보안레이블 확장을 통한 윈도우 서버 보안

정창성*, 이윤희*
*티에스온넷㈜
e-mail:csjung@tsonnet.co.kr

Security-Enhanced Windows Server with the Expansion of Security Label

Chang-Sung Jung*, Yun-Hee Lee*
*Research Center, TSonNet Co.,Ltd.

요 약

어플리케이션 또는 네트워크 레벨의 외곽 방어에 의한 보안 기능의 한계로 인하여 운영체제 내부 보안에 대한 필요성이 증대되고 있다. 그에 따라 시스템상에서의 또는 시스템에 의한 행동을 제어하기 위한 차세대 보안솔루션으로 보안 운영체제가 부각되고 있다. 이에 본 논문에서는 안전한 운영체제 구축을 위한 보안 요구 사항의 기준이 될 수 있는 다중등급 보안에 의한 윈도우 서버 보안 강화 기술을 소개하고 본 논문에서 설계하고 구현한 보안 커널의 기능을 중심으로 기술한다. 또한 기존의 전형적인 보안레이블을 확장하여 추가적으로 제어할 수 있도록 수정된 보안 모델을 제시한다.

1. 서론

정보통신 분야의 비약적인 성장으로 인하여 인터넷 등 컴퓨터 네크워크를 이용한 다양한 형태의 공격 패턴에 의한 정보 침해 사고가 빈번하게 발생하고 있다. 운영체제 및 어플리케이션 소프트웨어의 보안 결함에 대응하기 위한 기존의 방화벽이나 침입탐지 시스템의 한계로 인하여 중요 정보를 여러 가지 위협 요소들로부터 보호하기 위해 잘 정의된 보안 정책과 보안 요구사항을 만족하는 보안 운영체제의 필요성이 절실한 상황이다[1].

미국은 보안 및 신뢰성이 입증된 컴퓨터 시스템을 국방성 및 정부기관에 보급하기 위하여 1985 년에 신뢰성 컴퓨터 평가 기준(TCSEC : Trusted Computer System Evaluation Criteria)[2]을 7 가지 등급으로 분류하고 그에 따라 각 기관별 특성에 맞는 컴퓨터 시스템을 도입 및 운영하도록 권고하고 있다[2]. TCSEC 은 보안정책, 책임성, 보증 및 지속적인 보호 등의 기본적인 컴퓨터 보안 요구사항을 규정하고 있다. 국내에서도 정보보안의 제반 문제를 근원적으로 해결하기 위하여 보안 운영체제 개발에 대한 연구가 진행되고 있다. TCSEC B1 급 이상의 컴퓨터 시스템에서 구현하는 보안 운영체제는 보안 커널(Security Kernel)을 채택하고 있다.

본 논문에서는 자원에 대한 접근을 효율적으로 제어하기 위하여 수정된 BLP(Bell & LaPadula) 모델을 윈도우 커널에 적용하여 다중등급 보안(Multi-Level Security) 강제적 접근제어(MAC : Mandatory Access Control) 모델을 제시하고 그에 따라 시스템을 설계하고 구현하였다[3,4]. 추가적으로 보안레이블을 확장하여 불법적인 실행이나 변조로부터 윈도우 서버를 보호하기 위한 방법을 제시하였다.

본 논문은 2 장에서 수정된 BLP 모델이 적용된 다중등급 보안 시스템 설계 및 보안레이블에 대해 설명하고, 3 장에서는 윈도우 커널에서 보안 커널을 구현하기 위한 파일시스템 필터 드라이버 구현 기술을 중심으로 기술한다.

