한글 FreeBSD 핸드북

Korea FreeBSD Users Group – http://www.kr.FreeBSD.org/

Mandatory Access Control: 강제 접근제어

이 영 옥 <rick@rickinc.com> 역.

2004 년 8 월 7 일.
문서갱신번호 1.1

본 문서는 FreeBSD Documentation Project 에서 발행하는 FreeBSD Handbook 을 번역한 것으로 2004 년 8 월 7 일자 수정본을 기준으로 합니다. FreeBSD Handbook 은 지속적으로 갱신되기 때문에, 본 문서의 발간 이후에 갱신된 내용이 무엇인지 알고자 한다면, CVS Repository 를 조회하여 해당일 이후에 갱신된 내용만을 쉽게 찾아볼 수 있습니다. 또한 표지에 표시되는 날짜와 문서갱신번호는 이 문서의 갱신상태를 나타냅니다. 날짜는 번역의 대상이 되는 오리지날 핸드북의 최종 수정일자를 표시하며, 문서갱신번호는 문서에 수정이 있을 때 증가하므로 사용자가 이 문서의 갱신상태를 쉽게 파악할 수 있도록 합니다. 본 문서는 이영옥이 관리, 배포하므로 내용에 오탈자 및 오역이 발견될 경우엔 역자에게 알려주어 수정될 수 있도록 해 주십시오.

15 강제 접근제어

15.1 개요

더 넓은 세계로 뻗어 나가기 위해 더 안전한 보안 환경의 필요성이 증가하고 있다. 이러한 요구로 보안 이외의 사항은 고려하지 않는 TrustedBSD 프로젝트가 시작되었다. TrustedBSD 프로젝트는 POSIX®.1e 초안에 기반한 유저기반 유틸리티와 커널 인터페이스를 개발하여 FreeBSD 5.X 에 합병하는 것을 목표로 하고 있다. 아직은 개발 단계지만 대부분의 기능이 거의 안정화되었다. 여기에 파일시스템 접근제어 리스트(ACLs)와 강제 접근제어(MAC) 메커니즘이 포함된다.

그럼 MAC 이란 무엇인가? 강제 접근제어(Mandatory Access Controls)는 인증을 받지 않고 시스템이나 유저 데이터에 접근하는 것을 금지하기 위해 유저를 제어하는 룰이다; 또는 시스템 주체(object)나 객체(subject)에 완벽한 무결성을 제공하기 위한 것이다. 주체와 객체의 정의는 아래 내용을 본다. 정의의 필수요소는 관리자와 시스템이 제어하는 일방적인 접근제어이고 유저는 결정권이 없다.

이번 장 전체는 여기서 간단히 MAC 이라고 하는, 주로 강제 접근제어(Mandatory Access Control) 프레임 워크 기능에 대해 설명한다. 더 많은 기능이 FreeBSD 5.X 에 추가되기 때문에 추가된 기능은 이곳에서 추가적으로 설명할 것이다.

이번 장을 읽고 다음 사항을 알 수 있다:

- 현재 FreeBSD 에 포함되어 있는 MAC 모듈을 소개하고 관련된 정책을 설명한다.
- 라벨이 할당된 곳과 그렇지 않은 곳의 정책 사이에 어떤 MAC 정책을 사용할 수 있는가
- MAC 프레임 워크를 사용하도록 시스템을 어떻게 효과적으로 설정할 수 있는가
- MAC 모듈을 사용하여 다른 정책은 어떻게 설정하는가
- MAC 프레임 워크를 사용하여 안전한 환경을 어떻게 구축하는가

- 시스템이 적절히 동작하는지 확인하기 위해 MAC 설정은 어떻게 테스트하는가

이번 장을 읽기 전에 다음 사항을 알고 있어야 된다:

- 유닉스와 FreeBSD 기본을 이해해야 된다(3 장).
- 최신 FreeBSD 소스로 유지할 수 있어야 되고 커널 설정/컴파일의 기본을 알고 있어야 된다(8 장).
- 보안 사항을 이해하고 FreeBSD 에 적용할 수 있어야 한다(14 장).

주의: 다음 정보를 잘못 적용하면 시스템을 사용하기 어렵게 되어 유저들이 불편을 호소하고 Xfree86 이 제공하는 기능을 사용할 수 없게 된다. 그리고 MAC 이 완벽한 시스템 보안을 제공하는 것이 아니라는 점을 염두 해둔다. MAC 프레임 워크는 기존 보안정책을 증가시킬 뿐이다: 적절한 보안 정책과 정기적인 보안체크 없이 시스템이 완벽하게 안전하다고 믿는 것은 정말 바보 같은 짓이다.

이번 장에 포함되어 있는 예제는 단지 예제라는 것을 염두 해둔다. 이러한 설정을 중요한 시스템에 적용하지 않는다. 이런 정책을 시행하는 것은 고려해 볼만하지만 모든 것이 어떻게 동작하는지 정확하게 이해하지 못하는 사람은 전체 시스템을 되돌리고 수많은 파일과 디렉터리를 다시 설정해야 되는 것을 깨닫게 될 수 있다.

15.1.1 여기서 설명하지 않는 것은 무엇인가?

이번 장에서는 MAC 프레임 워크에 관련된 폭넓은 보안에 대해 설명한다. MAC 정책을 개발하는 것은 설명하지 않는다. 새로운 정책 작성은 이 글의 최초 목적을 완전히 벗어난다. 또한 MAC 프레임 워크에 포함된 몇몇 정책은 설명하지 않는다. 여기에는 mac_test(4), mac_stub(4)와 mac_none(4) 모듈/정책이 포함되어 있다.

이들 모듈과 이들이 제공하는 다양한 메커니즘에 대한 더 많은 정보는 매뉴얼 페이지를 참고한다.

15.2 이번 장의 핵심 용어

이번 장을 소개하기 전에 핵심 용어에 대해 설명한다. 이로서 발생할 혼란을 방지하고 새로운 용어와 정보의 갑작스런 출현을 예방한다. 다음 용어는 이번 장 전반에 걸쳐 사용되고 있다:

- *구획(compartment):* 구획은 유저가 시스템의 특정 컴포넌트에 접근할 수 있도록 주어진 프로그램 세트와 파티션 되거나 나누어진 데이터를 의미한다. 구획은 또한 워크 그룹, 부서, 프로젝트나 주제처럼 그룹화를 의미한다.

- *무결성(Integrity):* 중요한 개념으로서 무결성은 데이터에 둘 수 있는 신뢰 레벨이다. 데이터의 무결성이 증가하면 데이터의 신뢰성도 증가한다.

- *라벨:* 라벨은 파일, 디렉터리와 시스템의 다른 항목에 적용할 수 있는 보안 속성이다. 항목에 라벨을 붙여 기밀성을 표시할 수 있다; 파일에 라벨을 붙이면 그 파일에 대한 보안 레벨을 표시하여 파일, 유저, 리소스 등만 사용할 수 있다. 라벨의 의미와 판단은 정책에 따라 다르다: 어떤 정책은 라벨을 무결성을 나타내거나 비밀 주체로 처리하겠지만 다른 정책은 접근하기 위한 룰로 사용할 것이다.

- *레벨:* 보안속성의 설정을 증가시키거나 감소시킨다. 레벨을 증가시키면 보안도 향상된다.

- *멀티라벨: 멀티라벨*은 tunefs(8) 유틸리티로 싱글 유저모드에 설정할 수 있는 파일시스템 옵션이다; 부팅 중 fstab(5) 파일을 사용하여 설정하거나 새로운 파일시스템을 생성하는 동안 설정한다. 이 옵션은 파일시스템의 파일과 디렉터리에 멀티 MAC 라벨을 설정한다. 그리고 보안을 강화하기 위해 라벨을 사용하는 정책에만 적용한다.

- *주체(object):* 주체 또는 시스템 주체는 *객체(subject)*를 목표로 하는 정보 흐름의 실체다. 여기에는 디렉터리, 파일, 필드, 화면, 키보드, 메모리, 스토리지, 프린터 또는 다른 데이터 스토리지/이동용 장치가 포함된다. 기본적으로 주체는 데이터 저장소 또는 시스템 리소스다; 주체를 효과적으로 사용한다는 의미는 데이터를 효과적으로 사용한다는 것이다.

- *정책:* 주체가 어떻게 취급되는지 정의한 룰 모음이다. *정책*은 보통 특정 아이템이 어떻게 제어되는지 적어놓은 것이다. 이번 장에서 정책이라는 용어는 *보안 정책*을

의미한다.

- *민감도(sensitivity):* MLS 를 설명할 때 보통 사용된다. 민감도 레벨은 데이터가 얼마나 중요하고 비밀을 유지해야 되는지를 설명하는 용어다. 민감도 레벨이 증가되면 데이터의 중요도도 높아진다.

