프로세스에게 'object_t'를 타입으로 갖는 파일에 'read'라는 권한을 허용함을 도식화 한 것이다.



[그림 3] TE 모델 허용규칙 표현 예

그림 3은 그림 2를 TE 모델의 문법으로 표현한 것이다. 'object class' 는 SELinux에 정의된 55종류의 객체 유형을 의미한다. 'permissions'는 권한을 의미하는데 SELinux의 경우 280여개의 리눅스 시스템 호출을 객체 유형에 따라 연관성 있는 그룹으로 묶어 정의 하고 있다[4].

2. SELinux 보안정책의 복잡성

살펴본 바와 같이 TE 모델은 타입이라는 이름으로 주체와 객체의 관계를 표현한다. 그리고 타입전이를 통하여 관계가 변경되기 때문에 정책을 통해 실제 주체와 객체의 관계를 이해하기 어렵다. 또한 주체와 객체사이의 관계를 정의 하는데 다양한 권한 집합들이 사용되기 때문에 정책의 복잡성은 증가된다.

표 1은 페도라 코어4 배포판 기준의 SELinux 보안정책인, 'strict policy'와 'targeted policy'에 정의된 '타입'과 규칙의 수를 보여준다.

[표 1] SELinux 주요 정책의 규칙

정책 이름	타입 수(개)	규칙 수(개)
strict policy	1341	345,260
targeted policy	764	180,131

'strict policy'의 경우 보안정책 내부에 1341개의 '타입'들이 존재하며, 타입들 간의 관계를 정의한 규칙이 456,260개 존재한다는 것을 의미한다.

III. SELinux 보안환경 설정도구

SELinux의 보안정책의 설정 및 관리의 어려움을 극복하기 위해 개발된 도구로는 Tresys Technology의 'SETools', Hitachi Software의 'SELinux Policy Editor(SEEdit)'와 MITRE Corporation의 'SLAT'과 'polgen'등이 대표적이다[5],[7],[8],[10].

[표 2] SELinux 보안환경 설정도구 비교 분석

종 류	특 징
SETools	· GPL을 따르는 공개소스 배포 · 보안정책 분석 도구 모음 · 정책 구성요소 검색기능 · 보고서 생성기능 · X윈도우 환경 GUI 인터페이스 제공
SEEdit	· GPL을 따르는 공개소스 배포 · 웹 기반 GUI 도구제공 · 매개언어 사용 · 규칙 간소화
SLAT/ polgen	· GPL을 따르는 공개소스 배포 · 정보흐름 검사기능 · 정책 자동 생성기능

표 2는 대표적인 SELinux 보안환경 설정도구들에 대한 특징을 보여준다. SETools는 정책에 대한 세부적인 분석과 디버깅을 할 수 있도록 해주는 도구들의 모음으로 SELinux의 정책을 분석하고 보고서를 만들어 준다. SEEdit는 Webmin과 연동하여 웹 기반의 보안정책 설정 GUI를 제공하며, 자체적으로 매개언어를 사용하여 SELinux의 다양한 권한을 r, w, c, s로 매핑 하여 규칙을 간소화 시켰다. SLAT은 정보흐름을 검사하여 보안정책을 분석하고, polgen은 SELinux의 보안정책을 자동으로 생성하여 준다.

하지만 언급된 보안환경 설정도구들은 SELinux를 이용하기 편하도록 사용자 인터페이스를 제공하는 것에 그치고 있으며, 이 들을 목적에 맞게 사용하기 위해서는 SELinux의 보안정책에 대한 이해를 필요

로 한다. 즉, 편의성은 향상 되었으나 보안정책 설정의 복잡성은 여전히 존재한다는 것이다.

IV. SELT(SELinux Template)

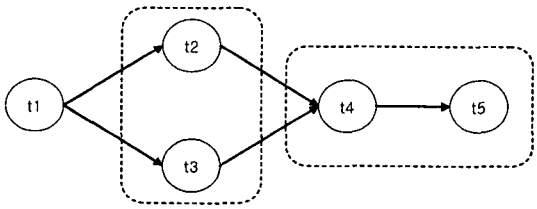
이 장에서는 2장에서 살펴본 보안정책 복잡성에 대한 개선 방안을 제시하고, 3장의 보안환경 설정도구들의 부족한 점들을 보완하여 전문지식이 없는 사용자들이 SELinux를 보다 쉽게 이용할 수 있도록 개발된 SELT를 제안한다.

1. 보안정책 복잡성 개선 방안

SELinux는 TE 모델 자체의 복잡성과 시스템 내의 객체에 대한 세분화된 객체 유형의 분류, 시스템 호출 기반의 접근통제로 인한 다양한 권한집합 등의 특성 때문에 보안정책이 복잡하다. 따라서 SELinux의 보안정책의 복잡성을 개선하기 위해서는 TE 모델에서의 타입과 객체 유형 및 권한의 종류와 수량을 줄여야 한다.

