macOS 커널 디버깅을 위한 심볼 이식 기법 연구

진규정[1], 고대현[2], 양재원[3], 오현규[4], 이건하[5], 이동하[6]

[1,2,3,4,5,6]차세대 보안 리더 양성 프로그램

wlsrbwjd643@kangwon.ac.kr, 4ncienth@gmail.com, yongsek01@ync.ac.kr,
22011755@sju.ac.kr, w1nsom3gna@korea.ac.kr, p@sswd.pw

Symbol Transplanation for macOS Kernel Debugging

Gyujeong Jin[1], Daehyeon Ko[2], Jaewon Yang[3], Hyungyu Oh[4],
Geonha Lee[5], Dongha Lee[6],

[1,2,3,4,5,6]Best of the Best

요 약

macOS release 커널은 심볼 제거와 CFI·PAC/BTI·최적화로 인해 정적 분석·디버깅이 어렵고, development 커널의 풍부한 심볼·DWARF도 release 빌드에서 곧바로 활용되기 어렵다. 본 연구는 이러한 간극을 메우기 위해, 동일 빌드의 development·release 커널을 병치하여 MRS 중심 흐름과 그 주변의 비트필드 해석, 전역 특성 비트 갱신, 디바이스 트리/부트 인자 질의, 프롤로그·에필로그/간접 분기 보호(PAC/BTI)와 같은 행동 단서를 토큰화·정규화한 앵커 서명으로 Dev to Release 매핑을 수행하는 심볼 이식 프레임워크를 제안한다.

1. 서론

현행 macOS release 커널 바이너리는 심볼이 제거되고, CFI 등이 적용되어 정적 분석과 디버깅이 구조적으로 어려운 실정이다. 함수 경계가 흐려지고, 호출 그래프가 축약 및 재배열되어 전통적인 시그니처 매칭으로는 대규모 분석을 재현하기 어렵다. 반면 Apple KDK(Kernel Development Kit)가 제공하는 development 커널은 풍부한 심볼과 DWARF 정보가 포함되어 있어 내부 구조를 파악하기 쉽지만, 그 자체로는 release 빌드 환경의 동등한 디버깅으로 이어지지 못하는 한계가 있다.

이러한 한계를 해결하기 위해, 실행 추적을 통해 인터페이스와 의존성을 자동 복원하고 점진적으로 정제하는 방법이 제안되었다[1]. 또한, 안정적인 앵커를 세워 탐색공간을 크게 줄이는 마이크로아키텍처 리버스 엔지니어링 연구가 제시되었다[2].

본 연구는 이러한 진전을 발판으로 macOS 15 환경의 하이퍼바이저 관련 함수를 case study로 하여, 동일 버전의 development, release 빌드를 쌍으로 비교한다. 이를 통해 development 빌드의 심볼 및 로직을 release 빌드의 함수에 체계적으로 이식하여 DWARF를 생성하는 프레임워크를 제시한다.

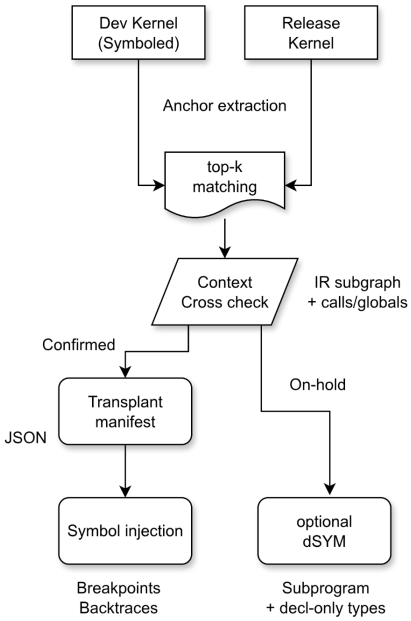
2. 분석

본 연구는 동일 빌드의 development 커널 바이너리를 기준으로 각 함수의 MRS(ARM system register read) 중심 패턴을 단서로 추출하고, release 커널의 무명 함수들과 동일 흐름을 보이는 후보를 대조해 역할 단위로 매핑을 진행하였다.

우선 각 development 함수에서 MRS 전후의 마스크·시프트·비교에 이르는 비트필드 해석, 전역 상태에 대한 플랫폼 특성 비트 갱신, 실행 환경 의존의 설정, 부트 데이터 질의, 초기화 직전의 리소스 준비, 그리고 함수 경계의 프롤로그·에필로그 및 간접 분기 보호(PAC/BTI) 를 증거 태그로 추출했다.

<표 1> release 빌드 함수 매핑 상태.

Subsystem	count
Trap dispatcher	17
VM memory ops	6
VM lifecycle & policy	5
Address space management	5
VCPU management	16
Guest state sync	8
Callbacks & event hooks	8
Low-level entry/return & init-support	5
State save/free	2
Total	72



(그림 1) 심볼 매핑 프레임워크.

