커널 내부 권한 분리 기술 연구 동향

신채원[1], 강하영[1], 송수현[2], 권동현†
[1]부산대학교 정보융합공학과 석사과정
[2]부산대학교 정보융합공학과 박사과정
†부산대학교 정보컴퓨터공학부 부교수, 교신저자

sinchaewon@puasn.ac.kr, rkdgkdud12345@pusan.ac.kr, sshpnu@pusan.ac.kr, kwondh@pusan.ac.kr

A Survey of Research Trends on Intra-Kernel Privilege Separation Techniques

Chae-Won Shin[1], Ha-Young Kang[1], Su-Hyeon Song[2], Dong-Hyun Kwon†
[1]Dept. of Information Convergence Engineering, Pusan National University
[2]Dept. of Information Convergence Engineering, Pusan National University
†Dept. of Computer Science and Engineering, Pusan National University, Corresponding Author

요 약

운영체제 커널은 현대 컴퓨팅의 TCB 핵심으로서 보안상 매우 중요하지만, 모놀리식 구조와 안전하지 않은 언어 기반으로 인해 취약점이 지속적으로 유입되고 있다. 이에 따라 커널 내부 권한 분리(intra-kernel privilege separation)가 커널 보안의 새로운 방향으로 부상했다. 본 논문은 최근 제안된 대표적 연구를 아키텍처 축(Intel, Arm, RISC-V)과 설계 목표 축(게이트 무결성, 상호 불신 모델 및 TCB 축소, 도메인 수 확장, 데이터/제어 동시 보호, 성능 최적화)으로 비교한다. 먼저 기존 접근의 공통 한계를 정리한다. 이어 BULKHEAD, NANOZONE, GENESIS 사례를 통해, 각 한계를 해결하는 핵심 아이디어와 트레이드오프를 분석한다. 본 분석은 커널 내부 권한 분리의 설계 공간을 체계화하고, 실용 배치를 위한 요건을 제시한다. 마지막으로 향후 연구 과제를 논의한다.

1. 서론

운영체제 커널은 시스템 소프트웨어의 초석이며, 다양한 사용자 프로그램과 하드웨어 장치를 지원하면서 코드 규모와 기능 범위가 지속적으로 확대되고 있다. 커널은 TCB(Trusted Computing Base)의 핵심이므로 보안상 중요도가 특히 높다. 그러나 커널은 여전히 모놀리식 구조와 메모리 안전성이 낮은 언어 기반에 의존하고 있어 취약점이 상시 유입된다. 실제로 널리 사용되는 리눅스 커널의 CVE 보고는 2013 년 대비 2023 년에 179% 증가했고, 최근 10 년 누적 2,814 건에 이른다[1]. 단일 취약점이라도 커널 전역 주소공간과 특권 명령 실행을 통해 전체 시스템으로 확산될 수 있어 위험은 구조적으로 증폭된다.

이에 따라, 최근 커널 내부 권한 분리가 커널 설계 연구의 새로운 방향으로 부상했다. 이 접근은 모놀리식 커널을 내부 커널과 외부 커널로 분리한다. 또한 외부 커널에게서 MMU/페이지 테이블 등 보안적으로 중요한 자원에 대한 제어 권한을 박탈하고 해당 권한을 내부 커널에만 부여한다. 결과적으로, 외부 커널은 특권 동작을 수행할 수 없도록 금지되며, 외부 커널에서 필요한 특권 동작은 게이트를 통해 내부 커널로 전달되어 수행된다. 두 구성요소가 동일한 하드웨어 특권 레벨에서 실행됨에도 비대칭 접근 권한이 소프트웨어 메커니즘으로 강제된다.

본 논문은 이러한 내부 권한 분리의 최신 연구를 체계적으로 정리/비교하고, 아키텍처별 설계 선택과 한계를 분석한다.

2. 커널 내부 권한 분리 연구 개요

커널 내부 권한 분리 기법이 점점 발전되어 오면서, 완전한 보호를 위해 해결해야 할 문제들과 권한 분리 방법의 기틀이 확립되어 왔다. 그러나 커널 내부 권한 분리에 관련한 이전 연구에는 공통적으로 넘어야 하는 과제가 몇 가지 있다.