2. 다중등급 보안

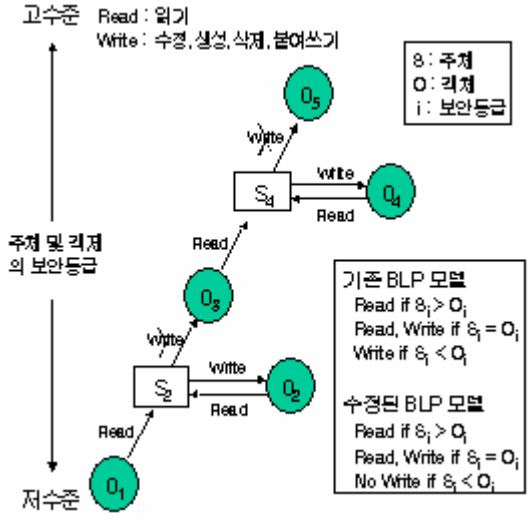
2.1 다중등급 보안의 개요

다중등급 보안이란 주체(사용자, 프로세스 등)에 보안등급(Clearance)과 보호범주(Category)를 부여하고 보안등급과 보호범주가 부여된 임의의 객체(파일, 디렉터리, 레지스터리 등)에 접근하는 것을 통제하는 방식으로서 보안 권한이 엄격히 요구되는

조직에서 필수적으로 요구되는 보안 기능이다. 다중등급 보안 정책은 주체의 다중등급 보안 범위와 객체의 다중등급 보안 범위간의 관계를 결정한 뒤 다중등급 보안 접근을 결정하는 BLP 모델의 한 형태이다. 본 논문에서 구현한 보안 커널은 모든 주체와 객체에 대한 보안레이블 부여 및 새로운 프로세스의 생성에 따른 보안레이블의 자동 상속화, 수정된 BLP 모델에 따른 강제적 접근제어 시스템 등을 포함한다.

2.2 적용된 BLP 접근제어 모델

BLP 모델은 접근제어 규칙을 기술한 정형화된 보안 정책 모델의 하나로 시스템 보안을 위한 규칙 준수 규정과 주체의 객체에 대한 접근 허용 범위를 규정하고 있다[1]. 강제적 접근제어 정책은 객체에 포함된 정보의 비밀성과 이러한 비밀 정보에 대하여 주체가 갖는 정형화된 권한에 근거하여 객체에 대한 접근을 제한하는 방법이다. 그러므로 강제적 접근제어 정책을 적용하기 위하여 컴퓨터에 저장된 객체들은 해당되는 비밀로 분류되어 존재하여야 한다. 현재까지 개발된 많은 강제적 접근제어 기법들은 미국방성의 다중등급 보안 정책에 근간을 두고 있는 것이 대부분이다. 이와 같이 컴퓨터에 저장된 비밀 정보를 보호하기 위한 하나의 방법론이 다중등급 보안이며, 이를 위하여 연구 개발된 최초의 수학적인 모델이 BLP 모델이다. BLP 모델은 서로 다른 보안 등급을 가지고 있는 데이터를 다루는 시스템에서 불법적인 정보 유출을 막기 위해 필요한 보안 요구 조건에 대해 다음 그림과 같이 정의하고 있다.



(그림 1) BLP 모델

① 주체 S 는 객체 O 를 오직 C(S) >= C(O)일 경우에만 읽을 수 있다.
② 주체 S 는 객체 O 를 오직 C(S) =< C(O)일 경우에만 쓸 수 있다.

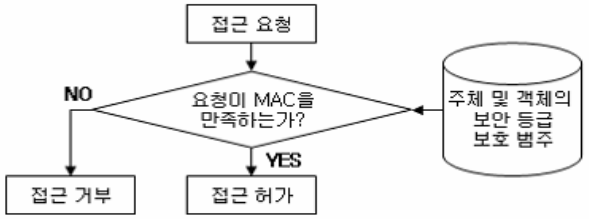
기존 BLP 모델은 ②와 같이 C(S) < C(O)일 경우에도 쓰기 연산을 허용한다. 그러나 낮은 등급의 주체가 더 높은 등급의 객체에 쓰기가 가능하다는 것은 보안상 문제가 발생되므로 쓰기 연산을 제한하여야 한다. 따라서 본 구현에서는 ②' 와 같이 수정된 BLP 모델을 적용하였다[4].

②' 주체 S 는 객체 O 를 오직 C(S) = C(O)일 경우에만 쓸 수 있다.

