제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

RBAC 기반 보안 OS 에서의 권한 전이 공격

김형찬*, 이기영*, 이동익*, 김형천**, 강정민**, 이진석**
*광주과학기술원 정보통신공학과
**국가보안기술연구소
e-mail : {kimhc, kylee, dilee}@kjist.ac.kr {khche, jmkang, jinslee}etri.re.kr

Privilege Transitive Attack in RBAC based Secure OS

Hyung Chan Kim*, Ki Young Lee*, Dong Ik Lee*,
Hyoung Chun Kim**, Jung Min Kang**, Jin Seok Lee**
*Dept. of Information and Communications, Kwangju Institute of Science and Technology
**National Security Research Institute

요 약

기존의 UNIX/LINUX 시스템에서는 setuid 가 걸린 프로그램의 취약점을 공격하여 슈퍼유저(root) 권한을 획득하는 공격이 일반적이다. 본 논문에서는 RBAC 기반 보안 OS 에서도 이와 유사한 권한 전이 공격이 가능한지를 실험한다. 또한 논리적 접근통제가 강화된 보안 OS 서 권한 전이 공격에 대해 대응하는 기술에 대하여 고찰한다.

1. 서론

해킹(Hacking)기법은 빠르게 발전하고 있다. Stack Buffer Overflow[1] 공격이 소개된 이후, static 영역이나 BSS 영역에서의 overflow, double free 공격, 그리고 format string 등 다양한 기법들과 그것을 응용한 방법들이 연구되고 있으며, 실제 많이 사용하는 프로그램에 대한 공격 코드들을 인터넷에서 쉽게 접할 수 있다. 또한 최근에는 리눅스 커널 코드 분석이 용이해짐에 따라, 운영체제의 커널에 직접적인 공격을 가하는 사례도 늘어가고 있다. 커널 모듈에 대한 공격이나, rootkit 의 사용, 커널 스택 및 커널 네트웍 스택에 대한 공격 가능성, 그리고 ptrace 공격을 통한 슈퍼유저 권한 획득 등의 최근 사례들이 이를 증명한다.

이러한 공격에 대응하는 노력도 꾸준히 진행되어 왔다. 라이브러리 수준에서 공격에 취약한 함수들을 패치하는 방법이나[2], 스택 오버플로우를 막기위하여 스택 복귀 주소 옆에 canary 워드를 코딩하는 방법과[3] 복귀 주소를 텍스트 영역에 보관하여 함수 종료 시 공격에 의해 변경된 주소와 비교하는 방법[4]등 다양한 방법들이 연구되어 왔다.

사용자 프로세스 레벨의 대응 방법 이외에 커널 수준에서 보안 기능을 강화한 보안 운영체제(Secure OS, Trusted OS)도 최근 연구용 및 상업용으로 많이 개발되고 있다. 기존 UNIX 운영체제의 파일 소유자 기반 접근통제는 파일 소유자가 자신이 가진 자원을 임의적 권한 전이가 가능한 임의 접근 통제 시스템이다(Discretionary Access Control, DAC). 여러 보안 OS 프로젝트들은 중앙 집중적 관리자의 권한을 향상 시키고 등급별 접근 통제, 혹은 역할별 접근통제를 하는 강제적 접근통제(Mandatory Access Control, MAC)나 역할기반 접근통제(Role-Based Access Control, RBAC)를 구현하여 기존 UNIX 보다 엄격한 접근통제 관리 기능을 제공하고 있다. 이와 같은 논리적 구조의 접근통제 이외에 Grsecurity 와 PaX[5,6]의 경우 RBAC 과 동시에 각종 프로그램 공격을 방어하기 위한 방법들을 커널 아키텍처 수준에서 제공하고 있다.

기존의 리눅스에서 setuid(Set User ID)가 할당되어 있는 프로그램에 버그가 존재하는 경우, 프로그램을 공격하여 쉘(shell)을 실행시키면 해당 uid 의 권한을 획득하게 된다. 즉 자신의 권한이 아닌 다른 권한을 실행시킬 수 있으며, 특히 해당 uid 가 0 인 경우 슈퍼유저(root)의 권한을 가지게 된다.

RBAC 기반의 보안 운영 체제에서도 그러한 공격이 가능하다. 본 논문에서는 RBAC 을 지원하는 보안 운영체제인 SELinux 에서 프로그램을 실행시켰을 경우 특정 권한으로 전이될 수 있는 조건을 만들었다. 그리고 그 프로그램을 공격하여 DAC 기반의 UNIX 에서의 setuid 공격과 비슷한 유형의 공격이 가능함을 실험하였다. 또한 이 공격에 대한 가능한 대응 기술들에 대해서도 논의한다.