첫째, 특권 전환을 수행하는 오버헤드가 크다. 페이지테이블 기반 격리는 전환 때 CR3/TTBR/SATP 등 상위 베이스 레지스터 갱신이 필요한데, 이때 PCID 같은 최적화가 있어도 전환 자체는 특권 경로를 이용

하기 때문에 높은 오버헤드를 동반한다. 이를 대체하기 위해 도입된 Intel MPK 등의 하드웨어 기법은 특권 전환 자체는 빠르게 수행되나, key 를 모두 소진하여 새로운 도메인에 할당할 key 가 없을 때에는 key 를 재사용하기 위해 이미 할당되어 있는 key 를 해제하고 새로운 도메인에 다시 할당하는 과정을 거쳐야 하는데, 여기서 높은 오버헤드가 발생한다[2,3].

둘째, 도메인 수가 제한되어 있다. 대부분의 커널 내부 권한 분리 연구에서는 앞서 말한 높은 오버헤드를 피하면서 높은 보안성과 성능을 보장하기 위해 하드웨어에 기반한 메모리 접근 제어 기법을 이용하지만, 하드웨어 필드는 대부분 한정되어 있으므로 구조적으로 도메인 개수가 제한된다. 예를 들어, Intel MPK 는 4 비트의 pkey 로 도메인을 구분하여 최대 16 개의 도메인만을 사용할 수 있다[2]. 또한 Arm POE 는 8 개의 key 를 이용하는데, 이중 기본값을 나타내는 1 개를 빼서 최대 7 개의 도메인만을 사용할 수 있다[3].

셋째, 방어 기법 자체를 우회할 수도 있다. 특권 전환을 수행하는 스위치를 악용하거나, 화이트리스트를 우회하는 방식이다. 프로세스 혹은 커널 내부에서 세부 도메인의 접근 제어를 수행하는 경우에는 특권 전환을 수행하는 트램폴린/게이트가 존재한다. 이때 공격자가 리턴 주소나 함수 포인터 등 제어데이터를 오염해 화이트리스트에 포함되어 있는 콜 사이트로 점프하면 합법적 진입처럼 보여 보안 정책에 탐지되지 않고 전환이 남용될 수 있다. 특히 커널 격리에서 단방향 격리만을 수행하면(즉 코어 커널을 신뢰하는 설계에서는), 악성 입력으로 코어가 특권 명령을 대리 실행하는 confused-deputy 문제가 열리는데, 이는 Hardy 가 고전적으로 정식화한 권한 오용 현상이다[4].

이를 바탕으로, 최신 커널 내부 권한 분리 연구들은 공통적으로 다음과 같은 설계 목표를 가진다. (1) OS/코어를 신뢰경계 밖으로 밀어내어 TCB 를 축소할 것, (2) 권한 도메인 수를 확장할 것, (3) 안전한 권한 전환 게이트를 설계하기 위하여 원자성/결정성/배타성을 보장할 것이다. 이러한 공통적인 설계 목표를 바탕으로 하여, 본 논문에서는 Intel, Arm, RISC-V 의 세 가지 아키텍처 상에서 커널 내부 권한 분리를 수행한 최신 연구들을 소개한다.

3. BULKHEAD

BULKHEAD[5]는 Intel x86 아키텍처를 대상으로 PKS 기반 커널 분할을 적용하여 앞서 정의한 문제들을 해결한다. 그림 1 은 BULKHEAD 의 디자인을 나타낸다. 설계의 출발점은 코어 커널조차 신뢰하지 않는 양방향 격리로, Intel PKS(Protection Keys for Supervisor)와 PKRS 권한 레지스터를 이용해 각 도메인의 메모리 접근을 하드웨어 수준에서 검증한다. PKRS 는 각 도메인에 부여된 pkey 의 읽기·쓰기 권한을 스레드/코어 단위로 즉시 제어하는 레지스터이며, 스위치 게이트(SGT)는 허용된 교차-도메인 전환의 유일 경로로서 전환 대상·권한·스택 정책을 사전에 고정·검증한다. 도메인 간 교차 호출은 오직 SGT 를 통해서만 허용되며, 게이트가 보유한 메타데이터(소스/타깃 도메인, 요구 PKRS 권한, 전용 스택 등)를 근거로 전환을 수행해 원자성/결정성/배타성을 보장함으로써 인터페이스 악용을 차단한다.