즉, 주체 S 는 객체 O 와 등급이 같은 경우에만 읽기 및 쓰기가 가능하고, 객체보다 등급이 높은 경우에는 낮은 등급의 객체를 읽기만 가능하고, 객체보다 등급이 낮은 경우에는 자신보다 높은 등급의 객체에 접근할 수 없다. 이러한 BLP 모델의 성질을 시스템에 적용하여 더 높은 등급의 정보를 읽는다거나 더 높은 등급의 주체가 더 낮은 등급의 객체에 정보를 쓰는 것을 방지하여 정보의 불법적인 흐름을 차단하였다.

2.3 강제적 접근제어

다음 그림과 같이 어떤 주체가 어떤 객체를 읽거나 쓰고자 할 때 그 주체가 그 객체에 대한 접근 권한을 가지고 있는지 체크한다.



(그림 2) 강제적 접근제어

주체와 객체에 부여되는 보안레이블은 강제적 접근제어에 필수적인 메커니즘으로 모든 주체와 객체에 안정적으로 유지되어야 한다[4]. 주체와 객체의 보안등급 및 보호범주를 포함하는 데이터 구조를 설계하여 주체의 보안 레이블은 보안 커널 내부의 각 프로세스의 정보 구조체에 설정하고 객체의 보안레이블은 윈도우의 NTFS(NT File System)내의 모든 파일에 존재하는 확장 파일 속성을 사용한다.

2.3.1 주체의 보안레이블

시스템 내에서 주체는 프로세스이다. 프로세스는 실행중인 프로그램을 의미하며 사용자가 컴퓨터에 로그인한 이후 사용자의 요청을 처리하기 위해 동작한다. 강제적 접근제어를 위한 프로세스의 보안레이블은 보안등급과 보호범주로 구성된다. 이 정보는 보안관리자에 의해 안전하게 등록되어야 하며 보안 커널 내부에서 획일적으로 유지되어야 한다. 다음 그림은 프로세스에게 보안레이블을 설정하기 위해 보안 커널 내부에 구성된 프로세스 정보 구조체를 보이고 있다.

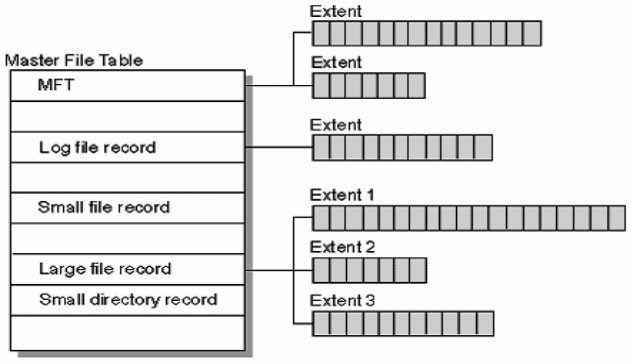
```
typedef struct _PROCESS_ENTRY {
 struct _PROCESS_ENTRY  *PreviousProcess;
  PCHAR    ProcessName;
  DWORD    ProcessID;
  ...
  ...
  ULONG    Clearance;
  ULONG    Category;
  struct _PROCESS_ENTRY  *NextProcess;
} PROCESS_ENTRY, * PPROCESS _ENTRY;
```

(그림 3) 프로세스 보안레이블

2.3.2 객체의 보안레이블

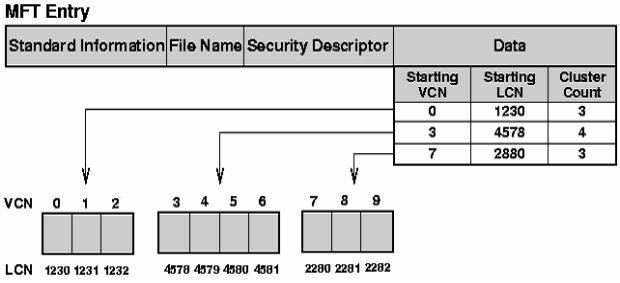
프로세스가 생성한 객체들은 객체가 담고 있는 정보의 기밀성에 따라 그에 적합한 보안등급과 보호범주가 부여되어야 한다. 이를 위해 각 객체에 대한 보안레이블 정보를 저장 및 안전하게 유지할 수 있는 보안레이블의 기억장소가 필요하다. 객체에 대한 보안레이블은 보안등급을 가진 프로세스가 새로운 객체를 만들어 낼 때 각 프로세스의 보안등급과 보호범주를 확인하여 윈도우 NTFS 내의 확장 파일 속성 구조 내에 저장한다.