- *싱글 라벨:* 싱글 라벨은 데이터 흐름에 접근제어를 시행하기 위해 파일시스템 전체에 하나의 라벨만 사용하는 것이다. *멀티라벨(multilabel)* 옵션이 설정되지 않았을 때 파일시스템이 이런 식으로 설정되면 모든 파일은 같은 라벨로 설정된다.

- *객체(subject): 객체*는 주체 사이의 정보 흐름으로 유발된 활동적인 존재다: 예를 들어 유저, 유저 프로세서, 시스템 프로세스 등이 포함된다.

15.3 MAC 설명

이 모든 용어를 생각하며 MAC 프레임 워크가 어떻게 시스템의 보안을 향상시키는지 생각해 보자. MAC 프레임 워크가 제공하는 다양한 정책은 네트워크, 파일시스템, 특정 포트나 소켓에 유저 접근제어 등에 사용된다. 아마도 정책을 가장 제대로 사용하는 방법은 다중 계층(multi-layer) 보안 환경을 위해 이들 정책을 혼합하여 사용하는 것이다. 유일한 단점은 다양한 파일시스템 라벨과 유저별로 네트워크 접근제어를 설정할 때 관리적인 업무가 증가한다.

이러한 단점은 프레임 워크의 영구적인 효과를 생각하면 별 것 아니다. 필요 없는 정책 지원을 감소시키면 선택의 폭을 넓힐 뿐만 아니라 시스템의 전체적인 성능을 향상시킬 수 있다. 그리고 이상적인 적용 방법은 필요한 모든 보안 요소와 프레임 워크가 제공하는 다양한 정책을 효과적으로 적용하는 것이다.

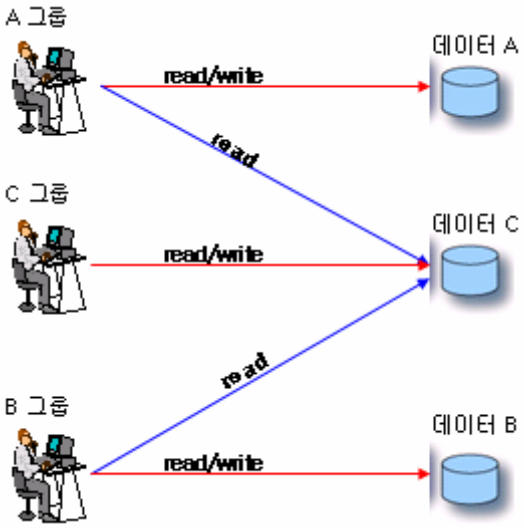
따라서 시스템에 사용되는 MAC 기능은 최소한 유저가 자신의 의지로 보안속성을 변경할 수 없어야 된다. 그리고 모든 유저 유틸리티 및 프로그램과 스크립트는 선택한 정책이 제공하는 접근 룰의 제한에서 동작해야 되며 모든 MAC 접근 룰은 시스템 관리자가 제어할 수 있어야 된다.

정확한 정책을 조심스럽게 선택하는 것은 시스템 관리자의 총 책임이다. 어떤 환경에는 네트워크를 통한 접근제어가 필요할 것이다; 이 경우 mac_portacl(4), mac_ifoff(4)와

mac_biba(4)를 생각해 볼만하다. 그렇지 않으면 파일시스템의 보안 아이템이 필요할 것이다. mac_bsdextended(4)와 mac_mls(4)와 같은 정책이 이러한 목적에 사용되고 있다.

정책 결정은 네트워크 설정을 따른다. 아마도 특정 유저 한 명만 ssh(1)가 제공하는 기능으로 네트워크나 인터넷을 사용하게 할 수 있을 것이다. mac_portacl(4)는 이런 상황에 선택하는 정책이다. 파일시스템의 경우는 어떻게 해야 되는가? 특정 디렉터리를 다른 그룹이나 유저가 사용할 수 있도록 해야 되는가? 또는 특정 주체를 등급으로 나누어서 유저나 유틸리티의 특정 파일 사용을 제한해야 되는가?

파일시스템의 경우 주체를 사용하는 것은 다른 사람이 아닌 특정 유저에게만 허용할 것이다. 예를 들어 대형 개발 팀은 각각의 작은 그룹으로 나누어 질것이다. 프로젝트 A 의 개발자들은 프로젝트 B 의 개발자들이 작성한 주체 사용은 금지될 것이다. 그러나 프로젝트 C 개발자들이 생성한 주체는 사용할 수 있어야 될 것이다; 상당히 난처한 상황이다. 이러한 경우 MAC 프레임 워크가 제공하는 다른 정책을 사용한다; 유저들을 이들 그룹으로 나누고 정보 유출에 대한 걱정 없이 적절한 영역을 사용할 수 있는 권한을 준다.



따라서 각 정책은 정보흐름과 관련된 유일한 방법을 가지게 된다. 정책 선택은 심사 숙고한 보안 정책에 전적으로 의존된다. 대부분 환경에 맞게 전체 정책을 수정하여 네트워크에 이행해야 된다. MAC 프레임 워크가 제공하는 여러 가지 정책을 이해하면 상황에 따른 최고의 정책을 선택하는데 도움이 된다.

기본 FreeBSD 커널은 MAC 프레임 워크 옵션을 포함하지 않는다; 따라서 다음 커널 옵션을 추가해야 된다:

```
options     MAC
```

그리고 커널을 다시 빌드하고 설치한다.

> **주의:** 다양한 MAC 모듈 매뉴얼 페이지에서는 커널에 빌드하도록 설명하지만 시스템에서 네트워크와 다른 요소들을 사용하지 못하게 될 수 있다. MAC 을 시행하는 것은 방화벽 시행과 비슷하지만 시스템을 완전히 잠그는 것은 피해야 된다. 이전 설정으로 되돌리는 능력을 고려해야 되겠지만 원격에서 MAC 을 시행하는 것은 굉장한 주의가 요구된다.

15.4 MAC 라벨 이해

MAC 라벨은 사실 시스템의 전체 주체와 객체에 적용할 수 있는 보안 속성이다.

라벨을 설정할 때 유저는 이것이 무엇이고 무엇과 연관되는지 확실히 이해해야 된다. 주체에 적용할 수 있는 속성은 로드된 정책에 따라 달라지고 따라서 속성의 해석도 달라진다. 이해 부족으로 설정을 잘못했다면 시스템이 예상하지 못한 동작을 하게 된다. 그리고 라벨을 너무 높거나 낮게 설정하면 디렉터리와 파일시스템의 전체 보안 구조에 결함이 생긴다

주체의 보안 라벨은 보안 접근 제어를 결정하는 일부분으로 정책에 의해 사용된다.
어떤 정책에서 라벨은 결정에 필요한 모든 정보를 가지고 있다; 다른 모델에서 라벨은 거대한 룰 세트의 일부분으로 처리된다.

예를 들어 파일에 *biba/low* 라벨을 설정하는 것은 "low" 값의 Biba 정책으로 라벨이 관리됨을 의미한다. 그리고 주체와 객체에 정책 이름과 자격을 설정하기 때문에 라벨 포맷은 간단하다.

FreeBSD 는 라벨 기능을 지원하는 정책으로 미리 정의된 라벨(low, high 와 equal 라벨) 3 가지가 제공된다. 이들은 각 정책과 다른 방식으로 접근 제어를 시행하지만 low 라벨은 가장 낮은 설정, equal 라벨은 객체와 주체를 비활성 하거나 영향을 받지 않도록 설정하고 high 라벨은 가능한 가장 높은 설정을 시행한다.

싱글 라벨 파일시스템 환경에서는 오직 하나의 라벨만 주체와 객체에 사용될 것이다. 여기서는 접근 퍼미션 한 세트를 전체 시스템과 세트 전체가 필요할 수 있는 많은 환경에 시행한다. 그러나 멀티 라벨이 시스템의 주체나 객체에 설정될 수 있는 몇 가지 경우가 있다.

이러한 경우 *멀티 라벨* 옵션이 tunefs(8)에 적용될 것이다.

Biba 와 MLS 의 경우 제어 계층의 정확한 레벨을 보여주기 위해 숫자 라벨이 설정될 것이다. 이 숫자 레벨은 파티션하거나 권한을 가진 다른 그룹으로 정보를 나누어서, 그 그룹이나 더 상위 그룹만 접근을 허용하는 등급으로 분류한다.

대부분의 경우 관리자는 파일시스템 전반에 싱글 라벨만 설정한다; 사실 이 제한은 조금 있다 설명할 *멀티라벨* 옵션을 필요로 한다.

이 방법은 DAC 와 비슷하다. MAC 은 관리자에게 엄격한 제어를 제공한다. root 는 제어와 정책을 설정할 수 있는 사람이기 때문에 유저를 적절한 카테고리/접근 레벨에 할당할 수 있다. 그렇지만 다양한 정책으로 root 유저도 제한할 수 있다. 주체를 통한 기본 제어는 그룹을 제외하지만 root 는 언제라도 설정을 변경할 수 있다. 이것은 Biba 와 MLS 같은 정책에서 다루는 계층적/인증 모델이다.