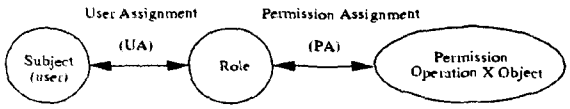

그림 1. 역할기반 접근통제

2. 관련 연구

2.1 역할기반 접근통제

RBAC[7]의 주요 개념은 주체(Subject)와 객체(Object)사이에 직접적으로 관계를 두지 않는 것에서 출발한다. 대신에, 한 시스템이나 조직에서 일의 직무(책임)를 나타내는 추상 개념인 역할(Role)이란 요소를 도입하여, 주체와 객체 사이의 관계를 설정한다[그림 1]. 이 역할을 도입함으로 RBAC 은 접근통제 관리의 용이성을 얻게 된다. 기존의 DAC(Discretionary Access Control, 임의 접근통제)이나, MAC(Mandatory Access Control, 강제 접근통제) 시스템들은 주체와 객체를 직접 연관시켜 구성되었다. 하지만, 이러한 방법들은 관리해야 할 통제 개체들이 너무 많을 경우, 관리하기가 힘들어진다. 일반적으로 규모가 큰 조직이나 대기업 환경에서 이러한 상황이 가능하다. 반면에 RBAC 은 오퍼레이션(Operation)과 접근 객체의 곱 집합으로 이루어진 권한(Permission)들을 직무에 필요한 만큼 해당 역할에 할당한다. 그리고, 관리자는 접근 주체를 직무 수행에 따른 역할에 할당하기만 하면 되기 때문에, 필요한 모든 접근 주체와의 관계를 1:1 로 두지 않아도 된다.

RBAC 은 정책 중립적인 특징을 가진다. 하지만, *RBAC 은 최소 권한의 원칙(Least Privilege)*과 *의무의 분리*(Separation of Duties)를 직접적으로 지원한다.

2.2 Security Enhanced Linux

미 NSA 에서 만든 *SELinux*[8]는 중앙집중적인 접근통제 정책을 Linux 시스템에 유연하게 구성할 수 있도록 Flask Architecture 를 적용한 보안 OS 이다. 접근통제 모델로 RBAC 과 Type Enforcement[9]를 지원한다. Flask Architecture 는 RBAC 을 포함한 일반적인 강제접근통제를 지원하기 위한 구조로, 다양한 보안 정책을 제공하기 위해서 논리로 구성되어 있는 보안 정책과 실제 정책에 대한 접근통제를 적용하는 *Enforcement facility* 를 명확하게 분리하였다. 리눅스 커널의 파일 시스템, 네트웍 스택 등의 자원접근이 이루어지는 커널 subsystem 에 Object Manager 를 두어 AEF(Access Enforcement Facility)역할을 하게 하고, Security Server 는 설정된 정책에 따라 Object Manager 에서 요청된 접근에 대한 판단을 하는 ADF(Access Decision Facility)역할을 한다[그림 2]. Security Server 는 커널 내부의 하나의 subsystem 으로 구성되어 있으며, 접근통제로 인한 성능의 감소(penalty)를 보완하기 위하여 Access Vector Cache 를 두어 여러 번 일어나는 접근에 대한 성능향상을 고려하였다. 이러한 정책적용 부분과 정책판단 부분의 분리는 운용 도중의 정책 변경을 지원한다.

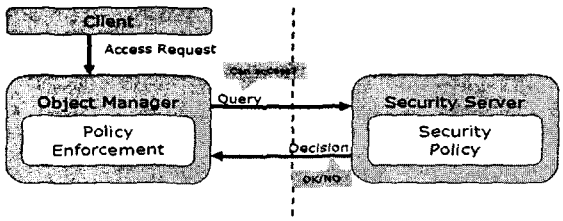

그림 2. SELinux 의 Security Server 와 Object Manager

3. 기존 UNIX/LINUX 에서의 setuid 프로그램 공격

3.1 Setuid(Set User ID)

UNIX/LINUX 운영체제에서 setuid program 은 해당 프로그램이 실행될 때, 실행파일의 소유자 권한을 주어 특정 권한의 작업을 하고자 할 때 사용된다. 예를 들어 /bin/passwd 란 프로그램은 root(uid=0) 소유의 /etc/passwd 파일을 변경해야 하기 때문에 setuid flag 가 켜져 있다. 그러므로, */bin/passwd* 프로그램은 실행하는 동안(process) root 의 권한으로 /etc/passwd 파일을 다룰 수 있다.

