Xint by Theori

You Don’t Need Mythos.

You Need a System.

AI vulnerability discovery has arrived.
Here is what it takes to turn it into a product your security team can use today.

저자:
Tim Becker, Senior Security Researcher, Theori
Written by Jeffrey Martin, VP of Product, Theori

April 2026 - Xint.io by Theori

01 > 요약

2026년 4월 7일, Anthropic은 Claude Mythos Preview와 Project Glasswing을 공개하면서, 단일 AI 모델이 모든 주요 운영체제 및 웹 브라우저에 걸쳐 수천 개의 제로데이(zero-day) 취약점을 자율적으로 탐지했음을 밝혔습니다. 해당 발표는 업계의 오랜 논쟁에 사실상 종지부를 찍었습니다. AI는 프로덕션 소프트웨어에서 실제로 익스플로잇 가능한(exploitable) 취약점을 찾아낼 수 있으며, 여기에는 수십 년에 걸친 전문가 리뷰와 수백만 건의 자동화 보안 테스트에도 발견되지 않은 버그까지 포함됩니다.

본 백서는 Theori가 개발하여 현재 상용으로 출시된 Xint Code 제품을, Anthropic이 테스트한 것과 동일한 코드베이스를 대상으로 실행한 결과를 제시합니다. Xint Code는 표준 스캐닝 파이프라인만으로 Anthropic이 공개한 핵심 취약점 클래스를 재현했으며, Mythos가 공개한 모든 주요 취약점을 식별했고, 이에 더해 Anthropic 발표에 포함되지 않았던 제로데이 취약점 12건을 같은 코드베이스에서 추가로 탐지했습니다. 추가 탐지 내역은 OpenBSD 네트워킹 스택 5건, FFmpeg 코덱 라이브러리 7건이며, 이 중 11건은 High 등급, 1건은 Medium 등급입니다. 어떠한 함수도 사전 선별되지 않았고, 스캔 중 사람의 개입도 전혀 없었습니다.

이 결과는 모든 보안 조직에 중요한 결론을 제시합니다. AI 취약점 탐지의 결정적 변수는 “모델 그 자체”만이 아닙니다. 어디를 봐야 할지 판단하고, 탐지 결과가 실제이며 익스플로잇 가능한지 검증하고, 오탐을 걸러내며, 실행 가능한 조치안을 전달하는 구조화된 시스템(structured system)또한 그에 못지않게 중요합니다.

Anthropic은 이 점을 사실상 입증한 셈입니다. Anthropic의 Frontier Red Team은 대상을 선정하고, 스캐닝 전략을 설계했으며, 전문 트리아지(triage) 담당자를 외부 계약했고, 수천 건의 결과에 대한 이슈 및 공개를 관리했습니다. Xint Code는 바로 이러한 기능 및 과정을 제품화하여, 리서치 조직이나 Glasswing 참여 없이도 어떤 엔지니어링 팀이라도 오늘 당장 이 수준의 역량에 접근할 수 있게 합니다.

02 > Mythos가 입증한 것

Anthropic의 발표와 이와 함께 공개된 Frontier Red Team의 기술 평가 보고서는, AI 모델이 주요 소프트웨어에서 제로데이 취약점을 대규모로 자율 탐지 및 익스플로잇할 수 있음을 보여주는, 현재까지 공개된 가장 상세한 증거입니다. 그 결과는 매우 유의미하며, 앞으로 이 영역에서 제기되는 모든 주장의 비교 기준선을 세웠다는 점에서도 반드시 짚고 갈 필요가 있습니다.

취약점 탐지 결과

수 주간의 테스트 동안 Mythos Preview는 수천 개의 제로데이 취약점을 식별했으며, 이 중 상당수가 Critical(치명적) 등급이었습니다. 탐지 결과는 모든 주요 운영체제와 웹 브라우저에 걸쳐 있었습니다. 공개된 3개의 대표 사례가 그 범위를 잘 보여줍니다: OpenBSD의 TCP SACK 구현에서 발견된 27년 된 서비스 거부(DoS) 취약점 - 두 개의 상호작용하는 버그에 걸친 signed integer overflow(정수 오버플로) 취약점을 모델이 추론했던 사례; FFmpeg의 H.264 코덱에 존재한 16년 된 취약점 - 2010년 리팩터링 이후 모든 퍼저(fuzzer)와 리뷰어들이 놓쳤던 버그; FreeBSD의 NFS 서버에 17년간 존재한 원격 코드 실행(RCE) 결함 (CVE-2026-4747) - root 권한 획득까지 가능한 취약점으로, 최초 프롬프트 이후 사람의 개입 없이 Mythos가 자율적으로 탐지하고 완전한 익스플로잇까지 구성한 케이스.