15.4.1 라벨 설정

사실상 라벨 정책을 설정하는 모든 사항은 4 개의 명령어 세트를 사용하여 시행된다. 이들 명령은 주체나 객체를 설정하거나 설정을 조정하고 확인하기 위한 간단한 인터페이스를 제공한다.

모든 사항은 setfmac(8)과 setpmac(8) 유틸리티를 각각 사용하여 설정할 수 있다. setfmac 명령은 시스템 주체에 MAC 라벨을 설정하는데 사용하지만 시스템 객체에 라벨을 설정할 때는 setpmac 명령을 사용한다. 확인해 보자:

```
# setfmac biba/high test
```

위 명령에서 에러가 없으면 프롬프트가 나타나지만 에러가 발생하면 반응을 보인다; chmod(1)과 chwon(8) 명령과 비슷하다. 제한된 주체에 라벨을 설정하거나 수정할 때 “Permission denied”가 보통 발생한다. 시스템 관리자는 이 문제를 해결하기 위해 다음 명령을 사용할 수 있다:

```
# setfmac biba/high test
``Permission denied''
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

위에서 확인 하였듯이 setpmac 는 관련된 프로세스에 다른 라벨을 할당하여 정책 설정을 덮어 쓰는데 사용할 수 있다. getpmac 유틸리티는 **sendmail** 처럼 현재 실행 중인 프로세스에 보통 사용된다: 명령어 대신 프로세스 ID 를 사용하지만 비슷한 논리다. 유저가 주체나 객체를 소유하고 있다면 그 유저만 정책 라벨을 무시할 수 있다. 유저가 자신의 접근 레벨이 아닌 파일을 조정하려고 하면 mac_set_link 기능으로 “Operation not permitted” 에러가 표시된다.

15.4.1.1 유저와 라벨 설정

유저가 직접 라벨을 가지고 있어야 되기 때문에 그들의 파일과 프로세스도 시스템에 정의된 보안 정책과 적절히 상호작용해야 된다. 이 설정은 로그인 클래스를 사용하여 login.conf 파일을 통해 설정된다. 라벨을 사용하는 모든 정책은 유저 클래스 설정을 시행한다.

모든 정책을 가진 예제 엔트리는 아래에 나열되어 있다:

```
default:₩
	:copyright=/etc/COPYRIGHT:₩
	:welcome=/etc/motd:₩
	:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:₩
	:path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:₩
	:manpath=/usr/share/man /usr/local/man:₩
	:nologin=/usr/sbin/nologin:₩
	:cputime=1h30m:₩
	:datasize=8M:₩
:vmemoryuse=100M:₩
	:stacksize=2M:₩
	:memorylocked=4M:₩
	:memoryuse=8M:₩
	:filesize=8M:₩
```

```
        :coredumpsize=8M:\
        :openfiles=24:\
:maxproc=32:\
        :priority=0:\
        :requirehome:\
        :passwordtime=91d:\
        :umask=022:\
        :ignoretime@:\
:label=partition/13,mls/5,biba/10(5-15),lomac10[2]:
```

label 옵션은 MAC 이 시행하는 유저 클래스의 기본 라벨을 설정하는데 사용한다. 유저가 이 값을 절대 수정할 수 없기 때문에 유저의 경우는 필수가 될 수 있다. 그러나 실제 설정에서 관리자는 모든 정책을 활성화하기를 원치 않을 것이다. 이들 설정을 시행하기 전에 이번 장의 나머지 내용도 검토하기 바란다.

> **Note:** 유저는 처음으로 로그인해서 그들의 라벨을 변경할 것이다; 그러나 이 변경은 객체를 제한하는 정책이다. 위의 예제에서 프로세스의 무결성은 최소 5 에서 최대 15 까지고 기본적인 라벨은 10 이라고 Biba 정책에게 지시한다. 따라서 그 프로세스는 라벨을 변경할 때까지 10 에서 실행되고 유저가 setpmac 명령을 사용한다면 로그인할 때 Biba 에 의해 강제로 범위가 정해진다.

login.conf 를 변경한 후 cap_mkdb 를 사용하여 로그인 클래스 특성 데이터베이스를 다시 빌드해야 되고 이것은 앞으로 보게 될 예제와 설명에 반영된다.

특히 많은 유저가 있는 여러 사이트에서 여러 개의 다른 유저 클래스가 필요할 때 유용할 것이다. 이 방법은 관리하기가 어렵기 때문에 상당히 심사숙고 해야 된다.

15.4.1.2 네트워크 인터페이스와 라벨 설정

네트워크를 통한 데이터 흐름을 제어하기 위해 네트워크 인터페이스에도 라벨을 설정할 수 있을 것이다. 주체에 대한 정책 기능처럼 모든 경우에 작용한다. 예를 들어 *biba* 에서 높게 설정된 유저는 낮은 라벨로 설정된 네트워크 인터페이스에 접근할 수 없다.

네트워크 인터페이스에 MAC 라벨이 설정되면 *maclabel* 이 ifconfig 에 적용될 것이다. 예를 들면 다음과 같다:

```
# ifconfig bge0 maclabel biba/equal
```

위 명령은 bge(4) 인터페이스에 *biba/equal* MAC 라벨을 설정한다. biba/high(low-high)와 비슷한 설정을 사용할 때 전체 라벨을 인용부호로 감싸야 한다. 그렇지 않으면 에러가 발생한다.

라벨을 지원하는 각 정책은 네트워크 인터페이스에서 MAC 라벨을 비활성 할 때 사용할 수 있는 tunable 를 가지고 있다. 라벨을 *equal* 로 설정하면 비슷한 효과를 가져온다. sysctl 의 결과나 정책 매뉴얼 페이지 또는 이번 장 다음에 찾을 수 있는 이들 tunables 에 대한 정보를 확인한다.

15.4.2 싱글라벨 또는 멀티라벨

기본적으로 시스템은 싱글라벨 옵션을 사용한다. 그러나 관리자에게 이것은 어떤 의미인가? 시스템 보안 모델의 유연성을 위해 허용과 거부권을 제공하기 때문에 관리자 권한에서는 약간 다르다.

싱글라벨은 오직 하나의 라벨만 허용한다. 예를 들어 각 객체나 주체에 *biba/high* 가 사용된다. 이 방법은 관리하기 편하겠지만 라벨을 지원하는 정책의 유연성을 떨어뜨린다. 많은 관리자들이 보안 정책에 멀티라벨 옵션을 사용하기를 원할 것이다.

멀티라벨 옵션은 전체 파티션에 오직 하나의 라벨만 허용하는 표준 싱글라벨 대신 각 객체나 주체가 각자의 독립적인 MAC 라벨을 가질 수 있도록 허용한다. 멀티라벨과 싱글라벨 옵션은 Biba, Lomac, MLS 와 SEBSD 정책을 포함하여 라벨 기능을 시행하는 정책에만 필요하다.

대부분 멀티라벨을 다 설정할 필요는 없을 것이다. 다음과 같은 상황과 보안 모델을 생각해보자:

- MAC 프레임워크와 다양한 정책을 섞어 사용하는 FreeBSD 웹 서버
- 이 머신은 시스템의 모든 사항에 *biba/high* 라벨 하나만 필요하다. 이곳의 파일시스템은 싱글라벨이 항상 영향을 끼치고 있기 때문에 멀티라벨 옵션은 필요 없다.

- 그러나 이 머신이 웹 서버가 되기 때문에 게시(write up) 기능을 방지하기 위해 *biba/low* 에서 웹 서버를 운용해야 된다. Biba 정책과 이 정책의 기능은 나중에 설명하기 때문에 이해하기 어렵다면 일단 계속 읽은 후 다시 되돌아온다. 실행 상태가 아니면 서버는 대부분 *biba/low* 에서 분리된 파티션 세트를 사용할 수 있다. 이 예제 설정과 유저 설정은 많이 부족하다(예를 들어 데이터 제한 같은). 그러나 이것은 단지 앞에서 설명한 것을 증명하는 예제다.

라벨이 아닌 정책이 사용된다면 멀티라벨 옵션은 필요 없다. 여기에는 *seeotheruids*, *portacl* 과 *partition* 정책이 포함된다.

파티션과 멀티라벨 기능 기반의 보안모델을 수립할 때 멀티라벨을 사용하면 파일시스템의 모든 것들이 라벨을 갖기 때문에 관리가 어려워질 수 있다. 이것은 디렉터리, 파일과 장치 노드까지 포함한다.

다음 명령은 멀티라벨을 갖기 위해 파일시스템에 멀티라벨을 설정한다. 이것은 싱글 유저 모드에서만 설정할 수 있을 것이다.