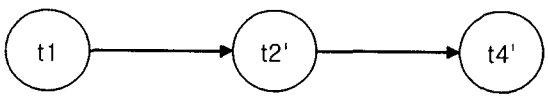
1) 타입 병합을 통한 방법

그림 4는 SELinux에서 프로그램이 실행되는 과정으로 프로세스의 타입 전이과정을 보여준다. 처음 프로세스가 시작할 때 't1'이라는 타입을 가지고 있다가, 작업흐름에 따라 't2' 또는 't3'로 분기하게 되고 다시 't4', 't5' 순서로 전이가 일어남을 나타낸다.



[그림 4] 작업흐름에 따른 타입 전이

그림 4는 간단한 타입 전이의 한 예를 보여주지만 실제로 아파치 웹서비스 데몬 등이 동작하는 과정은 훨씬 복잡한 작업흐름을 갖고 있다.



[그림 5] 단순화된 작업 흐름과 타입전이

그림 5는 그림 4의 타입을 단순화한 작업흐름을 보여준다. 이와 같이 연관성과 순차 관계를 고려하여 타입을 병합하면 작업흐름을 단순화 시켜 정책 복잡성 개선을 기대할 수 있다.

2) 객체 유형 및 권한 축소를 통한 방법

SELinux의 정책 복잡성의 다른 원인은 다양한 객체 유형과 그에 연관된 다수의 권한에 대한 정의이다. SELinux는 55개의 유형과 280여개 권한들을 정의하고 있다. 다양한 객체를 연관된 권한을 기반으로 통합하고, 권한의 연관성과 순차성을 파악하여 여러 개의 권한을 하나로 묶어 단순화 시키면 정책 복잡성의 개선을 기대할 수 있다. 표 3은 SELT의 객체 중에서 파일과 디렉터리에 객체 및 권한에 대하여 분류한 것이다.

[표 3] SELT의 객체 유형과 권한분류

유형	SELT	SELinux
dir	create	getattr, setattr, link, create, add_name, rename, relabelto, relablefrom
	remove	getattr, unlink, rmdir
	access	getattr, execute
	view	search, getattr, read
	mount	mount, remount, unmount, getattr
file	remove	unlink, remove_name
	read	read, getattr, ioctl, lock
	write	write, setattr, append
	execute	getattr, execute

SELT에서는 시스템 호출 수준의 권한들을 연관성과 순차성을 기반으로 그룹화하여 그 수를 줄여 보안정책 복잡성을 개선한다.

2. SELT 기술언어

그림 7은 템플릿 파일을 기술하는 문법을 표현하고 있다. [주체이름]은 SELT 정책 파일의 대상이 되는 프로그램이나 프로세스를 나타내는 이름으로 아파치 웹서비스 데몬의 경우 'apache_selt'와 같이 정의할 수 있다.

```
Template [주체이름] { SUB TRANS OBJS PERM }
```

[그림 6] SELT 정책 기술언어 형태

그림 7은 하나의 템플릿 파일 전체를 표현하고 있다. 템플릿 파일은 '*.s' 확장자를 가지며, 하나의 템플릿은 하나의 서비스나 프로그램을 위한 정책이 기술된다. 그림 7의 SUB, TRANS, OBJS, PERM은 그림 8에서의 *Subject, Transition, Object, Permission* 항목을 나타낸다.

```
Subject:     [주체이름]                    [프로그램유형]
Transition:  [초기타입]                    { 대상 타입 }
Object:      [객체타입]   [옵션]           { 객체정보, ...}
Permission:  [객체타입]   [객체유형]       { 권한, ...}
```

[그림 7] 세부항목 별 SELT 정책 기술언어

*Subject*는 하나의 템플릿에 오직 한 개만 정의 될 수 있으며, 대상이 되는 프로그램의 타입을 [주체이

름]으로 정해진다. [프로그램유형]은 'daemon, binary, user, none'의 네 가지 값 중 하나를 가질 수 있으며, 이 값에 따라 변환기에서 정책 변환 시에 주어지는 기본정책이 달라진다. 'daemon'은 아파치 웹서비스 데몬, 삼바데몬(smbd), DB서비스데몬(mysqld) 등의 서비스 데몬을 의미하고 'binary'인 경우에는 'ls' 나 'ps' 등의 바이너리 형태의 실행파일을 의미한다. 'user'의 경우, 시스템상의 사용자를 의미하며, 'none'의 경우에는 정의되지 않은 유형의 주체를 의미한다.

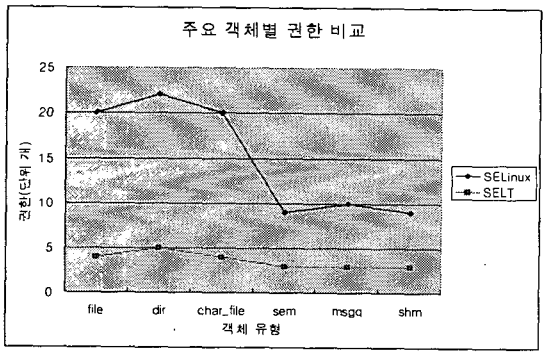
*Transition*은 SELinux의 도메인전이를 의미하며, [초기타입]을 타입으로 갖는 주체(프로그램)가 동작 중에 [대상타입]을 타입으로 갖는 객체를 실행하게 되면 Subject 항목에 정의된 [주체이름] 타입으로 전이됨을 의미하며, 복수로 정의 될 수 있다.