이후 release 함수 후보에 대해 IR 수준의 국소 부분 그래프와 함수 호출자·참조자 맥락을 대조하여 확정 여부를 판단하였다.

그 결과는 <표 1>과 같이 트랩 디스패처, VM 메모리 연산, vCPU 관리, 게스트 상태 동기화 등 하위 역할별로 정리되며, 각 역할에 대해 release 빌드 환경을 기준으로 확정된 매핑을 분리해 제시한다.

3. 프레임워크

본 연구는 이름이 제거된 macOS release 커널 바이너리를 디버깅 가능하게 하기 위한 Dev to Release 심볼 이식 프레임워크를 제시한다.

본 프레임워크는 (그림 1)과 같이 동일 빌드의 development 및 release 커널 바이너리를 나란히 놓고, development 쪽에서 보이는 MRS 중심의 흐름과 그 주변의 비트필드 해석, 전역 특성 비트 갱신, 부트 데이터 질의, 그리고 프롤로그·에필로그 및 간접 분기 보호와 같은 행동 단서를 토큰화해 정규화한 앵커 서명을 만든 뒤, release의 무명 함수에도 같은 절차를 적용하여 서로 닮은 후보를 추려낸다.

이렇게 모인 후보 함수들에 대해 앵커가 나타나는 주변 기본 블록과 분기·메모리 접근 같은 IR 기반의 국소 구조, 그리고 문맥 정보로 교차 점검하여 신뢰할 수 있는 쌍을 확정한다.

확정된 결과는 빌드 UUID와 KASLR 정보를 함께 담은 JSON 형태의 이식 매니페스트로 정리되고, 이 매니페스트에서 자동으로 생성되는 심볼 주입 스

```
// Before: prototype & temps (auto-decomp)
_QWORD *__fastcall hv_update_controls(_QWORD *result) {
  __int64 v1;  // x8
  __int64 v2;  // x8
  v1 = result[1];
  v2 = result[1];
  *(_QWORD *)(result[1] + 16440LL) =
    *(_QWORD *)(result[1] + 16440LL) & 0xFFFFFFFF7CFFFFFFLL | 0x80300000LL;
  /* ... */
}

// After: prototype & types (recovered)
void __fastcall hv_update_controls(hv_vcpu_t *vcpu) {
  arm_guest_context_t *kif = vcpu->kif;
  arm_guest_context_t *v2  = vcpu->kif;
  kif->ro.controls.cptr_el2 =
    (kif->rw.controls.cptr_el2 & 0xFFFFFFFF7CFFFFFFULL) | 0x80300000ULL;
  /* ... */
}
```

(그림 2) 함수 매핑 예시.

크립트를 통해 release 커널에서 바로 브레이크포인트를 걸고 콜스택을 복원할 수 있다. 추가로 재배포나 재현성 요구가 있을 때만 외부 dSYM(DWARF)을 선택적으로 합성하도록 한다. 이때는 release 기준 주소의 DW_TAG_subprogram과 선언 수준 타입만 보수적으로 포함해 lldb/atos가 UUID로 자동 결합되도록 한다. 확정되지 못한 항목은 산출물에서 제외하고, 후속 자동화나 수동 검증을 위한 보류 큐에 남겨 점진적으로 승격시킨다. 매핑된 심볼에 대한 예시는 (그림 2)와 같다.

4. 결론

macOS release 커널은 심볼 제거와 최적화, 빌드 편차로 인해 대규모로 재현 가능한 디버깅이 어렵다. 우리는 macOS 15 HVF 경로를 중심으로 행동 단서를 묶어 Dev to Release 매핑을 정리했고, 이를 일반화한 심볼 이식 프레임워크를 제안했다. 이 접근은 오탐을 줄이면서 release 빌드 환경에서 즉시 브레이크포인트와 콜스택 복원을 가능하게 한다. 향후에는 macOS 리버스 엔지니어링 자동화 및 타입·앵커 단서에서 추출한 입력 스키마와 상태 머신을 통한 퍼징 하네스 자동 생성을 연구한다.

이 논문은 2025년도 정부(과학기술정보통신부)의 재원으로 한국정보기술연구원 차세대 보안리더 양성 프로그램(BoB)의 지원을 받아 수행된 연구임

참고문헌

[1] Chen, Weiteng, et al. "Syzgen: Automated generation of syscall specification of closed-source macos drivers." Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021.

[2] Tuby, Adam, and Adam Morrison. "Reverse Engineering the Apple M1 Conditional Branch Predictor for Out-of-Place Spectre Mistraining." arXiv preprint arXiv:2502.10719 (2025).