```
# tunefs -l enable /
```

스왑 파일시스템에는 필요 없다.

> **Note:** 어떤 유저는 root 파티션에 멀티라벨 플래그를 설정할 때 문제가 있었다. 이러한 경우 이번 장의 문제해결 섹션(15.16)을 참고한다.

15.4.3 Tunables 로 MAC 제어하기

모듈을 로드하지 않고도 sysctl 인터페이스로 설정할 수 있는 MAC 이 있을 것이다:

- *security.mac.enforce_fs*는 기본적으로 활성화되고 파일시스템에 MAC 파일시스템 정책이 시행된다.
- *security.mac.enforce_kld*는 기본적으로 활성화되고 동적 커널 링커에 MAC 커널 링크 정책이 시행된다(kld(4)를 본다).
- *security.mac.enforce_network*는 기본적으로 활성화되고 MAC 네트워크 정책이 시행된다.

- *security.mac.enforce_pipe*는 기본적으로 활성화되고 파이프(pipe)에 MAC 정책이 시행된다.
- *security.mac.enforce_process*는 기본적으로 활성화되고 프로세스간의 통신에 사용되는 프로세스에 MAC 정책이 시행된다.
- *security.mac.enforce_socket*는 기본적으로 활성화되고 소켓에 MAC 정책이 시행된다(socket(2) 매뉴얼 페이지를 본다).
- *security.mac.enforce_system*는 기본적으로 활성화되고 accounting과 재 부팅 같은 시스템 동작에 MAC 정책이 시행된다
- *security.mac.enforce_vm*는 기본적으로 활성화되고 가상 메모리 시스템에 MAC 정책이 시행된다.

> **Note:** 모든 정책이나 MAC 옵션은 tunables 를 지원한다. 이들은 보통 *security.mac.<policyname>* 트리에 놓인다. MAC 에서 모든 tunables 를 보려면 다음 명령을 사용한다:
>
> # sysctl -da | grep mac

기본 MAC 정책이 일반적으로 모두 시행되기 때문에 설명해야 됐다. 모듈을 커널에 빌드했다면 시스템은 상당히 제한되기 때문에 대부분 로컬 네트워크나 인터넷 연결 등과 같은 통신을 할 수 없다. 이것이 모듈을 커널에 빌드하지 말라는 이유다. 따라서 sysctl 로 비활성할 수 있는 기능을 제한하는 것이 아니고 다시 빌드하여 새로운 시스템을 설치할 필요 없이 관리자가 시스템의 정책을 일시적으로 변경할 수 있다.

15.5 모듈 설정

MAC 프레임 워크에 포함된 모든 모듈은 위에서 말했듯이 커널에 포함되어 있거나 런타임 커널 모듈에 의해 로드된다. 권장하는 방법은 /boot/loader.conf 파일에 모듈 이름을 추가하여 부팅할 때 로드하는 것이다.

다음 섹션은 다양한 MAC 모듈과 특징을 설명한다. 이들을 특정 환경에서 사용하는 것도 이번 장에서 고려하고 있다. 어떤 모듈은 “이것은 허용되지만 저것은 안 된다”처럼 라벨을 시행하여 접근 제어하는 라벨을 지원한다. 라벨 설정파일에서는 파일이 어떻게 접근되며 네트워크로 통신이 가능하지 등을 제어할 것이다. 이전 섹션에서는 파일이나 파티션 접근제어별로 활성화하기 위해 멀티레벨 플래그를 파일시스템에 어떻게 설정하는지 보여줬다.

싱글라벨 설정은 오직 하나의 라벨만 시스템에 시행하기 때문에 tunefs 옵션을 멀티라벨이라고 한다.

15.5.1 MAC seeotheruids 모듈

모듈 이름: mac_seeotheruids.ko

커널 설정 라인: *options MAC_SEEOTHERUIDS*

부트 옵션: *mac_seeotheruids_load="YES"*

mac_seeotheruids(4) 모듈은 *security.bsd.see_other_uids* 과 *security.bsd.see_other_gids* sysctl tunables 을 모방하여 확장한다. 이 옵션을 설정하기 전에 라벨을 설정할 필요 없이 다른 모듈과 동작할 수 있다.

이 모듈을 로드한 후 다음 sysctl tunables 이 기능을 제어하기 위해 사용될 것이다:

- *security.mac.seeotheruids.enabled* 는 모듈의 기능을 활성화하고 기본 설정을 사용한다. 이들 기본 설정은 다른 유저의 프로세스와 소켓을 볼 수 없게 한다.

- *security.mac.seeotheruids.specificgid_enabled* 는 특정 그룹을 이 정책에서 제외시킨다. 이 정책에서 특정 그룹을 제외시키려면 *security.mac.seeotheruids.specificgid=XXX* sysctl tunable 을 사용한다. 위 예제에서 *XXX* 는 제외시키려는 그룹 ID 번호로 대체한다.

- *security.mac.seeotheruids.primarygroup_enabled* 는 이 정책에서 특정 주 그룹을 제외시킬 때 사용한다. 이 tunable 를 사용할 때 *security.mac.seeotheruids.specificgid_enabled* 는 설정되지 않을 것이다.

root 유저를 이 정책에서 제외할 수 없음을 기억한다. 이것이 MAC 버전과 기본적으로 *security.bsd.seeotheruids* 에 포함된 표준 tunable 버전과의 가장 큰 차이점 중 하나다.

15.6 MAC bsdextended 모듈

모듈 이름: mac_bsdextended.ko

커널 설정 라인: *options MAC_BSDEXTENDED*

부트 옵션: *mac_bsdextended_load="YES"*

mac_bsdextended(4) 모듈은 파일시스템 방화벽을 시행한다. 이 모듈 정책은 표준 파일시스템 퍼미션 모델을 확장하고 파일시스템 계층에서 관리자가 파일과 유틸리티 그리고 디렉터리를 보호하기 위해 방화벽 같은 룰 세트 생성을 허용한다.

정책은 ipfw(8)과 비슷한 구문을 가지고 있는 ugidfw(8) 유틸리티를 사용하여 생성할 수 있을 것이다. 더 많은 툴은 libugidfw(3) 라이브러리 기능을 사용하여 작성할 수 있다.

이 모듈을 적용할 때 아주 조심해야 된다. 잘못된 사용은 파일시스템의 특정 부분을 사용할 수 없게 된다.

15.6.1 예제

mac_bsdextended(4) 모듈이 로드되면 다음 명령이 현재 룰 설정을 나열하기 위해 사용될 것이다:

```
# ugidfw list
0 slots, 0 rules
```

예상대로 룰은 정의되어있지 않다. 이 의미는 모든 것에 접근할 수 있다는 것이다. root 는 영향을 받지 않고 모든 유저로부터 모든 접근을 막기 위해 룰을 생성하려면 단순히 다음 명령을 실행한다:

```
# ugidfw add subject not uid root new object not uid root mode n
```

Note: FreeBSD 5.3 이전 릴리즈에는 add 매개변수가 없다. 이 경우 set 변수를 대신 사용한다. 아래 예제를 참고한다.

ls 처럼 매우 단순한 명령이라도 모든 유저가 실행할 수 없기 때문에 좋아 보이지는 않는다. 좀더 효과적인 룰은 다음과 같을 것이다:

```
# ugidfw set 2 subject uid user object uid trhodes mode n
# ugidfw set 3 subject uid user object gid trhodes mode n
```

이 명령은 유저 이름 user 로 trhodes 의 홈 디렉터리에서 디렉터리 리스트를 포함하여 모든 접근을 막는다.

user 대신 *not uid trhodes* 를 적용할 수 있다. 이 명령도 위와 동일한 제한을 모든 유저에게 시행한다.

Note: root 유저는 이 설정으로 영향을 받지 않는다.

여기서는 파일시스템을 강화하기 위해 mac_bsdextended(4) 모듈을 어떻게 사용해야 되는지 일반적인 아이디어를 제공한다. 더 많은 정보는 mac_bsdextended(4)와 ugidfw(8) 매뉴얼 페이지를 본다.

15.7 MAC ifoff 모듈

모듈 이름: mac_ifoff.ko

커널 설정 라인: *options MAC_IFOFF*

부트 옵션: *mac_ifoff_load="YES"*

mac_ifoff(4) 모듈은 시스템이 최초로 부팅하는 동안 네트워크 인터페이스를 비활성화하기 위해 단독으로 존재한다. 시스템에 라벨을 설정할 필요 없고 다른 MAC 모듈에 의존하지도 않는다.

대부분은 아래에 나열된 sysctl tunables 를 통해 제어된다.

- *security.mac.ifoff.lo_enabled* 은 루프백(lo(4)) 인터페이스의 모든 트래픽을 활성/비활성 한다.
- *security.mac.ifoff.bpfrecv_enabled* 은 버클리 패킷 필터 인터페이스(bpf(4))의 모든 트래픽을 활성/비활성 한다.
- *security.mac.ifoff.other_enabled* 은 다른 모든 인터페이스의 트래픽을 활성/비활성 한다.

mac_ifoff(4)를 가장 일반적으로 사용하는 방법은 부팅하는 동안 네트워크 트래픽이 허용되지 않는 환경에서 네트워크를 모니터링 하는데 사용한다. 다른 권장되는 사용법은 보호되고 있는 디렉터리에서 새로운 파일이나 수정된 파일이 발견되면 자동으로 네트워크 트래픽을 차단하기 위해 security/aide를 사용하는 스크립트 작성에 사용하는 것이다.

15.8 MAC portacl 모듈

모듈 이름: mac_portacl.ko

커널 설정 라인: *MAC_PORTACL*

부트 옵션: *mac_portacl_load="YES"*

mac_portacl(4) 모듈은 다양한 sysctl 변수를 사용하여 로컬 TCP 와 UDP 포트 바인딩을 제약할 때 사용할 수 있다. 사실 mac_portacl(4)는 root 가 아닌 유저가 미리 정의된 특정 포트에(1024 보다 낮은 포트) 바인드 할 수 있게 한다.

로드 되면 이 모듈은 모든 소켓에 MAC 정책을 활성화한다. 다음과 같은 tunables 을 활성화할 수 있다:

- *security.mac.portacl.enabled* 정책을 완벽히 활성/비활성 한다(버그 때문에

security.mac.portacl.enabled sysctl 변수는 FreeBSD 5.2.1 이나 이전 릴리즈에서 작동하지 않는다).

- *security.mac.portacl.port_high* 는 mac_portacl(4)가 보호하려는 가장 높은 번호를 설정한다.

- *security.mac.portacl.suser_exempt* 는 0 이 아닌 값이 설정됐을 때 root 유저를 이 정책에서 제외시킨다.

- *security.mac.portacl.rules* 는 실제 mac_portacl 정책을 지정한다; 아래를 본다.

security.mac.portacl.rules sysctl 에서 지정하였듯이 실제 *mac_portacl* 정책은 *rule[,rule,...]* 형식의 텍스트 문자열로 필요한 룰을 정의한다. 각 룰은 *idtype:id:protocol:port* 형식으로 되어있다. *Idtype* 매개변수는 각각 유저 id 나 그룹 id 가 되는 *uid* 나 *gid* 가 되어 *id* 매개변수를 해석하는데 사용된다. *protocol* 매개변수는 *tcp* 나 *udp* 로 매개변수를 지정하여 룰을 TCP 에 적용할지 UDP 에 적용할지 결정하는데 사용한다. 마지막 *port* 매개변수는 유저나 그룹이 바인드할 수 있도록 지정된 포트번호다.

> **Note:** 룰 셋이 커널에 의해 직접 해석되기 때문에 오직 숫자 값만 유저 ID, 그룹 ID, 그리고 포트 매개변수에 사용할 수 있다. 다시 말해 유저, 그룹 그리고 포트 서비스 이름은 사용할 수 없다.

유닉스와 같은 운영체제에서 1024 이하의 포트는 특정 권한을 가진 프로세스에 바인드 하는데 사용한다. 즉 root 가 실행하는 프로세스를 의미한다. mac_portacl(4)는 권한이 없는 프로세스를 1024 이하의 포트에 바인드하기 위해 표준 유닉스 제한을 비활성 한다. 이것은 sysctl(8) 변수 *net.inet.ip.portrange.reservedlow* 와 *net.inet.ip.portrange.reservedhigh* 을 0 으로 설정하면 된다.

더 많은 정보는 아래 예제나 mac_portacl(4) 매뉴얼 페이지를 참고 한다.

15.8.1 예제

다음 예제는 위의 설명을 이해하기 쉽게 해준다:

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0 net.inet.ip.portrange.reservedhigh=0
```

첫째 표준 권한을 가진 포트에 mac_portacl(4)를 설정하고 일반적인 유닉스 바인드 제한을 비활성 한다.

```
# sysctl security.mac.portacl.suser_exempt=1
```

root 유저는 이 정책의 영향을 받지 않기 때문에 *security.mac.portacl.suser_exempt*을 0 이 아닌 값으로 설정한다. 이제 mac_portacl(4) 모듈은 기본적으로 유닉스처럼 동작하도록 시스템 동작을 설정한다.

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

UID 80 의(보통 www 유저) 유저가 포트 80 번과 바인드할 수 있게 한다. 따라서 www 유저가 root 권한 없이 웹 서버를 실행할 수 있다.

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

1001 의 UID 유저가 TCP 포트 110(“pop3”)번과 995(“pop3s”)번에 바인드 하게한다. 이 명령은 이 유저가 포트 110 과 995 번에서 연결을 허용하는 서버를 시작하도록 한다.

15.9 라벨 기능과 MCA 정책

다음 섹션 몇 가지는 라벨을 사용하는 MAC 정책에 대해 설명한다.

이 섹션부터 mac_biba(4), mac_lomac(4), mac_partition(4), 그리고 mac_mls(4)의 기능에 대해 설명한다.

> **Note:** 이것은 예제 설정이기 때문에 중요한 제품에는 사용하지 않는다. 이 문서의 목적은 시행과 테스트를 위한 구문과 예제를 보여 주는 것이다.

이들 정책이 정확히 작동하기 위해 몇 가지 준비가 필요하다.

15.9.1 라벨 정책 준비

다음 내용이 login.conf 파일에 필요하다:

- *insecure* 클래스나 비슷한 종류의 다른 클래스를 추가해야 된다. Insecure 로그인 클래스는 필요 없지만 여기 예제에서만 사용한다; 다른 설정은 다른 클래스 이름이 필요할 것이다.
- *Insecure* 클래스에는 다음 설정과 정의가 필요하다. 이들 중 몇 가지는 수정할 수 있지만 기본 라벨을 정의하는 라인은 필요하다.

```
insecure:₩
        :copyright=/etc/COPYRIGHT:₩
        :welcome=/etc/motd:₩
        :setenv=MAIL=/var/mail/$,BLOCKSIZE=K:₩
        :path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:₩
        :manpath=/usr/share/man /usr/local/man:₩
        :nologin=/usr/sbin/nologin:₩
        :cputime=1h30m:₩
        :datasize=8M:₩
        :vmemoryuse=100M:₩
        :stacksize=2M:₩
        :memorylocked=4M:₩
        :memoryuse=8M:₩
        :filesize=8M:₩
        :coredumpsize=8M:₩
        :openfiles=24:₩
        :maxproc=32:₩
        :priority=0:₩
        :requirehome:₩
        :passwordtime=91d:₩
        :umask=022:₩
        :ignoretime@:₩
        :label=partition/13,mls/5,biba/low:
```

유저가 새로운 클래스로 변경되기 전에 cap_mkdb(1) 명령을 login.conf(5)로 실행해야 된다.

root 도 login 클래스에 두어야 된다. 그렇지 않으면 root 가 실행하는 거의 모든 명령은 setpmac 를 사용해야 된다.

> **주의:** login.ocnf 데이터베이스를 다시 빌드하면 데몬 클래스와 나중에 어떤 에러가 발생할 수 있다. 간단히 데몬 계정을 주석처리하고 데이터베이스를 다시 빌드하는 것이 이 문제를 줄일 수 있다.

- MAC 라벨을 시행하는 모든 파티션은 멀티라벨을 지원해야 된다. 여기 있는 많은 예제는 테스트 목적의 다른 라벨을 가지고 있기 때문에 이렇게 해야 된다. 예방책으로 mount 명령의 결과를 확인한다.
- 새로운 유저 클래스에 시행된 높은 보안 메커니즘을 가진 유저를 변경한다. pw(8)이나 vipw(8)로 이 쉽게 변경할 수 있다.

15.10 MAC 파티션 모듈

모듈 이름: mac_partition.ko

커널 설정 라인: *options MAC_PARTITION*

부트 옵션: *mac_partition_load="YES"*

mac_partition(4) 정책은 프로세스를 MAC 라벨 기반의 특정 “파티션”에서 벗어나지 못하게 한다. 비교 하기는 어렵겠지만 이것을 특별한 종류의 jail(8)로 생각한다.

이것은 loader.conf(5) 파일에 추가해야 되는 하나의 모듈이기 때문에 부팅하는 동안 정책을 로드하고 활성화한다.

이 정책을 위한 대부분의 설정은 아래에서 설명하는 setpmac(8) 유틸리티를 사용하여 설정된다. 다음 sysctl tunable 을 이 정책에 사용할 수 있다:

- *security.mac.partition.enabled* 는 MAC 프로세스 파티션의 시행을 활성화한다.

이 정책이 활성화되면 유저는 자신의 프로세스만 볼 수 있게 되고 특정 유틸리티로 작업 하지 못하게 된다. 예를 들면 insecure 클래스 위의 유저는 top 명령과 다른 프로세스를 생성하는 많은 명령을 사용할 수 없게 된다.