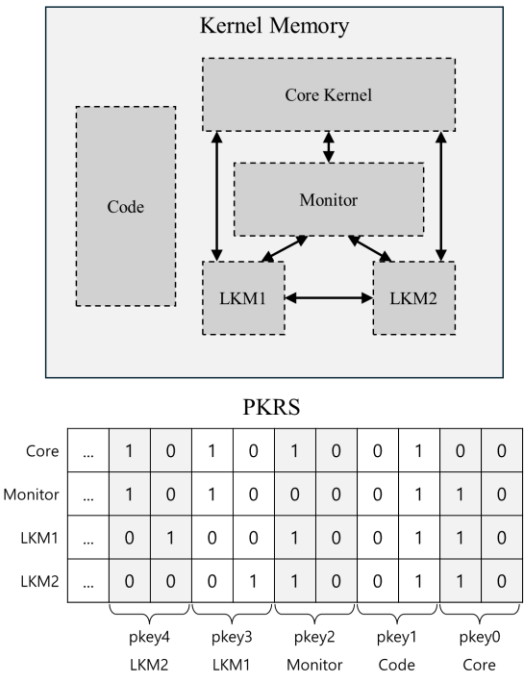

그림 1 Design of BULKHEAD

더불어 PKS 의 키가 16 개로 제한되는 구조적 한계를 넘기 위해 locality 에 기반한 2 단계 도메인 분리를 도입한다. 즉, 하나의 주소 공간 내부에서는 PKS 로 세분화하고, ASID(주소 공간 식별자)를 이용해 주소 공간 자체를 전환함으로써 pkey 를 재사용하여 사실상 무한대에 가까운 도메인 수를 지원한다. 이때 주소 공간 전환은 저수준 페이지 테이블만 교체하도록 구성하여 전환 비용을 낮춘다.

마지막으로 데이터 흐름 보호와 제어 흐름 보호를 결합한다. 각 도메인의 데이터는 전용 pkey 로 무결성을 보장하고, 모든 커널 코드 페이지를 실행 전용(Execute-only)으로 표기해 코드 재사용 공격을 어렵게 하며, 이종 도메인 간 호출은 게이트/프라이빗 스택을 통해 처리하여 제어 무결성을 강화한다.

4. NANOZONE

NANOZONE[6]은 Armv9-A 아키텍처를 대상으로 PIE/POE/PAS 등의 하드웨어 기능을 이용하여 앞서

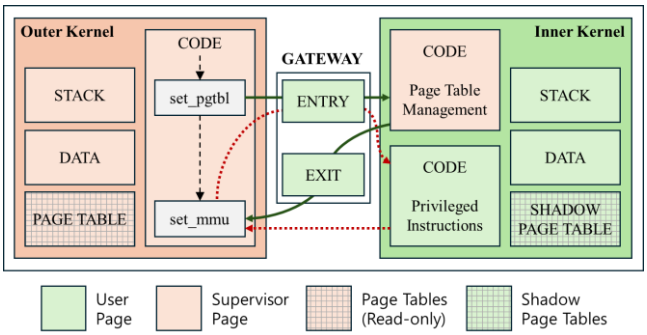

그림 2 Architecture of GENESIS

정의한 문제들을 해결한다. NANOZONE 은 사용자 수준의 POE 7 개와 커널 수준의 PIE 4 개를 조합해 총 28 개의 권한 도메인을 구성하고, Arm CCA 의 GPT/GPC3 를 이용해 물리 메모리를 복수의 PAS 로 분할함으로써 이 28 개 도메인 세트를 재사용한다. 이와 같은 ⟨POE, PIE, PAS⟩의 적층적 결합을 통해 NANOZONE 은 사실상 무한대의 도메인을 운용할 수 있다.

런타임 오버헤드를 최소화하기 위해 NANOZONE 은 POE · PIE · PAS 전환의 상대 비용을 고려해 도메인 할당과 권한 전환 경로를 최적화한다. 구체적으로, POE 전환은 L1 캐시 수준에서 처리되어 가장 빠르고, PIE 전환과 PAS 전환은 각각 L2 · L3 수준에서 처리되어 상대적으로 느리다. 이에 따라 NANOZONE 은 POE 전환의 비중을 최대화(빈번한 전환을 L1 로 수렴)하고 PAS 전환의 발생을 최소화하도록 도메인 배치를 설계한다. 그 결과 전체 전환의 96.7%를 L1 에서 처리하여, 평균 권한 전환 오버헤드를 기존의 라운드로빈 기반 도메인 할당 대비 4.87% 수준으로 낮춘다.

