XPC·IOKit 기반 macOS 공격 표면 식별 자동화

이동하 전승호*

가천대학교 (학부생) *가천대학교

Automated macOS Attack Surface Identification via XPC and IOKit

DongHa Lee Seungho Jeon*

Gachon University Gachon University

(Undergraduate student)

요 약

macOS는 마이크로 아키텍처 기반의 복잡한 구조로 인해 공격 표면 식별이 어렵고, 취약점을 발견하더라도 재현 불가능한 사례가 많아 실제 위협이 되는 공격 경로를 식별할 필요성이 있다. 본 논문은 macOS 커널 확장 모듈 간의 상호 관계를 자동으로 분석하여 의미 있는 공격 벡터를 식별하는 방법론을 제안한다. Plist 기반 정적 의존성 추출을 통해 모듈 간 호출 흐름을 그래프로 모델링하고, 이를 기반으로 우선순위가 높은 공격 경로를 선제적으로 식별한다. 실험을 통해 총 853개의 KEXT에 대한 상호 관계를 성공적으로 맵핑하였으며, 실제 익스플로잇 체인이 적절히 식별됨을 확인하였다. 이를 통해 취약점 분석이 필요한 부분에 대한 우선순위를 효과적으로 식별할 수 있음을 입증하였다.

I. 서론

macOS는 마이크로 아키텍처 기반의 XNU 커널과 서비스 기반 유저랜드로 구성된다. 유저랜드의 기능들은 다수의 프레임워크와 XPC 서비스로 분리되어 실행되며 커널은 IOKit/IOUserClient를 통해 사용자 영역과 통신한다. 이러한 마이크로 아키텍처 설계는 각 서비스가 독립적으로 실행되어 하나의 서비스 실패가 전체 시스템에 영향을 주지 않도록 격리된 환경을 제공한다. 이 구조는 모듈화와 보안을 제공하지만, 공격 표면 식별을 어렵게 만든다. 기존에 연구된 취약점 분석 방법으로 실행 중 경유하는 일부 하위 모듈에서 보안 취약점을 발견할 수 있다. 그러나 상당수가 실제로 재현할 수 없어 악용 불가능한 케이스이다. 본 논문은 실제 위협이 되는 높은 의미 있는 공격 경로만을 자동으로 식별하는 방법을 제안한다. 핵심 아이디어는 Plist·Mach-O 기반 정적 의존성 추출을 통하여 모듈 간 호출 흐름을 그래프로 모델링하고 재현성·실효성을 기준으로 의미 있는 공격 경로를 식별하는 것이다. 이를 통해 취약점 분석이 필요한 부분에 대해 우선순위를 식별할 수 있다.

II. 배경 및 관련 연구

2.1 macOS 아키텍처

XNU 기반 커널은 마이크로커널 적 요소와 모놀리식 요소가 혼재하는 구조를 가진다. 유저랜드는 다수의 프레임워크와 독립 XPC 서비스로 기능이 분리되며, 커널-유저 인터페이스로 BSD 시스템콜·VFS와 함께 IOKit/IOUserClient가 병행 사용된다. 이 계층화는 권한 분리와 안정성을 제공하나 런타임의 메시지·포트 전달로 인해 상관관계 분석이 어려워진다.

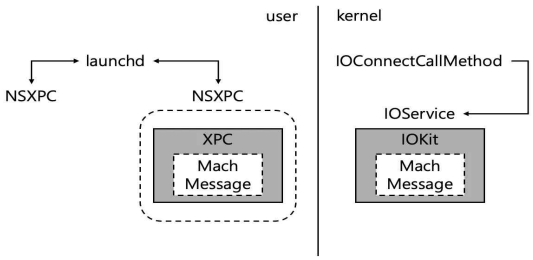

Figure 1. XPC/IOKit mechanism overview

2.2 KEXT와 DEXT 설계

전통적인 KEXT는 커널 공간에서 실행되는 확장 모듈로, 시스템의 핵심 기능을 확장하는 역할을 담당한다. KEXT는 커널 권한으로 실행되어 하드웨어 직접 접근과 시스템 리소스 관리가 가능하나, 커널 공간에서의 실행으로 인해 시스템 안정성의 위협과 권한 상승 공격 등의 잠재적 공격 벡터가 된다는 문제가 있다.