Linux 에서는 커널 프로세스 구조체에 euid 와 fsuid 가 실제 credential 의 역할을 하게 된다. fsuid 는 파일에 대한 권한 체크를 할 때 사용되고, euid 는 그 외 나머지 권한에 대한 체크를 할 때 비교 대상이 된다.

이러한 메커니즘은 권한의 전이(transition)할 수 있는 기능을 제공하는 것이므로 악용의 대상이 될 수 있다. 즉, setuid 프로그램과 다른 uid 를 가진 사용자가 해당 프로그램의 실행권한이 있는 경우, 실행 중에는 setuid 프로그램의 uid 를 가지게 된다. 만약 프로그램에 stack/heap overflow 나 format string 등과 같은 버그가 존재할 경우 공격자는 여러 가지 공격기법을 사용하여 프로세스가 종료하기 전 쉘(shell)을 실행시킨다. 공격이 성공한다면, setuid 프로그램 소유자의 쉘을 획득하게 되며, 해당 권한으로 악의적인 일을 할 수 있게 된다.

3.3 Setuid 프로그램의 공격 실험

[그림 3]은 웹 서비스(apache) 관리자 권한이 소유하고 있는 실행파일을 임의적으로 만들어 간단한 stack overflow 공격을 수행하는 실험이다. 결과에서는 일반 유저(kimhc)가 setuid 가 세팅 된(apacheowned_exec)실행파일을 공격하여 웹 서비스 관리자의 권한의 쉘을 실행시키고 있다.

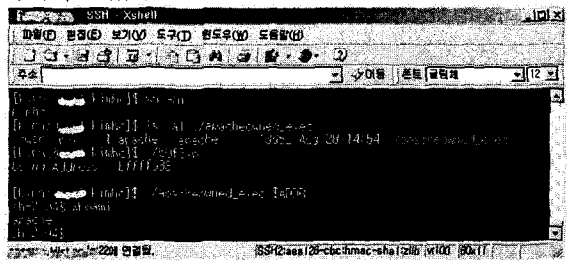

그림 3. 일반 Linux 에서 setuid 프로그램 공격

4. SELinux 에서의 특정 권한을 얻는 프로그램 공격

4.1 SELinux 에서의 권한 전이

접근 통제 모델에 근거한 보안 OS 는 논리적인 보안 설정 오류로 인한 공격이 가능하다. SELinux 의 경우 리눅스 커널 아키텍처에 의존적인 방어 메커니즘을 사용하는 것이 아니라, RBAC 과 TE 라는 논리적 구조에 의한 접근통제를 함으로써 시스템의 보안성을 향상시킨다. 만약 해당 보안 OS 에서 제공하는 논리적 접근통제 모델이 프로그램 실행시간(runtime)에 권한의 전이를 지원한다면, 기존의 DAC 기반 UNIX 시스템에서 uid 의 전이로 인한 권한 전이 공격과 유사한 형태의 공격이 가능할 것이다.

SELinux 의 RBAC 모델에서 역할의 전이(role transition)는 현재 newrole 이란 명령어로만 가능하며, 접근통제 설정에서 역할 전이 keyword 는 사라졌다(deprecated). 하지만 TE(Type) Transition 이 가능하기 때문에, 프로그램 실행시간에 권한 전이가 가능하다. TE Transition 이란 새로 생성되는 접근 주체 및 객체에 대하여 새로운 labeling 을 하는 것이다. 이것은 프로세스가 새로 생성되거나 파일이 생성될 때, 새로 생성된 오브젝트에 대하여 어떤 label 을 줄 것인지 결정하는 메커니즘이다(labeling decision mechanism). SELinux 에서는 접근통제를 함에 있어, 주체 및 객체의 sid(security identifier)를 접근 판단 조건 인자로 사용한다. 즉, 주체의 sid 가 type transition 에 의해 변경되었을 경우, 주체는 변경된 sid 로써 행할 수 있는 권한을 가지게 된다. 예를 들어,

type_transition sshd_t shell_exec_t:process user_t;

라는 type 전이의 경우, SSH 가 어떤 사용자의 login 에 의해 쉘을 실행시킬 때 user_t 라는 타입으로 전이시켜준다. 즉, 보안 컨텍스트가 SSH 프로그램에서 일반 사용자 쉘 프로세스로 바뀜에 따라 식별자인 sid 도 바뀌어야 하는 것이다.

Sid 의 변경은 논리적으로 설정된 접근 통제 정책에 의해 판단된다. 따라서 보안 관리자가 접근 통제 설정의 미숙이나, 설정의 복잡성으로 인하여 의도하지 않은 type transition 이 가능할 수도 있다.