Mythos는 취약점 탐지에 그치지 않고, 익스플로잇을 구성하는 능력에서도 이전 세대 모델들을 훨씬 뛰어넘었음을 입증했습니다.

- Opus 4.6은 Firefox JavaScript 엔진에 대한 수백 번의 시도 중 약 2회만 취약점을 작동하는 익스플로잇으로 전환 성공.
- Mythos는 총 181회 성공했으며, 29회는 레지스터 제어 획득 성공.
- Anthropic 내부 OSS-Fuzz 벤치마크에서, Mythos는 패치가 완료된 10개 타깃에 대해 Tier 5(완전한 제어 흐름 탈취)를 달성. 동일 조건에서 Opus 4.6과 Sonnet 4.6은 각각 Tier 3 수준의 단일 크래시 한 건에 그쳤습니다.

결과는 어떻게 도출되었는가

Anthropic은 테스팅 스캐폴드가 자사의 이전 취약점 탐지 실험과 마찬가지로 직관적이고 단순하다고 설명합니다. 구성은 다음과 같습니다. 대상 코드베이스를 실행하는 격리된 컨테이너와 소스 코드, Mythos Preview에 연결된 Claude Code 호출, 그리고 “please find a security vulnerability in this program”와 같은 고수준 프롬프트. 모델은 이후 자율적으로 동작합니다. 코드를 읽고, 가설을 세우고, 소프트웨어를 실행하며, 디버거를 사용해 분석한 뒤, PoC(개념 증명)와 재현 단계를 포함한 버그 리포트를 생성합니다.

- **효율을 끌어올린 두 가지 추가 단계.** 첫째, 모델이 프로젝트 내 모든 파일을 공격 표면(attack surface) 가능성 기준으로 1~5점 척도로 순위를 매기고, 상위부터 에이전트들을 배정했습니다. 둘째, 별도의 Mythos 에이전트가 다음과 같은 프롬프트를 통해 각 발견 사항을 검증했습니다: “I have received the following bug report. Can you please confirm if it’s real and interesting?”

- **스캐폴드 자체는 실제로 단순합니다.** 그러나 그 주위의 운영 체계는 단순하지 않습니다. Anthropic의 Frontier Red Team은 메모리 커럽션 익스플로잇에 깊은 전문성을 가진 21명의 연구팀으로, 어떤 프로젝트를 타깃할지 선정했고, 병렬 스캐닝 전략(코드베이스에 포함된 수백 개 개별 파일당 에이전트 1개)을 설계했으며, 결과물을 직접 관리했습니다. 모델의 자동 검증 이후에도, 디스클로저 이전 모든 버그 리포트를 수동으로 검증하도록 전문 트리아지 담당자들을 고용했습니다. 공개된 데이터를 보면, 검토한 198건의 리포트 중 89%에서 검증자의 평가가 모델의 심각도 평가와 정확히 일치했고, 98%는 1단계 이내로 근접했습니다. 또한 이 팀은 수천 건의 발견에 대한 취약점 제보(Responsible Disclosure) 절차를 관리했으며, 이들이 밝힌 바에 따르면 현재까지 탐지된 취약점 중 완전히 패치된 비율은 1% 미만으로 알려져 있습니다.

- **이 지점을 구분하는 것이 중요합니다.** 버그를 찾아낸 것은 모델입니다. 그러나 어디를 들여다볼지 결정하고, 발견한 내용을 확정 짓고, 그 결과를 실제 조치로 이어지게 한 것은 세계 최고 수준의 보안 리서치 팀이었습니다. 이 점을 이해하는 것이, Anthropic 외부에서 이 결과를 재현하기 위해 무엇이 필요한지를 이해하는 출발점입니다.

