경량화된 타입 강제를 이용한 안전한 Embedded Linux의 설계

박성진*, 하홍준*, 이창훈*
*건국대학교 컴퓨터공학과
e-mail:{graphia, greatsk, chlee}@konkuk.ac.kr

A Design of Secure Embedded Linux using Light-weighted Type Enforcement

Sung-Jin Park*, Hong-Joon Ha*, Chang-Hun Lee*
*Dept of Computer Engineering, Konkuk University

요 약

여러 임베디드 시스템 운영체제 중에서 임베디드 리눅스는 다양한 오픈 소스 S/W를 사용할 수 있고, 다양한 임베디드 시스템에 이식할 수 있다는 장점 때문에 널리 사용되고 있다. 하지만, 임베디드 리눅스는 리눅스의 기본 접근제어 메커니즘인 임의적 접근제어(Discretionary Access Control, DAC) 기법을 그대로 사용하고 있어서 사용자의 Identity가 도용 당하거나 Trojan Horse와 같은 프로그램이 설치될 경우, 접근제어가 효력을 상실하게 된다는 결점을 가지고 있다. 더욱 문제가 되는 것은 DAC의 특성상, 프로세스가 필요 이상의 과도한 특권을 가지고 실행되며, 그 결과 잘못된 프로세스가 그 자신과 관계없는 프로그램이나 운영체제의 커널마저 손상시키는 결과를 낳을 수 있다는 것이다. 이에 따라 보다 강건한 접근제어 메커니즘에 대한 연구의 필요성이 대두되고 있다. 본 논문에서는 임베디드 리눅스 운영체제의 접근제어 메커니즘이 가지고 있는 보안적 결점에 대해서 알아보고, 이 결점을 보완하기 위해 타입 강제(Type Enforcement, TE) 기법을 사용함으로써, 임베디드 시스템에 적합하면서 강력한 접근제어를 제공할 수 있는 안전한 임베디드 리눅스 시스템에 대한 설계 모델을 보여주고자 한다.

1. 서론

기술의 발달과 시장의 요구에 의해서 많은 임베디드 시스템들이 등장하고 있다. 초기의 임베디드 시스템은 단순한 기능들만이 요구되었기 때문에 하드웨어로 구성한 단순한 회로와 시그널 등으로 충분한 운영이 가능했지만, 점차 다양한 기능과 관리가 요구됨에 따라, 요즈음은 비교적 높은 사양의 하드웨어로 구성되며, 이를 효과적으로 운영하기 위해서 임베디드 운영체제가 이식되고 있다.

여러 임베디드 운영체제 중에서, 임베디드 리눅스는 리눅스를 임베디드 시스템에 적합하도록 경량화한 운영체제이기 때문에, 리눅스에 사용 가능한 다양한 오픈소스들을 그대로 설치해 볼 수 있다는 장점을 가지고 있다. 실제로 많은 임베디드 장비들에 임베디드 리눅스 운영체제가 이식되어 운영되고 있다. 구체적인 예로는 기업이나 기관의 내부 네트워크를 외부 네트워크로부터 보호하는 방화벽(Firewall)이나 LAN간의 연결을 제공하는 라우터(Router)등을 들 수 있다.

이러한 장비들에 임베디드 리눅스가 이식됨에 따라 장비는 다양한 하드웨어 사양으로 구성되어 서비스를 제공할 수 있게 되었고, 다른 운영체제에 비해 빠른 유지/보수가 가능하게 되었다.

하지만, 이러한 장점의 이면에서는 임베디드 리눅스가 가지고 있는 보안적인 취약성으로 인한 여러 가지 문제점들이 발생하여, 시스템의 정상적인 동작을 저해하는 사례가 적지 않다.

임베디드 리눅스의 보안적 취약성은 리눅스의 임의적 접근제어(Discretionary Access Control, 이하 DAC) 메커니즘에 기인하며, 과도한 특권, 파일의 취약성, 파괴된 실행, 취약한 인증과 같은 것들을 예로 들 수 있다. 시스템이 악의적인 사용자들에게 이러한 취약점들에 대하여 집중적으로 공격을 받게 되면, 임베디드 시스템은 서비스 제공에 방해를 받

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성지원사업의 연구결과로 수행되었음

거나 최악의 경우에는 서비스를 제공할 수 없게 된다.