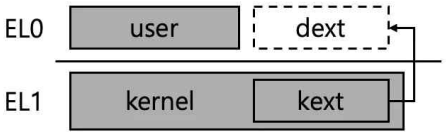

Figure 2. Concepts of DEXT and exception levels

최근에는 일부 드라이버 로직을 EL0의 Driver Extension(DEXT)로 이동하려는 전환이 진행 중이다. 이 변화는 사용자 공간에서의 실행으로 인해 공격 벡터를 줄이며 시스템 안정성을 향상한다.

2.3 XPC와 IOKit 통신 메커니즘

XPC는 프로세스 간 Mach 메시지 전달을 통해 서비스를 구성한다. IOKit은 Mach 포트와 IOConnectCallMethod를 통해 유저 측에 드라이버 기능을 노출한다. 두 메커니즘 모두 문서화가 제한적이고 런타임 전달 경로의 복잡성이 높아 서비스 간 상호 관계성 파악에 어려움이 발생한다.

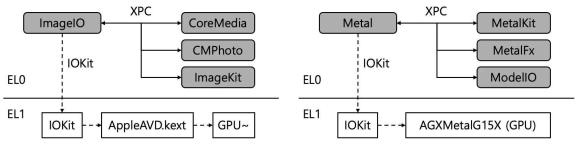

Figure 3. Example of using XPC/IOKit interface

애플 프레임워크 중 일부를 예시로 살펴보면 ImageIO·CoreMedia·ImageKit 등은 문서화된 고수준 API와 내부의 프라이빗 서비스(XPC 경로)를 통해 다층으로 호출되며, 하드웨어의 접근이 필요한 경우 IOKit을 통해 KEXT와 통신한다. 유사한 구조로 Metal 또한 고수준 API인 MetalKit·ModelIO으로부터 XPC를 통해 호출되고 IOKit을 통해 GPU 관련 KEXT가 호출된다.

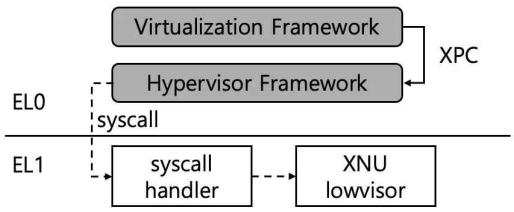

Figure 4. Example of using the system call interface

일부 프레임워크는 system call과 같은 IOKit 이외의 인터페이스를 사용하기도 하지만, 대부분의 KEXT는 IOKit을 통한 Mach 메시지를 기반으로 구현된다.

2.4 관련 연구

본 연구와 유사한 연구로 IMF가 있다. 이는 드라이버 상호작용 로그와 동적 트레이싱으로 상태를 추론해 상태 의존적 시퀀스 기반 퍼징을 수행함으로써 드라이버 취약점 탐지 효율을 개선하였다[1].

또한 Play Fuzzing Machine은 macOS/iOS 환경에서 IOUserClient·IOKit 진입점을 자동 식별하고 호출 시퀀스를 추출해 재현성 높은 퍼징 파이프라인을 제시하였다[2]. 본 연구는 두 선행 연구가 제공하는 동적 모델화·자동 호출 추출과 달리, 정적 관계 맵을 통해 우선순위가 높은 공격 경로를 선제적으로 식별하여 퍼징 대상과 검증 경로를 좁히는 점에서 차별화된다.

III. KEXT 상호 관계 분석 자동화

이 장에서는 macOS 커널의 유의미한 공격 벡터를 식별하기 위해 KEXT 간의 상호 관계

를 자동으로 분석하는 방법론을 제시한다.

1) Plist 구조 분석. Plist 구조 분석. MacBook Pro M3, macOS 26.0.1 환경에서 총 853개의 KEXT가 존재하며, 부팅 시 259개의 KEXT가 로드된다. KEXT는 사용자 요청을 처리하기 위해 서로 호출하며 상호작용을 한다. 이러한 상호작용 로직을 파악하기 위해 모든 KEXT를 리버싱하는 것은 상당한 시간이 소요된다. KEXT는 *.kext 형태의 디렉터리 내에 바이너리로 존재하며, Info.plist 파일이 함께 존재한다. Info.plist는 Property List(plist) 형식의 XML 파일로 KEXT 번들의 메타데이터를 포함한다. Info.plist 내부에는 번들의 식별자와 라이브러리 의존성, IOKit 드라이버 설정이 존재한다.