다음 그림에서 보이고 있는 마스터 파일 테이블은 윈도우 NTFS 파일 포맷의 핵심적인 기술이다[5]. 파일 포맷을 NTFS 로 지정하면 FAT 파일 포맷과는 달리 NT 볼륨에서의 파일에 대한 정보 파일의 일부분인 마스터 파일 테이블을 생성한다.



(그림 4) 마스터 파일 테이블

마스터 파일 테이블은 각 파일과 디렉터리에 대한 정보를 저장한다. 마스터 파일 테이블의 복사본과 마스터 파일 테이블 및 복사본의 포인터는 디스크의 부트 섹터에 저장되며, 부트섹터의 복사본이 디스크의 논리적 중앙에 보관된다. NTFS 파일 포맷에서의 객체들은 내용 이외에도 접근 권한, 생성 시간, 최근 수정한 날짜 등과 같은 여러 가지 메타 정보를 갖는다. 다음 그림에서 보이고 있는 확장 파일 속성은 윈도우 운영체제가 지원하는 기본적인 메타 데이터 외에 사용자가 임의로 파일에 메타 정보를 부여하기 위해서 사용된다.



(그림 5) 확장 파일 속성

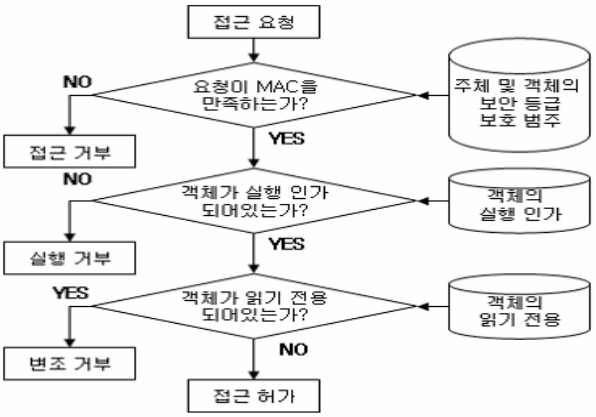
2.4 확장된 강제적 접근제어

본 논문에서는 전형적인 보안 모델을 수정하여 보안레이블의 확장을 시도하였다. 윈도우 운영체제는 바이러스나 웜 등의 공격에 치명적이다. 바이러스나 웜 혹은 트로이 목마, 백도어 등은 사용자가 인식하지 못하도록 실행되어 시스템을 공격하거나 특정 실행 파일을 변조하여 차후 공격을 위한 데이터를 수집한다. 이러한 공격을 방지하고자 객체의 보안레이블을 확장하여 전형적인 다중등급 보안의 보안등급, 보호범주 보안레이블 이외에 실행인가 및 읽기전용 보안레이블을 추가하였다.

실행인가 보안레이블은 윈도우 서버에서 실행이 인가되지 않은 실행파일의 실행을 금지하기 위한 목적으로 추가되었다. 즉, 수정된 BLP 모델에 따라 주체가 특정 실행파일에 대한 모든 접근 권한을 획득하였다 하더라도 실행파일의 실행인가 보안레이블에 실행인가 권한이 부여되어 있지 않다면 실행이 차단된다.

읽기전용 보안레이블은 불법적인 파일 변조로부터 파일을 보호하기 위해 추가된 보안레이블이다. 윈도우 운영체제가 기본적으로 지원하는 읽기전용 속성은 사용자가 변경할 수 있지만 읽기전용 보안레이블은 인가된 보안 관리자만 변경할 수 있다.

다음 그림은 실행인가 및 읽기전용 보안레이블이 확장된 강제적 접근제어 모델을 보이고 있다.