유틸리티를 파티션 라벨에 설정하기 위해 setpmac 유틸리티를 사용한다:

```
# setpmac partition/13 top
```

이 명령은 *insecure* 클래스 유저에게 라벨 셋으로 top 명령을 추가한다. 유저가 생성하는 *Insecure* 클래스의 모든 프로세스는 *partition/13* 라벨에서 실행된다.

15.10.1 예제

다음 명령은 파티션 라벨과 프로세스 리스트를 보여준다:

```
# ps Zax
```

다음 명령은 다른 유저의 프로세스 파티션 라벨과 그 유저가 현재 실행하는 프로세스를 보여 준다:

```
# ps -ZU trhodes
```

> **Note:** 유저는 mac_seeotheruids(4) 정책이 로드되지 않는다면 root 라벨의 프로세스를 볼 수 있다.

아주 교묘하게 실행하는 방법은 /etc/rc.conf 에서 모든 서비스를 비활성하고 적절한 라벨 세트로 시작되는 스크립트를 실행한다.

> **Note:** 다음 정책은 제공되는 3 개의 기본라벨 대신 정수 설정을 지원한다. 제한을 포함하고 있는 이들 옵션은 모듈 매뉴얼 페이지에 더 자세히 설명되어 있다.

15.11 MAC 다중 계층 보안 모듈

모듈 이름: mac_mls.ko

커널 설정 라인: *options MAC_MLS*

부트 옵션: *mac_mls_load="YES"*

mac_mls(4) 정책은 정보 흐름 정책을 엄격하게 시행하여 시스템의 객체와 주체에 접근하는 것을 제어한다.

MLS(다중 계층 보안) 환경에서 “clearance” 레벨을 구획에 따라 각 객체나 주체의 라벨에 설정한다. 이들 clearance 나 sensibility 레벨은 6000 이상의 번호에 영향을 미친다; 이들 레벨을 각 객체나 주체에 완벽하게 설정하는 것은 시스템 관리자에게 엄청난 일이 된다. 다행히 3 개의 “instant” 라벨은 이미 이 정책에 포함되어 있다.

이들 라벨은 *mls/low* 와 *mls/equal* 그리고 *mls/high* 다. 이들 라벨은 매뉴얼 페이지에 자세히 설명되어 있기 때문에 여기서는 간단히 설명한다:

- *mls/low* 라벨은 다른 모든 주체에 의해 제어되는 낮은 설정을 포함한다. *mls/low* 로 라벨된 모든 것들은 낮은 clearance 레벨을 가지고 있기 때문에 높은 레벨의 정보에 접근할 수 없다. 게다가 이 라벨은, 이들 라벨에 쓰기나 정보를 추가하는 높은 clearance 레벨 주체를 허용하지 않는다.

- *mls/equal* 라벨은 정책에서 제외해야 되는 주체에 적용해야 된다.

- *mls/high* 라벨은 가능한 가장 높은 레벨의 clearace 다. 이 라벨에 할당된 주체는 시스템의 다른 모든 주체보다 우선권을 가진다; 그러나 낮은 클래스 주체에 정보를 누출시키지 못하게 한다.

MLS 는 다음과 같은 것을 제공한다:

- 계층이 없는 카테고리 설정으로 계층적인 보안 레벨을 제공한다.

- 읽을 수 없고 작성할 수 없는 고정된 룰을 제공한다(객체는 자신의 레벨이나 하위 주체에 접근하기 위해 읽을 수 있지만 상위 레벨은 읽을 수 없다).
- 비밀을 제공한다(부적절한 데이터 공개를 방지한다).
- 시스템 디자인의 기본적인 목적은 여러 개의 민감한 레벨에서(비밀과 신뢰 사이에서 정보 누출 없이) 데이터를 동시에 제어하기 위함이다.

다음 sysctl tunables 은 특별한 서비스와 인터페이스를 설정하는데 사용할 수 있다:

- *security.mac.mls.enabled* 는 MLS 정책을 활성/비활성 한다.
- *security.mac.mls.ptys_equal* 는 *mls/equal* 을 생성하는 동안 모든 pty(4) 장치에 라벨을 할당한다.
- *security.mac.mls.revocation_enabled* 은 라벨이 낮은 계층의 라벨로 변경된 후 주체의 접근을 폐지한다.
- *security.mac.mls.max_compartments* 는 주체에 최대치의 구획 레벨을 설정할 때 사용된다. 그리고 기본적으로 최대 구획 번호는 시스템에서 허용된다.

MLS 라벨을 조정하기 위해 setfmac(8) 명령이 제공되었다. 주체에 라벨을 할당하기 위해 다음 명령을 사용한다:

```
# setfmac mls/5 test
```

파일 test 에 MLS 라벨을 할당하려면 다음 명령을 사용한다:

```
# getfmac test
```

여기서는 MLS 의 정책 기능을 요약하였다. 또 다른 방법은 MLS 정책 정보를 지정하는 마스터 정책파일을 /etc 파일에 생성하고 setfmac 명령에 이 파일을 적용한다. 이 방법은 모든 정책을 다룬 후 설명한다.

확인: 낮은 clearance 의 주체는 높은 clearance 프로세스를 볼 수 없다. 기본 정책은 쓰기가 필요 하더라도 읽을 수 없는 모든 것에 *mls/high* 를 시행한다. 읽기가 필요하더라도 쓸 수 없는 모든 것에 *mls/low* 를 시행한다. 그리고 나머지에 *mls/equal* 을 시행한다. *Insecure* 로 표시된 모든 유저는 *mls/low* 에 설정해야 된다.

15.12 MAC Biba 모듈

모듈 이름: mac_biba.ko

커널 설정 라인: *options MAC_BIBA*

부트 옵션: *mac_biba_load="YES"*

mac_biba(4) 모듈은 MAC Biba 정책을 로드한다. 이 정책은 정보흐름이 약간 반대로 되어 있는 것을 제외하고 MLS 정책과 거의 비슷하게 동작한다. 이 정책은 민감한 정보가 아래로 흐르는 것을 방지 하지만 MLS 정책은 민감한 정보가 위로 흐르는 것을 방지한다; 따라서 이 섹션의 대부분을 두 정책에 적용할 수 있다.

Biba 환경에서 “무결성” 라벨을 각 객체나 주체에 설정한다. 이들 라벨은 계층적인 등급과 계층적이지 않은 구획을 생성한다. 주체나 객체의 등급이 올라가기 때문에 무결성도 마찬가지로 올라간다.

아래에서 설명하듯이 지원되는 라벨은 *biba/low*, *biba/equal* 그리고 *biba.high* 다:

- *biba/low* 라벨은 주체나 가지고 있을 객체의 무결성도를 가장 낮게 설정한다. 이것을 주체나 객체에 설정하면 높게 표시된 주체나 객체에 쓰기 접근을 방지한다.
- 이 정책에서 제외시키기 위해 *biba/equal* 라벨을 주체에 설정할 수 있다.
- *biba/high* 라벨은 낮은 라벨로 설정된 주체에 쓰기를 허용하지만 이 주체를 읽을 수 없다. 이 라벨을 전체 시스템의 무결성에 영향을 주는 주체에 설정할 것을 권장한다.

Biba 는 다음과 같은 사항을 제공한다:

- 계층적이지 않은 무결성 카테고리를 설정하여 계층적인 무결성 레벨을 제공한다.
- 작성할 수 없고 읽을 수 없는 고정된 룰을 제공한다(MLS 와 반대). 객체는 상위가 아닌 자신의 레벨에서 주체에 쓰기 접근을 할 수 있다. 비슷하게 객체는 하위가 아닌 자신의 레벨 위의 주체에 읽기 접근을 할 수 있다;
- 무결성(데이터의 부적합한 변경을 방지한다).
- 무결성 레벨(민감한 레벨의 MLS 대신).

다음 sysctl tunables 는 Biba 정책을 조정하는데 사용할 수 있다.

- *security.mac.biba.enabled* 는 타겟 머신에 Biba 정책 시행을 활성/비활성 하는데 사용할 것이다.
- *security.mac.biba.ptys_equal* 는 pty(4) 장치에 Biba 정책을 비활성 하는데 사용할 것이다.
- *security.mac.biba.revocation_enabled* 는 라벨이 객체를 제어하도록 변경되었다면 주체의 접근을 강제로 취소한다.

시스템 주체에 설정된 Biba 정책에 접근하기 위해 setfmac 와 getfmac 명령을 사용한다:

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

확인: 무결성이 낮은 객체는 무결성이 높은 객체에 쓰기를 할 수 없다; 무결성이 높은 객체는 확인할 수 없거나 무결성이 낮은 주체를 읽을 수 없다.

15.13 MAC LOMAC 모듈

모듈 이름: mac_lomac.ko

커널 설정 라인: *options MAC_LOMAC*

부트 옵션: *mac_lomac_load="YES"*

MAC Biba 정책과 다르게 mac_lomac(4) 정책은 무결성 레벨이 낮아진 후 무결성 룰을 깨지 않고 무결성이 더 낮은 주체로 접근하는 것을 허용한다.