이러한 문제점들은 보다 강건한 접근제어 메커니즘을 도입함으로써 해결할 수 있는데, 여러 접근제어 기법들 중에서도 타입 강제(Type Enforcement, 이하 TE) 기법은 강제적 접근제어(Mandatory Access Control, 이하 MAC) 기법과 같은 강력한 접근제어를 제공하면서도, 다양한 시스템 구조를 소화할 수 있는 장점을 가지고 있다.

본 논문에서는 2장에서 임베디드 리눅스 시스템의 보안적 취약성에 대하여 알아보고, 그에 대한 대책으로서 타입 강제 기법을 제시한다. 3장에서는 타입 강제 기법을 임베디드 리눅스에 사용하기 위한 설계와 경량화 방안을 제시한다. 마지막으로 4장에서는 결론을 제시한다.

2. 임베디드 리눅스의 보안적 취약성 및 보안대책 연구

2.1 임베디드 리눅스의 보안적 취약성

DAC은 어떤 주체(Subject)가 어떤 객체(Object)에 접근하고자 할 때, 주체나 그것이 속해있는 그룹의 Identity에 근거하여 객체에 대한 접근을 제한하는 방법이며, 접근을 제한하기 위해서 사용되는 허가(Permission)를 변경할 수 있는 사용자는 객체의 생성자 혹은 소유자이다. 그리고 허가권을 변경할 수 있는 사용자는 임의의 다른 사용자에게 자신이 소유하고 있는 객체에 대한 허가권을 부여할 수 있다. DAC의 이러한 특성은 사용자에게는 편리성을 주었지만, 보안적인 측면에서는 중앙 집중적인 특권 관리나 접근제어를 어렵게 하고, 다음과 같은 취약성을 만들어 낸다.

①과도한 특권 : 프로그램이 수행에 필요한 권한 이상의 특권을 가지고 수행됨.

②파괴된 실행 : 시스템 외부에서 유입된 데이터 내부에 있는 프로그램 코드나 명령들이 시스템에 의해서 실행됨.

③파일의 취약성 : 보안적으로 중요한 설정파일이나 로그파일이 외부의 공격자에게 쉽게 노출됨.

④취약한 인증 : 보안적 결정들이 공격받기 쉬운 인증 데이터에 기반해서 이루어짐.

위와 같은 취약성을 이용하여 악의적인 사용자는 다음과 같은 과정을 거쳐 다른 사용자의 Identity를 획득할 수 있다.

버그가 있는 어떤 프로그램이 과도한 특권을 가지고 실행되고 있는 상황에서, 악의적인 사용자는 잘 다듬어진 공격 데이터를 프로그램으로 전송함으로써 파괴된 실행을 일으킨다. 그 결과 데이터 내부에 있는 프로그램 코드나 명령들은 보안적으로 중요한 설정파일등을 악의적인 사용자의 필요에 맞게 수정할 수 있게 된다. 인증에 사용되는 파일등을 수정하게 되면, 악의적인 사용자는 취약한 인증 메커니즘을 통과할 수 있게 되고 그 결과, 다른 사용자의 Identity를 획득하게 된다.

2.2 보안대책 연구

DAC이 가지는 취약성을 보완하려면, 강건한 접근제어 기법이 필요하며, 다음과 같은 사항들이 고려되어야 한다.

①특권의 최소화(Least Privilege) : 모든 프로그램은 그 프로그램 자신이 정상적으로 동작하기 위해 필요한 만큼의 특권만을 가지고 실행되어야 함.

②프로그램의 실행 영역 제한 : 프로그램은 자신과 관계없는 다른 영역의 프로그램이나 운영체제에 접근할 수 없어야 함.

③관리의 일원화 : 보안적으로 중요한 환경 설정 파일을 수정하거나, 로그 파일을 열람하거나 하는 행위들을 특정 관리 프로그램에게만 허용하고, 이 관리 프로그램이 시스템 외부로부터 유입된 데이터 내에 존재하는 실행 코드로부터 실행될 수 없도록 하여야 함.