한편, 전환 경로(트램폴린/게이트)의 스위치 남용을 방지하기 위해 NANOZONE 은 Armv9.4-A 의 GCS(Guarded Control Stack)를 활용해 리턴 주소를 보호하고, 모든 함수 포인터에 대해 PIM 기반의 경량 CPI 를 LLVM 패스/바이너리 스캔으로 삽입해 백업/검증을 수행한다. 이 결합은 트램폴린 호출체의 화이트리스트를 우회하는 공격을 포함하여 confused-deputy 와 ROP 계열 남용을 사용자 모드만으로 차단한다.

5. GENESIS

Genesis[7]는 특정 하드웨어 기능에 의존하여 아키텍처에 종속되던 기존 구현의 한계를 해소하기 위해, 여러 ISA 에 공통으로 존재하는 기능을 활용하는 범용적 커널 내부 특권 분리 기법을 제안한다. 핵심은 다양한 ISA 에 구현되어 있는 Privileged Access Restriction(PAR)을 재해석하여 커널 내부 접근 자체를 제한하는 것이다. PAR 은 x86 의 SMAP, Arm 의 PAN, RISC-V 의 SUM 등으로 구현되어 있어, 제안 기법을 모든 주요 아키텍처에 적용할 수 있다. 이 모델에서 외부 커널은 사용자 메모리에 접근할 수 없고, 내부 커널만 게이트를 경유해 페이지 테이블 갱신이나 CRx/MSR 설정과 같은 특권 명령을 일시적으로 수행한다.

구체적으로, Genesis 는 실제 페이지 테이블을 읽기 전용으로 두고, 동일한 물리 페이지를 사용자 페이지 접근 권한으로 새도 매핑한 뒤 PAR 로 통제함으로써 내부 커널만 수정 가능하게 만든다. 외부 커널은 항상 특권 모드에서 실행되기 때문에, PAR 이 활성화된 동안에는 사용자 페이지로 매핑된 페이지 테이블에 접근할 수 없다. 페이지 테이블 접근이 필요할 경우에는 PAR 을 해제하고 내부 커널로 진입하여 안전하게 작업을 수행한 뒤, 외부 커널로 복귀 시 PAR 을 재설정함으로써 보안성을 유지한다. 또한 외부 커널에 남아 있는 특권 명령은 컴파일러 자동 계측을 통해 트랩으로 전환되며, 해당 연산은 내부 커널이 대행하도록 강제된다.

아울러 Genesis 는 안전한 도메인 전환을 보장한다. x86 의 경우 PAR 구현인 SMAP 을 제어하는 stac/clac 명령이 레지스터 독립적이고 결정적이기 때문에, 추가 검증 루프 없이 짧은 시퀀스만으로 원자성/결정성/배타성을 만족하는 전환을 달성한다. 인터럽트 경로 역시 내부 커널이 SMAP 을 재활성화한 이후 처리하도록 구성되어, 전환 중 인터럽트를 계기로 외부 커널 코드가 실행되는 시도를 차단한다. 더 나아가 외부 커널 코드에서 특권 명령 시퀀스를 제거하여, 외부 경로에서의 직접 특권 실행 가능성을 사전에 봉쇄한다.

6. 결론 및 고찰

본 논문은 커널 내부 권한 분리의 최근 흐름을 종합적으로 검토하고, 서로 다른 아키텍처 위에서 제안된 구현들이 공유하는 핵심 설계 목표와 반복적으로 드러난 공통 한계를 체계적으로 정리하였다.

분석 결과, 현 단계의 연구는 (1) OS/코어를 신뢰경계 밖으로 밀어내어 최소 신뢰 컴포넌트(모니터/내부 커널) 중심으로 만든 TCB 축소, (2) 제한된 하드웨어 인덱스/키 폭을 우회하는 도메인 확장, (3) 전환 경로를 단일화하고 원자성/결정성/배타성을 확보하는 게이트 무결성이라는 공통 분모 위에서, 각 아키텍처가 제공하는 상이한 하드웨어 기능을 목적 지향적으로 조합한다는 점을 확인하였다.