예전 lomac(4) 시행과 혼돈하지 말아야 되는 MAC 버전의 낮은-워터마크 무결성 정책은 예비 등급 구획으로 객체 등급을 낮추기 위해 플로팅(floating) 라벨을 사용하는 것을 제외하고 Biba 와 아주 비슷하다. 이 두 번째 구획은 *[auxgrade]* 형식을 가지고 있다. 예비 등급(auxiliary grade)으로 lomac 정책을 할당할 때 숫자 2 가 예비 등급인 *lomac10[2]* 형식과 비슷하다.

MAC LOMAC 정책은 무결성이 낮은 주체를 객체가 읽을 수 있게 하고 나중에 무결성이 높은 주체에 쓰기를 방지하기 위해 객체의 라벨을 강등시키는 무결성 라벨과 시스템의 모든 주체의 라벨에 산재해 있다. 이것은 위에서 설명한 *[auxgrade]* 옵션이기 때문에 정책은 더 많은 호환성을 제공하고 Biba 보다 초기 설정이 적다.

15.13.1 예제

Biba 와 MLS 정책처럼 setfmac 와 setpmac 유틸리티가 시스템 주체에 라벨을 적용할 때 사용될 것이다.

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

여기서 예비 등급은 *low* 이고 MAC LOMAC 정책으로만 제공되는 기능이다.

15.14 MAC 으로 안전한 환경 구축

다음 데모는 적절히 설정된 정책과 다양한 MAC 모듈을 사용하여 구축한 안전한 환경이다. 이것은 오직 테스트이므로 실제 환경에는 적용하지 않는다. 이 말을 무시한다면 절대 작동하지 않고 실제 환경에서 문제가 될 수 있다.

이 설명을 진행하기 전에 *멀티라벨* 옵션을 각 파일시스템에 설정해야 된다. 설정하지 않으면 에러가 발생한다.

15.14.1 insecure 유저 클래스 생성

다음 유저 클래스를 /etc/login.conf 파일에 추가한다:

```
insecure:₩
:copyright=/etc/COPYRIGHT:₩
:welcome=/etc/motd:₩
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:₩
:path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
:manpath=/usr/share/man /usr/local/man:₩
:nologin=/usr/sbin/nologin:₩
:cputime=1h30m:₩
:datasize=8M:₩
:vmemoryuse=100M:₩
:stacksize=2M:₩
:memorylocked=4M:₩
:memoryuse=8M:₩
:filesize=8M:₩
:coredumpsize=8M:₩
:openfiles=24:₩
:maxproc=32:₩
:priority=0:₩
:requirehome:₩
:passwordtime=91d:₩
:umask=022:₩
:ignoretime@:₩
```

```
:label=partition/13,mls/5:
```

그리고 다음 라인을 기본 유저 클래스에 추가한다:

```
:label=mls/equal,biba/equal,partition/equal:
```

이것이 끝나면 데이터베이스를 다시 빌드하도록 다음 명령을 실행한다:

```
# cap_mkdb /etc/login.conf
```

15.14.2 정확한 모듈로 부팅

다음 라인을 /boot/loader.conf 에 추가하여 시스템이 초기화하는 동안 필요한 모듈을 로드한다:

```
mac_biba_load="YES"
mac_mls_load="YES"
mac_seeotheruids_load="YES"
mac_partition_load="YES"
```

15.14.3 모든 유저를 Insecure 로 설정

root 나 시스템 유저가 아닌 모든 유저 계정은 이제 로그인 클래스가 필요하다. 로그인 클래스가 필요하지만 없다면 유저는 vi(1)처럼 일반적인 명령을 사용하는데 제약을 받는다. 다음 sh 스크립트를 사용하면 된다:

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' ₩
    /etc/passwd`; do pw usermod $x -L insecure; done;
```

위 명령이 실행된 후 /etc/master.passwd 로 cap_mkdb 명령을 실행해야 된다.

15.14.4 완벽한 설정

이제 contexts 파일을 생성해야 된다; 다음 예제는 Robert Watson 의 예제 정책으로 /etc/policy.contexts 를 생성한 것이다

```
# This is the default BIBA/MLS policy for this system.

.*                                  biba/high,mls/high
/sbin/dhclient                      biba/high(low),mls/high(low)
/dev(/.*)?                          biba/equal,mls/equal
# This is not an exhaustive list of all "privileged" devices.
/dev/mdctl                          biba/high,mls/high
/dev/pci                            biba/high,mls/high
/dev/k?mem                          biba/high,mls/high
/dev/io                             biba/high,mls/high
/dev/agp.*                          biba/high,mls/high
(/var)?/tmp(/.*)?                   biba/equal,mls/equal
/tmp/₩.X11-unix                     biba/high(equal),mls/high(equal)
/tmp/₩.X11-unix/.*                  biba/equal,mls/equal
/proc(/.*)?                         biba/equal,mls/equal
/mnt.*                              biba/low,mls/low
(/usr)?/home                        biba/high(low),mls/high(low)
(/usr)?/home/.*                     biba/low,mls/low
/var/mail(/.*)?                     biba/low,mls/low
/var/spool/mqueue(/.*)?             biba/low,mls/low
(/mnt)?/cdrom(/.*)?                 biba/high,mls/high
(/usr)?/home/(ftp|samba)(/.*)?      biba/high,mls/high
/var/log/sendmail₩.st               biba/low,mls/low
/var/run/utmp                       biba/equal,mls/equal
/var/log/(lastlog|wtmp)             biba/equal,mls/equal
```

이 정책은 왼쪽에 나열된 디렉터리의 유틸리티가 보내고 받는 정보 흐름에 제한을 설정하여 보안을 시행한다.

이 파일을 다음 명령으로 시스템에 읽어 들인다:

```
# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /usr
```

Note: 위의 파일 시스템 레이아웃은 환경에 따라 다를 것이다.

/etc/mac.conf 파일은 메인 섹션에서 다음과 같이 수정해야 된다:

```
default_labels file ?biba,?mls
default_labels ifnet ?biba,?mls
default_labels process ?biba,?mls,?partition
default_labels socket ?biba,?mls
```

15.14.5 설정 테스트

이 테스트를 위해 adduser 명령으로 유저를 추가하여 *insecure* 클래스에 넣는다.

아래 예제는 root 와 일반 유저를 혼합하여 테스트 한다; 구분하기 위해 프롬프트를 확인한다.

15.14.5.1 기본 라벨 테스트

```
% getpmac
biba/15(15-15),mls/15(15-15),partition/15
# setpmac partition/15,mls/equal top
```

Note: top 프로세스는 다른 top 프로세스를 실행하기 전에 종료된다.

15.14.5.2 MAC Seeotheruids 테스트

```
% ps Zax
biba/15(15-15),mls/15(15-15),partition/15   1096 #C:   S       0:00.03 -su (bash)
biba/15(15-15),mls/15(15-15),partition/15   1101 #C:   R+      0:00.01 ps Zax
```

다른 유저들의 프로세스를 보는 것은 허용하지 않는다.

15.14.5.3 MAC Partition 테스트

나머지 테스트를 위해 MAC *seeotheruids* 정책을 비활성 한다:

```
# sysctl security.mac.seeotheruids.enabled=0
% ps Zax
LABEL                                                PID   TT   STAT       TIME
COMMAND
  biba/equal(low-high),mls/equal(low-high),partition/15   1122 #C:   S+      0:00.02
top
  biba/15(15-15),mls/15(15-15),partition/15               1096 #C:   S       0:00.05
-su (bash)
  biba/15(15-15),mls/15(15-15),partition/15               1123 #C:   R+      0:00.01
ps Zax
```

Biba 정책은 높은 라벨이 적용된 주체를 읽을 수 있게 한다.

```
# setpmac partition/15,mls/equal,biba/low top
% ps Zax
LABEL                                          PID   TT   STAT       TIME COMMAND
  biba/15(15-15),mls/15(15-15),partition/15   1096 #C:   S       0:00.07 -su (bash)
  biba/15(15-15),mls/15(15-15),partition/15   1226 #C:   R+      0:00.01 ps Zax
```

Biba 정책은 낮게 라벨 된 주체를 읽지 못하게 하지만 MLS 는 읽을 수 있게 한다.

```
% ifconfig bge0 | grep maclabel
maclabel biba/low(low-low),mls/low(low-low)
% ping -c 1 192.0.34.166
PING 192.0.34.166 (192.0.34.166): 56 data bytes
  ping: sendto: Permission denied
```

유저는 이러한 이유로 example.com 이나 다른 도메인에 핑을 보낼 수 없다.

이 에러가 발생하는 것을 방지하기 위해 다음 명령을 실행한다:

```
# sysctl security.mac.biba.trust_all_interfaces=1
```

이 명령은 기본 인터페이스 라벨을 insecure 모드로 설정하기 때문에 기본 Biba 정책 라벨은 시행되지 않는다.