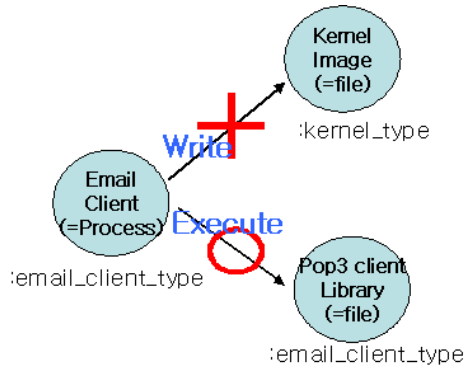
위의 세 가지 사항을 만족시키는 예로 강제적 접근제어(MAC) 기법을 들 수 있는데, 이 기법은 각 프로그램과 리소스에 등급(Level)을 나타내는 레이블(Label)을 달고, 그 레이블을 바탕으로 접근을 제어하는 방법이다. 주체와 객체의 레이블이 같은 수준을 나타내면 접근을 허용하고, 다른 수준을 나타내면, 비밀 강조 정책이나, 무결성 강조 정책에 의해서 접근 허용 여부를 판단해서 접근을 제어하게 된다. 등급에 따른 엄격한 분류를 이용하기 때문에, 계층적 구조를 가진 집단에서 사용될 경우, 강력한 접근 제어를 제공해주지만, 다양한 프로그램과 리소스가 동등한 입장에서 실행되어야 하는 비 계층적 구조에 대해서는 부적절하다. 이러한 비 계층적 구조에서는 TE를 사용하면 MAC과 같은 강력한 접근제어를 제공받을 수 있다.

TE는 프로그램과 리소스에 각각 적절한 타입(Type)을 배정하고, 타입들 간에 허용할 접근(Access)만을 명시한 후, 접근 제어 행렬을 구성하여 이를 바탕으로 접근을 제어하는 방법이다[3, 4, 7]. 즉, 모든 프로그램은 다른 프로그램이나 리소스에 대해서 명시적으로 허용된 접근이외에는 어떤 접근도 할 수 없고, 실행되고 있는 동안에도 타입에 의해 강제를 받기 때문에, 위의 세 가지 사항을 모두 만족 시킨다.

TE는 접근 결정(Access Decision)과 전이 결정(Transition Decision), 두 가지 결정을 통하여 접근제어를 수행한다. 접근 결정은 프로세스가 다른 프로세스나 파일등에 접근할 때, 접근을 허용할 것인지 제한할 것인지를 판단하는 결정이며, 전이 결정은 프로세스 실행이나 파일 생성의 경우, 그 프로세

스나 파일에게 어떤 타입을 부여할 것인가를 판단하는 결정으로서, 접근 결정을 통과한 후에 이루어진다. 예를 들어, 프로세스1이 프로그램A를 실행할 경우, 접근 결정 과정에서는 프로세스1이 프로그램A를 실행할 권한이 있는지 판단하고, 권한이 있다면, 전이 결정 과정을 수행한다. 전이 결정 과정에서, 프로그램A가 일반적인 프로그램일 경우에는 타입 전이가 이루어지지 않고, 프로세스1과 같은 타입으로 결정이 되어 실행된다. 하지만, 프로그램A가 다른 타입으로 전이할 수 있는 특별한 프로그램으로 지정되어 있는 경우, 프로그램A는 타입은 프로세스1의 타입에서 지정된 타입으로 전이되게 된다. 파일의 경우에도 이와 비슷한 과정을 통해 전이 결정이 이루어진다. 결과적으로, 접근 결정 과정은 타입간의 격리를 제공해주고, 전이 결정 과정은 여러 프로그램들이 각각의 타입을 가지고 실행될 수 있는 환경을 제공해준다.

아래의 <그림1>은 TE를 이용한 접근제어의 예로서, Email 클라이언트 프로그램이 자신에게 배정된 타입과 같은 타입을 가진 POP3 클라이언트를 실행하는 것은 허가되지만, 운영체제 커널의 이미지를 수정하는 것은 허가되지 않음을 보여준다.