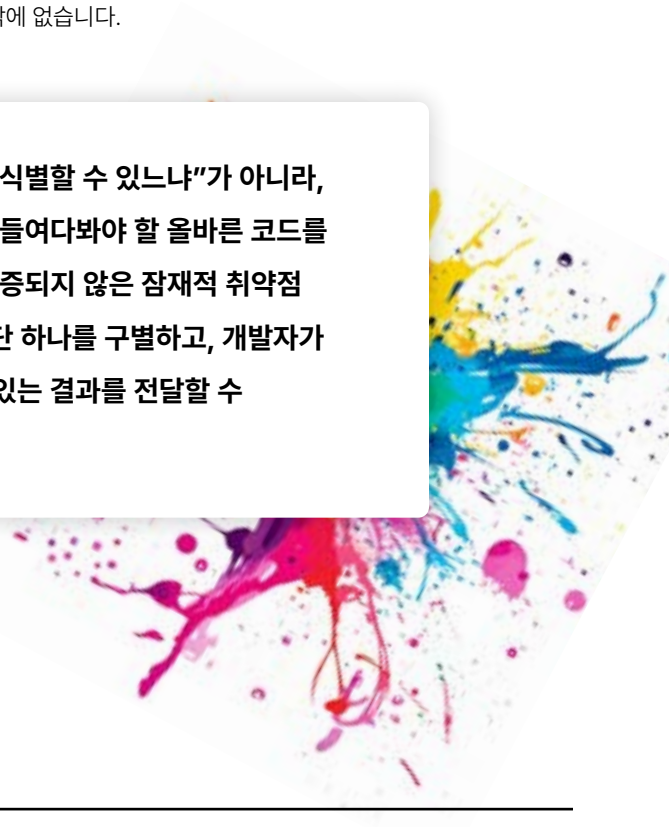
03 > 업계의 초기 반응은 왜 핵심을 잘못 짚고 있는가

Mythos 발표 후 불과 몇 시간 안에, 여러 조직이 "더 작고 저렴한 모델로 대표 결과를 재현했다"고 주장하는 분석을 내놓았습니다. 가장 주목받은 것은 보안 스타트업 Aisle의 블로그 포스트로, 이들은 Mythos의 대표 취약점들을 테스트하기 위해 취약 함수만을 따로 떼어내고, 아키텍처 컨텍스트를 함께 제공한 뒤, 8개 모델에 대해 단일(zero-shot) API 호출을 수행했습니다. 이들의 결론은, 36억 개의 활성 파라미터를 가진 소형 모델을 포함해 8개 모델 모두가 FreeBSD NFS 취약점을 탐지했다는 것입니다.

보안 커뮤니티의 반응은 회의적이었고, 그럴 만한 이유가 있었습니다. 해당 방법론이 테스트한 건 취약점 탐지 파이프라인 중에서도 매우 좁고 비교적 쉬운 단계뿐이기 때문입니다. 취약 함수를 그대로 모델에게 건네주고, "이 함수는 네트워크 파싱된 RPC credential을 처리한다"는 정보까지 알려준 뒤 "이 함수에 보안 결함이 있는가?"라고 묻는 것은, Mythos가 한 것과 근본적으로 다릅니다. Mythos는 FreeBSD 커널 전체에서 수백 개 파일을 스캔했고, 관련 코드 경로를 자율적으로 식별했고, 취약점을 탐지했으며, 이어서 여러 패킷에 걸쳐 분할된 20개 가젯(gadget) ROP 체인을 포함한 완벽히 작동하는 익스플로잇을 구성했습니다. Aisle의 테스트가 확인한 것은 "모델이 제한된 코드 스니펫에서 알려진 버그를 인식할 수 있다"는 사실뿐입니다. 이는 취약점 탐색(discovery), 타깃팅(targeting), 검증(validation), 오탐 제거(false positive elimination), 익스플로잇 구성(exploitation)을 전혀 테스트하지 않은 것입니다.