```
# ifconfig bge0 maclabel biba/equal₩(low-high₩),mls/equal₩(low-high₩)
% ping -c 1 192.0.34.166
PING 192.0.34.166 (192.0.34.166): 56 data bytes
  64 bytes from 192.0.34.166: icmp_seq=0 ttl=50 time=204.455 ms
  --- 192.0.34.166 ping statistics ---
  1 packets transmitted, 1 packets received, 0% packet loss
  round-trip min/avg/max/stddev = 204.455/204.455/204.455/0.000 ms
```

더 정확한 라벨을 설정하여 ping 을 보낼 수 있다.

몇 가지 읽기와 쓰기 테스트를 위해 몇 개의 파일을 생성한다:

```
# touch test1 test2 test3 test4 test5
# getfmac test1
test1: biba/equal,mls/equal
# setfmac biba/low test1 test2; setfmac biba/high test4 test5; ₩
  setfmac mls/low test1 test3; setfmac mls/high test2 test4
# setfmac mls/equal,biba/equal test3 && getfmac test?
  test1: biba/low,mls/low
```

```
   test2: biba/low,mls/high
   test3: biba/equal,mls/equal
test4: biba/high,mls/high
   test5: biba/high,mls/equal
# chown testuser:testuser test?
```

이 모든 파일은 testuser 유저가 소유해야 된다. 읽기 테스트를 해보자:

```
% ls
test1    test2    test3    test4    test5
% ls test?
ls: test1: Permission denied
ls: test2: Permission denied
ls: test4: Permission denied
test3    test5
```

이제 쌍을 유지하지 못한다; 예: *(biba/low,mls/low)*와 *(biba/low,mls/high)* 그리고 *(biba/high,mls/high)*쌍. 물론 읽기 접근도 거부된다. 쓰기 테스트를 해보자:

```
% for i in `echo test*`; do echo 1 > $i; done
-su: test1: Permission denied
-su: test4: Permission denied
-su: test5: Permission denied
```

읽기 테스트처럼 쓰기 접근도 쓰기 쌍으로 허용되지 않는다; 예: *(biba/low,mls/high)*와 *(biba/equal,mls/equal).*

```
% cat test?
cat: test1: Permission denied
cat: test2: Permission denied
1
cat: test4: Permission denied
```

이제 root 에서 다음 명령을 실행한다:

```
# cat test2
1
```

15.15 다른 예제: 웹 서버를 고립시키기 위해 MAC 사용

유저가 접근할 수 있는 웹 데이터를 저장할 분리된 위치를 지정한다. 여기서 *biba/high* 프로세스가 웹 데이터에 접근할 수 있도록 한다.

웹 데이터를 저장할 디렉터리를 생성한다:

```
# mkdir /usr/home/cvs
```

이제 cvs 로 초기화 한다:

```
# cvs -d /usr/home/cvs init
```

첫 번째 목적은 *biba* 정책을 활성화하는 것이기 때문에 /boot/loader.conf 에 *mac_biba_enable="YES"*를 입력한다. 여기서는 MAC 지원을 커널에서 활성화했다고 가정한다.

여기서 시스템의 모든 것에 *biba/high* 를 기본적으로 설정해야 된다.

login.conf 파일의 기본 유저 클래스 아래를 다음과 같이 변경한다:

```
:ignoretime@:₩
    :umask=022:₩
    :label=biba/high:
```

모든 유저를 기본 클래스에 등록해야 된다; 다음 명령을 사용하면 몇 분 후 완료된다.

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' ₩
    /etc/passwd`; do pw usermod $x -L default; done;
```

기본 클래스를 복사하여 *biba/low* 라벨 설정으로 web 클래스를 생성한다.

cvs 저장소에 저장된 메인 웹 데이터에 작업하는 유저를 생성한다. 이 유저를 새로운 로그인 클래스 web 에 등록한다.

모든 곳에 *biba/high* 가 기본이기 때문에 저장소도 마찬가지다. 웹 데이터도 마찬가지이므로 유저가 웹 데이터를 읽고 쓸 수 있다; 그러나 웹 서버는 *biba/high* 유저가 접근해야 되는 데이터를 서비스하기 때문에 전체 데이터를 강등시켜야 된다.

이를 위한 완벽한 툴은 FreeBSD 에서 기본적으로 제공하는 sh(1)과 cron(8)이다. 다음 스크립트가 원하는 것을 해결해준다:

```
PATH=/bin:/usr/bin:/usr/local/bin; export PATH;
CVSROOT=/home/repo; export CVSROOT;
cd /home/web;
cvs -qR checkout -P htdocs;
exit;
```

Note: 보통 cvs ID 태그를 웹 사이트 데이터 파일에 적용해야 된다.

이 스크립트를 web 의 홈 디렉터리에 두고 다음 crontab(1) 엔트리를 추가한다:

```
# Check out the web data as biba/low every twelve hours:
0       */12    *       *       *       web     /home/web/checkout.sh
```

이 명령은 매 12 시간마다 머신에서 HTML 소스를 확인한다.

웹 서버를 시작하는 기본 시작 방법도 *biba/low* 에서 프로세스를 시작하도록 수정해야 된다. 이것은 /usr/local/etc/rc.d/apache.sh 스크립트에 다음 명령을 입력하면 된다:

```
command="setpmac biba/low /usr/local/sbin/httpd"
```

Apache 설정을 *biba/low* 정책으로 동작하도록 수정해야 된다. 이 경우 *biba/low* 로 설정된 디렉터리에 로그파일을 추가하도록 소프트웨어를 수정해야 되고 그렇지 않으면 “access denied” 에러를 보게 된다.

> **Note:** 이 예제는 *docroot* 가 /home/web/htdocs 를 지정하도록 설정한다. 그렇지 않으면 **Apache** 는 제공할 문서를 저장하려고 할 때 실패한다.

PID 파일, *Scoreboardfile*, *DocumentRoot*, log 파일 위치 또는 쓰기 접근에 필요한 다른 변수를 포함한 다양한 설정도 수정해야 된다. *biba* 를 사용할 때 *biba/low* 가 설정되지 않은 곳의 모든 쓰기 접근은 거부된다.

15.16 MAC 프레임 워크 문제 해결

개발 단계 동안 몇 명의 유저가 일반적인 설정 문제를 보고했다. 이들 문제 중 몇 가지를 설명한다:

15.16.1 /에 *multilabel* 옵션을 활성화할 수 없다.

mltilabel 플래그가 내 root(/) 파티션에서 활성화되지 않았다

50 명 중 한 명의 유저가 이 문제를 겪게 됐고 우리도 초기 설정을 하는 동안 이 문제를 겪게 됐다. 이 문제를 자세히 확인한 결과 이 버그는 정확하지 않은 문서나 문서를 제대로 이해하지 못해서 발생하는 것이었다. 어쨌든 다음 절차로 이 문제를 해결할 수 있다:

① root 파티션을 읽기만할 수 있도록 /etc/fstab 에 *ro* 로 설정한다.

② 싱글 유저모드로 재 부팅한다.

③ /에서 tunefs *-l enable* 을 실행한다.

④ 보통모드로 시스템을 재 부팅한다.

⑤ mount *–urw* /를 실행하고 /etc/fstab 의 *ro* 를 *rw* 로 변경한 후 시스템을 다시 재

부팅한다.

⑥ root 파일시스템에 *멀티라벨*이 제대로 설정되었는지 확인하기 위해 mount 결과를 다시 확인한다.

15.16.2 MAC 설정 이후 Xfree86 이 시작되지 않는다.

MAC 으로 안전한 환경을 구축한 후 Xfree86 을 실행할 수 없다.

이 문제는 MAC *파티션* 정책이나 MAC 라벨정책 중 잘못된 라벨로 발생할 수 있다. 디버그 하려면 다음 사항을 따른다:

① 에러 메시지를 확인한다; 유저가 *insecure* 클래스에 있다면 *파티션* 정책이 문제가 된다. 유저의 클래스를 다시 *default* 클래스로 설정하고 cap_mkdb 명령으로 데이터베이스를 다시 빌드한다. 그래도 해결되지 않는다면 다음 단계를 시행한다.

② 라벨 정책을 다시 확인한다. 정책이 유저, XFree86 어플리케이션 그리고 /dev 엔트리에 정확히 설정되어있는지 확인한다.

③ 둘 다 문제를 해결하지 못한다면 에러 메시지와 환경을 설명한 메일을 TrustedBSD 웹 사이트(http://www.trustedbsd.org/)에 있는 TrustedBSD 토론 리스트나 FreeBSD 일반 질문 메일링 리스트 (http://lists.freebsd.org/mailman/listinfo/freebsd-questions)에 보낸다.