<그림 1> TE를 이용한 접근제어의 예

3. 임베디드 리눅스에 TE를 사용하기 위한 설계와 경량화 방안

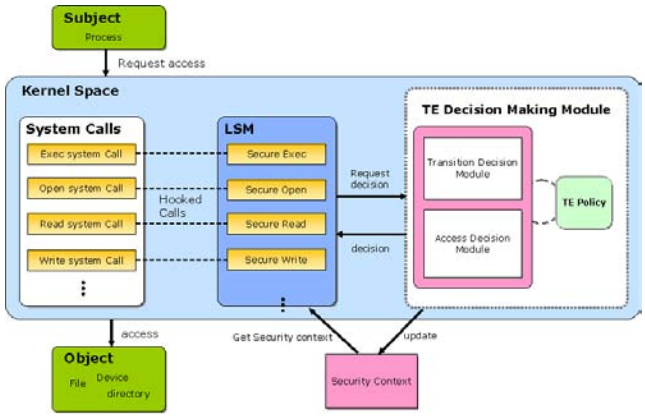
3.1 TE사용을 위한 설계

강력한 접근제어를 위해서, TE는 커널 영역의 프로세스와 Object에서 사용자 영역의 응용프로그램, 파일에 이르기까지 다양한 주·객체들에 대하여 접근제어를 할 수 있도록 설계되어야 한다. 일반적으로 리눅스의 접근제어나 보안코드들은 커널 내부에 직접 작성되거나 Loadable Kernel Module(이하 LKM) 방식을 통해 이루어졌다. 이중에서도 LKM 방식은 커널을 새로 작성할 필요가 없고, 필요에 따라 동적으로 로드/언로드 할 수 있다는 장점 때문에 많이 사용되어 왔다. 하지만, LKM을 이용한 방식은 시스템 콜이나 파일 시스템을 후킹해야만 하고, 이를 위해서는 시스템 콜 테이블 심볼이 노출되어 있어야 한다. 이러한 방법은 시스템의 성능 저하를 일으킬 수 있고, 보안적 측면에서도 좋지 않은 방법이다.

LKM의 이러한 단점으로 인해, 최근의 리눅스(커널 버전 2.6)에는 이를 보완할 수 있는 Linux Security Modules (이하 LSM) 방식이 기본적으로 채택되고 있고, 이전 버전(2.4)의 커널에도 패치를 통해 LSM을 사용할 수 있다. 실제로 임베디드 시스템에 많이 이식되어 사용되고 있는, 몬타비스타 리눅스, Embedix, Blue Cat Linux등의 임베디드 리눅스들도 최근에는 커널 2.2버전에서 2.4나 2.6버전의 커널로 옮겨오면서 LSM을 지원하고 있다[5].

LSM은 실제로 특정한 모듈을 지칭하는 것이 아니라, 일종의 프레임워크(Framework)로서 여러 가지 서로 다른 접근제어 및 보안코드들을 적재 가능한 모듈로 제작하는데 편리하도록 도와준다[6]. 시스템 콜들을 쉽게 후킹할 수 있는 인터페이스를 마련해주며, 대부분의 커널 내부의 자료구조에 대해서 보안 필드가 존재하여, 커널 내부의 어떤 위치에서든 보안코드 및 접근제어를 수행할 수 있도록 해준다. 따라서, 임베디드 리눅스에서 TE를 사용하기 위한 방법으로서, 위의 세 가지 방법을 생각해 본다면, LSM을 이용한 방법이 현재로서는 가장 안전하고 빠르면서도, 효율적인 구조를 제공해 준다.

아래의 <그림 2>는 LSM을 이용한 TE 모델을 보여준다.



<그림 2> LSM을 이용한 TE 모델

<그림 2>가 보여주는 접근제어를 간략히 설명하면 다음과 같다. Subject가 Object에 접근하려고 하면, 원래의 시스템 콜이 실행되기 전에 LSM내의 후킹 시스템 콜이 먼저 수행되며, 후킹 시스템 콜은 Subject와 Object의 타입(security context)을 얻어온 후, 이를 바탕으로 TE 접근제어 엔진(Decision Making Module)에게 접근결정을 내려줄 것을 요청한다. TE 접근제어 엔진은 TE Policy를 바탕으로 접근결정을 내린 후, 허가(Permit)나 거부(Deny)를 반환하고, 이 결과에 따라서 Subject는 Object에 접

근할 수 있거나, 접근할 수 없게 된다.

3.2 경량화 방안

TE를 경량화하기 위한 방법은 크게 두 가지로 나눌 수 있다. 첫 번째 방법은 임베디드 시스템의 특성을 고려하여, TE를 다른 접근제어 기법과 분리해서 사용하는 것이며, 두 번째 방법은 TE의 접근 결정과 전이 결정에 쓰이는 정책(Policy)을 경량화 하는 것이다.

우선, 첫 번째 방법에 대해서 생각해보자. 실제로 TE를 사용하는 PC환경이나 서버 환경의 예를 보면, TE는 단독으로 사용되기 보다는 다른 접근제어 방법과 더불어 사용되는 경우가 많다. 그 이유는 서버가 운영되는 조직의 구조적인 특징으로 인하여 TE 외에 다른 접근 제어 기법이 추가로 필요할 수 있기 때문이다. 그 예로서 NSA의 SELinux를 들 수 있는데, SELinux에서 TE는 역할 기반 접근제어 (Role Based Access Control, 이하 RBAC)[8, 9] 기법 및 다중 등급 접근제어(Multi-Level Security, 이하 MLS)[10] 기법과 함께 사용된다. 기업이나 국방부와 같은 조직에서는 TE를 RBAC이나 MLS와 함께 사용함으로써, TE의 강력한 접근제어와 조직의 구조적 특징에 맞는 접근 제어 동시에 제공받을 수 있다.