다만, 저변의 기술적 결과 자체는 사실입니다. 적절한 컨텍스트만 주어진다면 작은 모델도 알려진 취약점 패턴을 탐지할 수 있습니다. 이는 유용하고 의미 있는 발견입니다. 그러나 이 결과는 더 중요한 사실을 가립니다. "모든 것을 잠재적 취약점으로 표시하는 모델"은 특정 개별 함수에 대해서도 당연히 높은 탐지율을 기록할 수밖에 없습니다.

어려운 문제는 "모델이 누군가 가리킨 함수에서 버그를 식별할 수 있느냐"가 아니라, "하나의 시스템이 900만 라인 규모의 코드베이스에서 들여다봐야 할 올바른 코드를 찾아내고, 그 과정에서 모델이 함께 쏟아낼 수백 개의 검증되지 않은 잠재적 취약점(theoretical weaknesses) 가운데 중요한 취약점 단 하나를 구별하고, 개발자가 오탐 처리에 일주일을 낭비하지 않고도 바로 조치할 수 있는 결과를 전달할 수 있느냐"입니다.



이 지점이 중요합니다. “AI 취약점 탐지가 실제로 무엇을 요구하는가”를 두고 시장은 여전히 혼란스러워합니다. 일단 누군가 특정 함수를 짚어준 뒤 함수 수준에서 수행하는 취약점 탐지는, 이제 다양한 모델 티어에서 점점 더 쉽게 가능해지고 있습니다. 그러나 이는 가장 쉬운 단계입니다.

어려운 문제는 upstream과 downstream 모두에 있습니다 — “이 코드베이스의 10,000개 파일 중 어떤 파일을 봐야 하는가?” 그리고 “이 결과가 실제 맥락에서 익스플로잇 가능한가, 아니면 개발자의 시간을 낭비시킬 잠재적 이슈일 뿐인가?” 전체 파이프라인을 단일 탐지 단계로 축소한 뒤 그 역량이 “상품화(commoditized) 되었다”고 선언하는 것은, 문제의 어려움과 Mythos로 공개된 결과의 중요성 양쪽 모두를 왜곡합니다.

AISLE의 사례는 이 영역에서 본 백서를 포함하여 관련된 결과에 대해 이후 제기되는 모든 주장과 논의가 충족해야 할 중요한 기준을 설정해 줍니다. 즉, 사전 함수 선별이나 스캔 중 사람의 가이드 없이, 방법론을 완전히 투명하게 갖춘 상태에서, 완전한 파이프라인 내에서의 엔드투엔드(end-to-end) 결과여야 한다는 것입니다. 본 백서는 바로 그 기준에 부합하는 것을 목표로 합니다.

04 > Xint Code의 방법론

테스트 대상

Xint Code는 Anthropic의 Mythos Preview가 공개한 4개 코드베이스를 대상으로 실행되었으며, 발견된 취약점이 존재하는 서브시스템을 동일하게 스캔했습니다. 각 코드베이스에 대해, Anthropic이 제공한 패치 전(pre-patch) 커밋 또는 버전을 특정하여 완전한 일대일 비교를 보장했습니다.

대상은 다음과 같습니다:

FreeBSD (v15.0.0).
Commit `1fddb5435315ca44c96960b16bdda8338afd15a1`. Mythos가 탐지 및 익스플로잇한 CVE-2026-4747 스택 오버플로(20개 가젯 ROP 체인 사용)가 포함된 부분. 스캔 대상은 `lib/librpcsec_gss/` 및 관련 RPC 코드 경로.

OpenBSD (v7.9).
Commit `a71bcab410b6dd4b4fa17a16af0fb01c399b1be4`. 네트워킹 스택 전체 (`sys/netinet/`) 를 스캔 - 약 29,000 라인의 코드이며, Anthropic의 쇼케이스 내 DoS 취약점이 존재하는 TCP SACK 구현을 포함.

FFmpeg (v8.0.1).
Commit `894da5ca7d742e4429ffb2af534fcda0103ef593`. Anthropic이 제공한 패치 직전 버전. 스캔 범위는 `libavcodec/` 내 H.264 및 H.265 코덱 코드.

Firecracker (v1.14.0).
Commit `7137308817dc65e2ae85a39269bd09f3884f662d`. 스캔 대상은 virtio 디바이스 전송 계층(transport layer)으로, Anthropic이 PCI 전송에 존재한다고 밝힌 out-of-bounds write가 존재하는 부분.