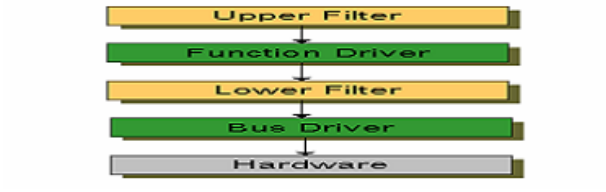
(그림 6) 확장된 강제적 접근제어

3. 파일시스템 필터 드라이버

윈도우 운영체제 상에서 수정된 BLP 모델을 구현하기 위해서는 커널 모드에서 동작하는 파일 시스템 필터 드라이버를 작성하여야 한다.

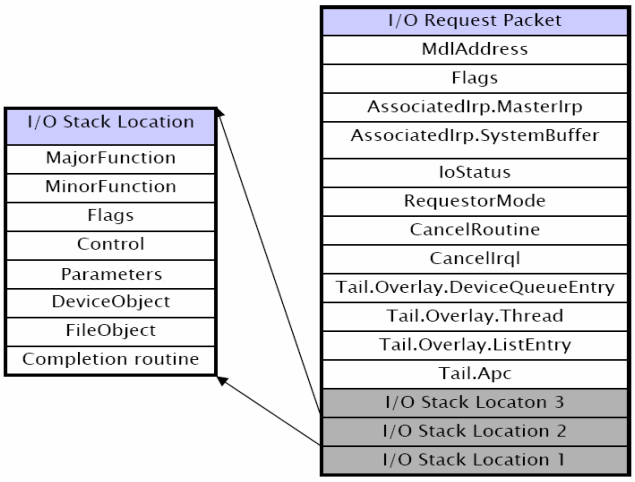
윈도우 드라이버 모델(WDM : Windows Driver Model)의 계층상 평션 드라이버는 디바이스의 주된 기능 및 기초 I/O 연산을 관리한다[6]. 파일시스템에서의 평션 드라이버인 파일시스템 드라이버는 운영체제 내의 파일을 관리하기 위한 시스템 드라이버이다. 즉, 파일 읽기, 쓰기, 실행 등 파일과 디렉터리에 관련된 모든 동작을 수행한다.

다음 그림과 같이 평션 드라이버의 상위 혹은 하위에 위치하여 표준적인 행동을 변경하는 드라이버를 필터 드라이버라고 한다. 파일시스템 상에서의 파일시스템 필터 드라이버는 특정 드라이버와의 입출력을 감시하거나 지원되지 않는 기능들을 추가할 목적으로 작성된 드라이버이다[7,8,9].



(그림 7) 윈도우 드라이버 계층

실질적으로 다른 디바이스 드라이버를 구현할 때와 마찬가지로 필터 드라이버를 구현할 때도 드라이버가 처리할 것으로 예상되는 입출력 요청 패킷(IRP : Io Request Packet)의 타입들을 제외한 IRP 의 모든 타입을 위해 디스패치 루틴을 등록해야 한다. IRP 는 운영체제가 커널 모드 드라이버와 통신하기 위해 사용하는 구조체이다[6]. 커널 모드 드라이버가 IRP 를 생성할 때면 해당 IRP 와 조합된 IO_STACK_LOCATION 구조체가 생성된다. 이 구조체는 IRP 를 처리할 각각의 드라이버를 위한 스택 공간과 IRP 를 위한 타입 코드와 매개 변수 정보 등을 포함하고 있다.



(그림 8) IRP 와 IO_STACK_LOCATION

그림 8 에서 MajorFunction 은 IRP 와 관련된 주된 함수 코드이다. 이는 드라이버 객체의 Major Function 테이블 안의 디스패치 함수 포인터 중의 하나와 일치하며 IRP_MJ_CREATE 등과 같은 값을 갖는다. 이 코드는 특정한 드라이버를 위한 I/O 스택 공간에 있다. IRP 는 IRP_MJ_CREATE 와 같이 IRP 를 시작할 수 있고 드라이버의 스택 하위로 진행되면서 그 밖의 무엇인가로 변형된다고 할 수 있다.