4.2 권한 전이 공격 실험

DAC 기반 UNIX 에서 setuid 가 설정된 것처럼, RBAC 기반의 SELinux 에 다음과 같이 Type 전이를 통하여 다른 권한을 사용할 수 있도록 임의로 설정하였다.

allow user_t httpd_admin_t:process transition;
type_transition user_t www_exec_t:process httpd_admin_t;

위의 설정에 의하여 일반 사용자의 도메인(user_t)은 www_exec_t 로 레이블된 프로그램의 실행에 의하여 웹서비스 관리 권한인 httpd_admin_t 도메인 타입으로 전이가 가능하다. 즉, www_exec_t 프로그램을 실행시키는 동안 해당 프로세스는 웹서비스 관리 권한을 갖는다.

[그림 4]에서 avc_enforcing 명령의 결과인 enforcing 은 현재 SELinux 에서 기본적인 DAC 이외의 RBAC/TE 강제접근통제가 활성화되어 있음을 말해준다. 버그가 있는 www_exec_t 로 레이블된 프로그램을 실행키는 중간에 프로그램을 공격하여 쉘을 실행시키자, 그 쉘 프로세스의 권한은 root:user_r:httpd_admin_t 의 웹 서비스 관리 권한의 context 를 갖게 되었다. (SELinux 에서 root 의 의미는 일반유저와 같다. 즉, [그림 4]에서 root 는 DAC 에서의 root 권한일 뿐, 일반유저 user_r 역할에 할당되어 있어 시스템의 모든 권한을 남용할 수 없다.)

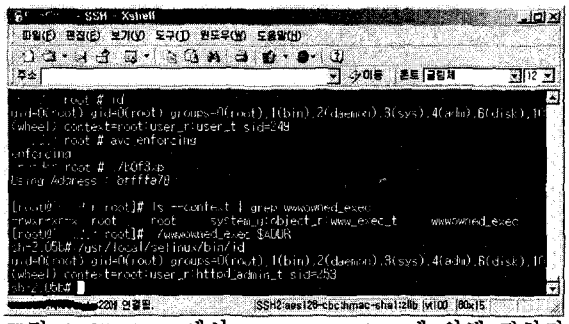

그림 4. SELinux 에서 Type transition 에 의해 권한전이가 가능한 프로그램 공격

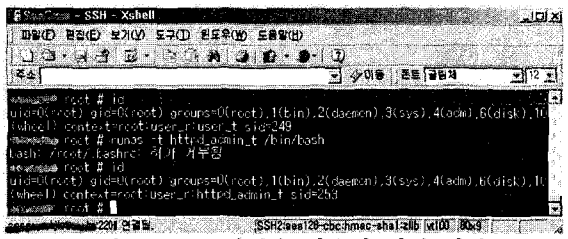

그림 5. runas 명령을 이용한 권한 전이

SELinux 에서는 [그림 5]와 같이 runas 란 명령어를 지원한다. 이는 SELinux 의 보안 API 인 execve_secure 를 이용한다. 현재 컨텍스트에서 다른 컨텍스트로의 전이가 가능하게 설정된 경우, 이 함수를 통하여 곧바로 권한 전이가 가능하다. 논리적인 접근통제의 잘못 설정된 부분을 공격자가 파악한 경우, 따로 프로그램을 공격할 필요 없이 이 명령어를 통하여 권한 전이를 할 수 있는 것이다. 실험사례[13]에서 공격자는 웹 관리자 권한을 얻기 위해 웹 서버와 보안 OS 의 논리적 설정 상태를 list_sids 명령과 실제 권한 이용 등을 통해 분석을 하는 양상을 보여주었다.

5. 대응방법에 관한 고찰

3, 4 절에서 각각 기존 DAC 시스템에서 setuid 프로그램 공격을 통한 권한 전이와, RBAC 이 적용된 SELinux 에서 type 전이가 가능함에 따른 권한 전이 공격이 가능함을 살펴보았다.

보안 운영체제에서 이러한 공격을 방어하는 대응 노력에 대하여 두 가지 관점에서 살펴볼 수 있다. 첫 번째는 아키텍처 수준에서 프로그램 공격을 차단하는 것이다. PaX[6]에서 스택 위치를 Random 하게 하거나, non-executable stack(open wall)을 적용하여 방지하는 것처럼, 개별적인 프로그램 공격 유형에 대하여 분석하

고 이를 방어하기 위한 메커니즘을 구현 및 적용하는 것이다.