2) 상호 관계 맵핑. 앞서 파악된 plist 구조를 기반으로 모든 KEXT의 plist를 파싱하여 JSON 구조로 정리하였다. 이후 의존성 필드의 KEXT 식별자와 번들 식별자 필드의 KEXT 식별자를 매칭하여 상호 관계를 확보한다.

분석 대상 인터페이스를 선택한 후, 해당 경로를 거치는 KEXT들을 추출한다. 이를 통해 하위 노드에 존재하는 KEXT의 취약점을 트리거하기 위해 어떤 KEXT를 분석하고 공격해야 하는지 식별할 수 있다.

IV. 평가

이 장에서는 3장에서 제안한 KEXT 상호 관계 분석 자동화 방법론의 효과를 입증하기 위한 실험과 그 결과를 제시한다.

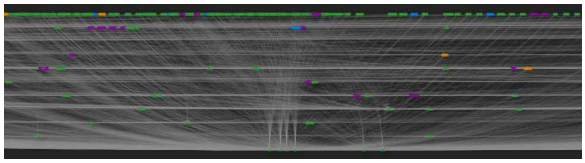

Figure 5. Visualization of mapping data

4.1 실험 환경

실험을 위해 상호 관계를 모두 맵핑한 데이터 파일을 GraphML 프레임워크를 통해 시각화하였다. 이후 2021년 Pwn2Own 대회에 사용된 익스플로잇 체인을 시각화한 그래프 내에서 확인하였다.

4.2 관계성 맵핑에 대한 유효성 평가

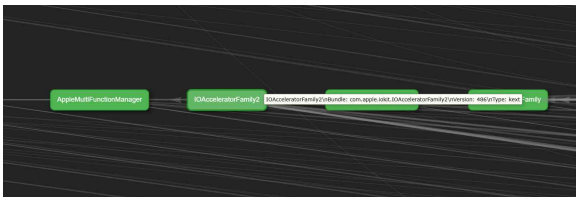

Figure 6. Exploit scenario in mapping data

CVE-2021-30734, CVE-2021-30735를 사용한 익스플로잇은 다음 경로를 거쳐서 실행된다.
Safari → GPU 프로세스 → IOAccelerator → 커널 내부의 VPHAL 파서 호출
Fig 1을 통해 해당 경로를 관계 맵핑 데이터에서 적절하게 확인할 수 있음을 증명하였다.

V. 결론

본 논문은 macOS의 복잡한 마이크로 아키텍처 환경에서 실제 위협이 되는 공격 경로를 자동으로 식별하는 방법론을 제안하였다. Plist와 Mach-O 기반 정적 의존성 추출을 통해 KEXT 간의 상호 관계를 그래프로 모델링하고, 재현성과 실효성을 기준으로 의미 있는 공격 벡터를 식별하는 접근법을 제시하였다. 실험을 통해 총 853개의 KEXT 중 부팅 시 로드되는 259개의 KEXT에 대한 상호 관계를 성공적으로 맵핑하였으며, 실제 익스플로잇 체인(CVE-2021-30734, CVE-2021-30735)이 제안한 방법론으로 적절히 식별됨을 확인하였다. 이를 통해 취약점 분석이 필요한 부분에 대한 우선순위를 효과적으로 식별할 수 있음을 입증하였다. 향후 더 다양한 공격 시나리오에 대한 검증과 런타임 동적 분석을 통한 정확도 향상을 통해 제안 기법의 실용성을 더 강화할 예정이다.

[참고문헌]

[1] HyungSeok Han and Sang Kil Cha, IMF: Inferred Model-based Fuzzer, ACM SIGSAC Conference on Computer and

Communications Security (CCS), 2017.

[2] Lilang Wu and Moony Li, Play Fuzzing Machine: Hunting iOS and macOS Kernel Vulnerabilities Automatically and Smartly, Virus Bulletin Conference (VB), 2019.