강제적 접근제어 및 실행, 변조 제어는 본 논문에서 구현한 파일시스템 필터 드라이버의 핵심이다. 데이터를 파일시스템에 전송하기 전에 필터링하여 필요한 작업을 수행하기 위한 선처리 루틴과 파일시스템에서 처리한 IRP 결과를 응용단에 리턴하는 시점을 필터링하여 필요한 작업을 수행하는 후처리 루틴으로 구분하여 처리한다. 응용단에서 특정 객체를 접근하기 위한 IRP 를 생성하면 선처리 과정에서 주체 및 객체의 보안레이블을 확인하여 접근 권한을 체크하여 강제적 접근제어를 실시한다. 후처리 과정에서는 객체 생성시 주체의 보안레이블을 상속하여 객체의 보안레이블을 생성한다.

4. 결론

본 논문에서는 다중등급 보안 정책을 적용하여 강제적 접근제어 윈도우 시스템을 구현하고 기존의 전형적인 보안레이블을 확장하여 윈도우 서버의 보안을 강화하였다. 기존의 BLP 모델을 시스템에 적용하기에 부적절한 부분이 있어 이를 수정한 BLP 모델을 적용하였다. 수정된 BLP 모델이 적용된 강제적 접근제어 시스템에서는 보안레이블이 부여되지 않은 주체 또는 보안레이블이 맞지 않는 주체가 보안레이블이 부여되어 있는 객체에 접근하는 것을 근본적으로 차단할 수 있다. 다른 측면으로 시스템 관리자라고 하더라도 다른 주체의 보안레이블이 부여되어 있는 파일을 임의적으로 접근하는 것이 불가능하다. 이와 같이 구현된 시스템은 커널 모드에서 동작하기 때문에 운영체제의 하위 계층인 커널에서 정보보안의 주요 결정이 이루어진다. 그 결과 정보가 존재하는 기억장치의 가장 근접한 곳에서 접근제어가 이루어지므로 시스템의 오버 헤드를 줄일 수 있어 정보보안에 따른 성능저하를 최소화할 수 있다.

본 논문에서는 객체에 보안레이블을 부여하기 위해 윈도우 확장 파일 속성을 사용하였다. 하지만 이 필드는 한계가 있기 때문에 보안레이블의 사용 범위가 한정적이거나 다양한 레이블의 사용이 제한된다. 또한, 다른 프로그램에서 같은 필드를 사용하는 경우가 발생할 수 있으며, 차후 파일 시스템 변경 등의 원인으로 인하여 확장 파일 속성을 사용하는데 예기치 않은 문제가 발생할 수 있다. 이러한 문제점을 근본적으로 해결하기 위해서는 운영체제 차원의 지원이 필수적이다. 즉, 공통된 표준을 마련하고 이를 기반으로 안정적이면서도 시스템 성능에 최소한의 영향을 미치도록 설계된 강제적 접근제어 솔루션을 만들어야 한다.

참고문헌

[1] Bell. D. and Lapadula, "Secure Computer System : Mathematical Foundations and Model", MITRE Report MTR 2547, 1973.
[2] DoD, “Trusted Computer System Evaluation Criteria”, DoD 5200.28.STD, 1985.
[3] 김현정, 박태규, 조인구, 임연호, "다중 등급 보안 리눅스 시스템 개발과 보안 인터페이스", 정보보호학술발표회논문집, 2000.
[4] 한서대학교, “정보통신 기반 구조 보호를 위한 보안서버 모델 연구”, 한국정보보호센터, 1999.
[5] http://technet.microsoft.com
[6] Walter Oney, "Programming the Windows Driver Model", Microsoft Press, 1999.
[7] Rajeev Nagar, “Windows NT File System Internals” O'Reilly. 1997.
[8] Dabak, Phadke and Borate, "Undocumented Windows NT", M&T Books, 1999.
[9] Viscarola, Mason, "Windows NT Device Driver Development", New Riders, 1999.