*Object*는 *Subject*항목에 정의된 주체가 접근하는 객체들에 대한 타입과 옵션을 정의 한다. 즉 접근하는 객체([객체정보]를 통해 얻어짐)가 갖게 될 타입을 [객체타입]으로 정의하며, [옵션]의 값으로 "in"이 선언되어 있으면 강제로 객체에 [객체타입]을 할당한다. [객체정보]에는 권한 집합 맵 파일에 정의된 객체 유형에 알맞은 값(파일이나 디렉터리의 경우 경로, 네트워크 객체의 경우 port 번호나 소켓형태 등)을 갖으며 여러 개 정의될 수 있다.

*Permission*은 주체와 객체의 권한관계를 정의한다. 주체가 Object항목에 정의된 [객체타입]을 타입으로 갖는 [객체유형]의 형태의 객체에 대해 [권한]에 정의된 권한을 갖는 다는 것을 의미하며 복수로 정의될 수 있다.

V. 연구 결과

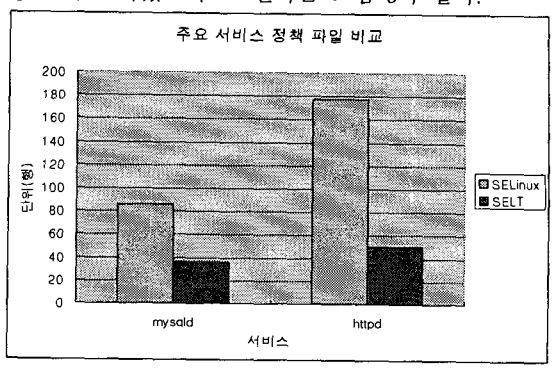
그림 8는 SELT와 SELinux의 주요 객체들에 대해서 그 유형별로 갖는 권한들의 수를 비교한 것이다.



[그림 8] 객체 유형별 권한 수

기준이 된 SELinux의 정책은 "targeted policy"이며, ".te"파일과 ".fc"파일 결과를 합한 것이다. 결과값에 해당하는 행의 수는 허용/거부 규칙, 타입전이 및 매크로를 포함한다. 비교 대상이 된 객체 유형은 "파일, 디렉터리, 문자장치 파일(chr_file), 세마포어(sem)와 메시지큐(msgq), 공유메모리(shm)"이다.

그리고 아파치 웹 서비스 데몬과 DB 서비스 데몬에 대해 템플릿 파일과 SELinux의 정책파일의 복잡성을 비교 하였으며 그 결과는 그림 9와 같다.



[그림 9] SELinux와 템플릿의 정책 복잡성비교

".te" 파일과 ".fc" 파일이 따로 관리하는 SELinux와 비교하여 SELT의 정책파일은 통합된 형태이기 때문에 보안정책을 설정하고 관리하기가 용이하다.

VI. 결론

본 논문 에서는 보안 운영체제의 필요성과, 보안정책 설정에 있어서 SELinux가 갖고 있는 복잡성과 이를 개선하기 위한 관련 연구들을 살펴보고, 보안정책 복잡성을 개선하고, 기존의 연구들이 가진 문제를 해결하기 위한 새로운 방법인 SELT를 제안 하였다. 타입 병합과 객체 유형 및 권한의 축소와 템플릿을 통한 SELinux의 보안정책의 복잡성 감소를 확인하였다.

향후에는 템플릿을 통해 일반 사용자들이 보다 쉽게 SELinux의 보안정책을 설정할 수 있도록 자동적으로 템플릿을 생성해 주는 SELT 자동 생성기에 대한 연구가 진행 되어야 한다.

[참고문헌]

[1] Bill McCarty, "SELINUX - NSA's Open Source Security Enhanced Linux", O'REILLY, 2005
[2] P. Loscocco, S. Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating".
[3] S. Smalley, Configuring the SELinux Policy, Technical report, NSA, Feb. 2002.
[4] Tresys Technology, "Security Policy Development Primer for Security Enhanced Linux", 2003.
[5] Yuichi Nakamura "Progress of SELinux Policy Editor " 2006 SELinux Symposium
[6] Flask, http://www.cs.utah.edu/flux/flask
[7] Hitachi Software, http://www.selinux.hitachi-sk.co.jp/
[8] MITRE Corporation SLAT, polgen, http://www.mitre.org/
[9] SELinux, http://www.nsa.gov/selinux/index.cfm
[10] Tresys Technology, http://tresys.com/selinux