하지만, 임베디드 시스템의 경우에는 일반적으로 사용자가 소수이거나, 단일한 경우가 많고, 프로그램이나 리소스들을 등급에 따라 분류할 수 있는 구조와는 거리가 멀기 때문에, RBAC이나 MLS를 사용하는 것은 적절치 못하고, 오히려 시스템의 성능상 저하를 가져올 수 있다. 따라서, 임베디드 시스템에 불필요한, RBAC이나 MLS를 제거하고, 단순하게 TE만을 사용하거나, DAC과 TE를 결합시켜 사용하게 되면, 임베디드 리눅스는 가벼우면서도, 강력한 접근 제어를 제공할 수 있다.

두 번째 방법은 정책을 경량화 하는 방법이다. 정책을 경량화 함으로써 TE가 접근 결정 및 전이 결정시에 검색하는 공간을 줄일 수 있고, 보다 신속한 접근 제어를 수행할 수 있다. 정책은 임베디드 리눅스의 버전과 종류, 그리고 응용 프로그램들에 따라 많이 달라질 수 있으므로, 정책의 경량화를 위해서는 임베디드 리눅스의 커널 Object 및 응용 프로그램들에 대한 프로파일링(Profiling)이 선행되어야 한다. 프로파일링이 끝나면, 그 결과로 만들어진 프로파일을 이용하여 임베디드 리눅스에 최적화된 정책을 생성할 수 있게 되고, TE는 이를 이용하여 빠르고, 강력한 접근제어를 수행할 수 있다.

4. 결론

임베디드 리눅스는 오픈 소스와 높은 시스템 이용성 덕분에 많은 임베디드 시스템에 이식되어 운영되고 있다. 하지만, 현재의 임베디드 리눅스에서는 여러 보안적인 결점들이 발견되고 있어서, 이러한 결점들을 보완하기 위해 타입 강제(TE)를 소개하였다. 그리고 실제로 TE를 임베디드 리눅스에 사용하기 위한 설계와 성능의 저하를 방지하기 위한 경량화 방안을 제시했으며, 이러한 설계와 경량화를 통해 임베디드 리눅스는 보다 강력한 보안을 제공해줄 수 있을 것이라 생각한다.

타입 강제 기법이 주는 높은 보안성은 타입 사이의 강제를 위한 정책이 좌우하게 되고, 높은 성능은 정책의 경량화를 통해서 이루어진다. 따라서, 향후 임베디드 시스템에 적합한 정책을 만들고 경량화하기 위한 노력이 지속적으로 이루어 진다면 보다 높은 보안성을 얻을 수 있을 것이라 기대한다.

참고문헌

[1] 손형길, 박태규, 이금석, “리눅스 운영체제 기반의 보안 커널 구현”, 정보처리학회논문지C 제10-C권 제2호, 동국대학교 컴퓨터 공학과, 2003

[2] Peter A. Loscocco, Stephen D. Smalley, "Meeting Critical Security Objectives with Security-Enhanced Linux"

[3] Lee Badger, Daniel F. Sterne, David L. Sherman, Kenneth M. Walker, Sheila A. Haghighat, "Practical Domain and Type Enforcement for UNIX", Trusted Information Systems, Inc

[4] Serge E. Hallyn, Phil Kearns, "Domain and Type Enforcement for Linux", College of William and Mary

[5] Blue Mug, Inc. “Embedded Linux Survey”, December, 2002, http://www.bluemug.com

[6] Linux Security Modules (LSM), http://lsm.immunix.org/

[7] NSA (National Security Agency), "Security Enhanced Linux (SELinux)", http://www.nsa.gov/selinux

[8] NIST (National Institute of Standards and Technology), “Role Based Access Control”, http://csrc.nist.gov/rbac/

[9] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. 1996. Role-based access control models. IEEE Computer 29, 2(February), 38-47.

[10] Multi-Level Security, "System V/MLS Labeling and Mandatory Policy Alternatives", Proc. of USENIX-Winter '89