하지만 이러한 각각의 공격에 대한 방어장치는 이미 다양한 우회기법들이 존재한다. Stack guard, Stack shield, PaX 의 몇몇 방어기제들에 대한 우회기법들이 발표되거나 연구되고 있다. 이렇게 특정한 아키텍처 수준의 메커니즘에 대한 공격에 대해서는, 방어기법들이 나오고 이에 대한 우회기법들이 나오는 순환 프로세스가 지속될 것이다.

이에 비해 BLP 나 RBAC 과 같이 논리적인 접근통제가 보안 OS 에 적용되는 경우는, 특정한 아키텍처에 의존한 공격에 대응할 수 있는 장점이 있다. 예를 들어, RBAC 에서 어떤 사용자의 프로그램이 공격을 받더라도 그 파급효과가 논리적인 관점에서 해당 역할에 한정된다. 하지만, 관리 객체가 상당히 많고 접근통제 설정이 복잡할 경우에 잘못된 설정으로 인한 논리적 결함이 발생할 수 있다. 아키텍처 수준의 공격방어도 여전히 유효하다. 하지만, 논리적인 수준에서 보안 정책을 검증하는 체계가 이러한 유형의 보안 OS 에 필요하다. 보안 관리자가 일일이 논리적인 설정 사항을 체크하는 것은 매우 어렵다. 따라서, 접근통제 모델에 대한 정형검증기법 체계를 확립하고, 자동화되고 시각적인 접근통제 설정을 통해 시스템의 논리적 접근 흐름을 제어하는 노력이 필요하다. 접근통제의 정형기법 적용 예로써, Alloy[10]나 Z[11]언어를 통하여 RBAC 을 검증하려는 시도가 되고 있다. DTOS[12]의 경우는 Z 를 이용하여 보안 OS 를 명세한 대표적인 사례이다. 하지만 실제 보안 OS 에 특화되어 접근 통제 설정 툴과 연동된 예는 없다. 또한 SELinux 의 경우는 GUI 기반의 보안 설정 도구가 있지만, 각 보안 컨텍스트에 대한 흐름을 정형적으로 분석하여 시각적으로 보여주지 못하기 때문에, 여전히 복잡한 설정에 대한 부담을 관리자가 가지고 있다. 따라서 정형검증과 함께 적절하게 시각화되어 보안 관리자가 논리적 접근 흐름을 파악할 수 있도록 해주는 보안 OS 분석 도구의 개발이 필요하다.

6. 결론

본 논문에서는 기존 UNIX/LINUX 시스템에서 프로그램 버그와 권한 전이를 통한 공격과 RBAC 이 적용된 보안 OS 인 SELinux 에서의 권한 전이 공격에 대하여 살펴보았다. 보안 OS 에서도 기존 LINUX 시스템의 setuid 와 같이 권한 전이가 논리적으로 가능하다면, 잘못된 접근 통제 설정에 의해 공격이 가능하다. 따라서, 보안 OS 에도 정형검증 기법과 시각화된 접근 통제 흐름 제어가 가능한 보안 도구가 개발되어 보안 관리 부담을 줄여야 한다.

참고문헌

[1] Aleph One, "Smashing The Stack For Fun And Profit," Phrack Vol.7 Issue. 49, File 14 of 16, 1996

[2] http://www.research.avayalabs.com/project/libsafe/

[3] http://www.immunix.org/stackguard.html

[4] http://www.angelfire.com/sk/stackshield/

[5] http://www.grsecurity.net/

[6] http://pageexec.virtualave.net/

[7] D.F. Ferraiolo, J. Cugini, D.R. Kuhn "Role Based Access Control: Features and Motivations", Proc. of Annual Computer Security Applications Conference, IEEE Computer Sociery Press, 1995.

[8] Peter Loscocco, and Stephen Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating System," In Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX'01),Jun.2001. (http://www.nsa.gov/selinux/index.html)

[9] L. Badger, D.F Sterne, D.L Sherman, K.M Walker, and S.A. Haghighat, "Practical Domain and Type Enforcement for UNIX," In Proceedings IEEE Symposium on Security and Privacy, pp. 66-77, May. 1995.

[10] John Zao, Hoetech Wee, Jonathan Chu, Daniel Jackson, "RBAC Schema Verification Using Lightweight Formal Model and Constraint Analysis," MIT Software Design Group Case Study, December 2002

[11] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn and Ramaswamy Chandramouli, "Proposed NIST Standard for Role-Based Access Control Model," *ACM Transactions on Information and Systems* Security, Vol. 4, No. 3, pp. 224-274, Aug. 2001.

[12] "DTOS Formal Security Policy Model," Technical Note of Secure Computing Corporation, 1996

[13] K-JIST(광주과학기술원) Hacking Festival 2003 Workshop 자료집, http://seecure.kjist.ac.kr/~KHF2003/