동시에, 공통 한계 역시 명확하다. 첫째, 아키텍처 의존성이 강한 설계 요소가 호환성을 제약한다. 권한

계층을 구현하는 하드웨어 프리미티브는 기능/지연/오류 처리 의미론이 서로 달라, 동일한 보안 불변식(원자성/결정성/배타성, 데이터/제어 무결성)을 각 ISA 에서 동등하게 충족시키기 어렵다. 이로 인해 게이트 시퀀스, 인터럽트 처리, TLB 일관성 유지가 플랫폼별 분기를 전제로 구현되며, 재사용성이 떨어진다. 둘째, 평가 및 검증의 상호 비교가 불가능하다는 점이다. workload 특성이 시스템/플랫폼에 따라 달라, 결과 수치가 재현 및 비교되기 어렵고, 포팅 시 성능 회귀(regression)를 조기에 탐지 및 격리하기도 어렵다.

이러한 진단을 바탕으로, 향후 과제를 다음과 같이 구체화한다.

(1) 아키텍처 간 표준화와 이식성 강화: 권한 계층(예: L1/L2/L3)과 전환 게이트의 추상 인터페이스를 표준화하여, ISA 별 프리미티브 차이를 어댑터 층에서 흡수하고 상위 정책/도구 체인은 공통화한다. 특히 pkey/overlay/indirection/PAS 의 상호 대응 관계와 전환 불변식(스택 상태, 인터럽트 마스킹, 권한 레지스터 세트)을 명세 수준에서 통일할 필요가 있다.

(2) 컴파일러/바이너리 재작성의 자동화: 트램폴린 삽입, 특권 명령 제거/대행, CPI 계측을 언어에 무관하고 링크 단계 독립으로 보장하는 파이프라인을 마련한다.

커널 내부 권한 분리는 게이트 무결성, 도메인 확장, TCB 축소, 비특권 중심 최적화라는 설계 원칙을 중심으로 실용적 성숙 단계에 진입하고 있으며, 다음 단계의 연구/개발은 표준화된 추상화 및 자동화된 도구 체인, 현실적 검증, 형식적 보증을 축으로 수렴되어야 한다. 이러한 로드맵이 정립될 때, 본 접근은 모놀리식 커널의 구조적 위험을 체계적이고 점진적으로 줄이는 핵심 보안 아키텍처로 자리매김할 것이다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 융합보안핵심인재양성사업의 연구 결과로 수행되었음 (RS-2022-II221201)

참고문헌

[1] SecurityScorecard, “Linux kernel vulnerabilities,” Apr. 2024. [Online]. Available: https://www.cvedetails.com/product/47/Linux-Linux-Kernel.html?vendor_id=33

[2] Intel Corporation, Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3: System Programming Guide, Santa Clara, CA, Intel, 2024

[3] Arm Limited, Arm® A-profile Architecture Reference Manual (DDI 0601): Permission Indirection and Permission Overlay Extensions, Cambridge, UK, Arm, 2024–2025

[4] Hardy, N., The Confused Deputy (or Why Capabilities Might Have Been Invented), ACM SIGOPS Operating Systems Review, 22(4), pp. 36–38, 1988

[5] Yinggang Guo, Weiheng Bai, Kangjie Lu, and others. BULKHEAD: Secure, Scalable, and Efficient Kernel Compartmentalization with PKS. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security ’24), Philadelphia, USA, 2024, pp. 1–18

[6] Shiqi Liu, Yongpeng Gao, Mingyang Zhang, and Jie Wang. NANOZONE: Scalable, Efficient, and Secure Memory Protection for Arm CCA. In Proceedings of the 34th USENIX Security Symposium (USENIX Security ’25), Seattle, USA, 2025, pp. 1–16

[7] Seongman Lee, Seoye Kim, Chihyun Song, Byeongsu Woo, Eunyeong Ahn, Junsu Lee, Yeongjin Jang, Jinsoo Jang, Hojoon Lee, and Brent Byunghoon Kang. GENESIS: A Generalizable, Efficient, and Secure Intra-kernel Privilege Separation. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC ’24), Avila, Spain, 2024, pp. 1365–1375