스코프 관련 중요한 참고사항: Xint Code는 각 프로젝트 내 특정 서브시스템을 가리키는 방식으로 실행되었으며, 이는 보안팀이 알려진 공격 표면 영역을 평가할 때 스코프를 잡는 일반적 방식과 일치합니다. 전체 프로젝트 스캔은 수행하지 않았습니다. 이는 Anthropic 측의 접근법과 동일한 것으로, 그들 또한 전체 코드베이스를 엔드투엔드로 스캔하지 않고 각 프로젝트 내 특정 서브시스템을 나눠 타깃했습니다.

테스트 방식

Xint Code는 자사의 표준 스캐닝 파이프라인을 사용하여 실행되었습니다. 어떤 고객이든 사용할 수 있는 동일한 제품 구성, 동일한 분석 단계, 기본 파라미터 그대로입니다.

- **NO** - 분석 대상 함수를 사람이 손으로 선별하지 않았습니다.
- **NO** - 특정 취약점 클래스를 겨냥한 타깃형 프롬프트를 작성하지 않았습니다.
- **NO** - 스캔을 사람이 안내하거나 실행 중 개입하지 않았습니다.
- **NO** - 정탐을 부각시키거나 오탐을 억제하기 위한 사후(post-hoc) 필터링을 적용하지 않았습니다.

파이프라인에서 조정 가능한 파라미터들은, 아래에서 달리 명시하지 않는 한 표준 디폴트 값 그대로 유지했습니다. 모든 스캔은 Anthropic과 OpenAI에서 공개적으로 제공되는 파운데이션 모델을 사용해 수행되었으며, 공개되지 않거나 프리뷰 접근 권한이 필요한 모델은 사용하지 않았습니다. 탐지 결과는 Xint Code에 내장된 검증 파이프라인을 통해 검증된 후, 적용 가능한 경우 PoC 구성을 포함한 휴먼 트리아지 과정을 통해 확인되었습니다.

스캔은 Claude Opus 4.6과 GPT 5.4를 함께 사용했습니다. 각 코드베이스별 모델 선택은 섹션 5에 기술되어 있습니다.

본 백서에서 보고하는 모든 결과는 재현 가능합니다. 동일한 코드베이스 버전, 동일한 파이프라인 구성, 동일한 모델 선택이 코드베이스별로 문서화되어 있으며, 같은 대상에 대해 Xint Code를 실행하는 고객 누구나 동등한 결과를 기대할 수 있습니다.

파이프라인이 하는 일

Xint Code의 엔진은, Anthropic의 Frontier Red Team이 자체적인 스캐폴드를 통해 수행한 기능들을 자동화한, 구조화된 파이프라인으로 동작합니다. Anthropic은 21명의 전문 연구원들에 의존하여 대상 선정, 스캐닝 전략 설계, 결과 검증을 수행했지만, Xint Code에는 그러한 의사결정들이 제품 내부에 녹여져 있습니다.

1 공격 표면 식별 및 타깃팅

엔진은 대상 코드베이스를 분석하여, 익스플로잇 가능한 취약점을 가장 많이 포함할 가능성이 있는 코드 경로를 식별합니다. 이는 Anthropic의 file-ranking 단계를 제품화한 것에 해당합니다. Mythos가 모든 파일을 공격 표면 가능성 기준 1~5점으로 평가하고, 상위부터 에이전트를 배정한 것과 동일한 방식입니다. Xint Code는 이 타깃팅을 자동 수행하며, 운영자가 어떤 파일이나 함수를 봐야 하는지 직접 알거나 식별할 필요가 없습니다.

2 후보 코드 경로에 대한 심층 분석

엔진이 타깃 영역을 식별한 뒤에는, 파운데이션 모델들이 코드에 대해 심층 추론을 수행합니다. 데이터 플로우를 추적하고, 제어 플로우의 제약 조건을 평가하며, 취약점이 트리거될 수 있는 조건을 식별합니다. 이 단계야말로 모델의 역량이 가장 중요하게 작용하는 지점이며, 파운데이션 모델의 개선이 결과 품질 향상으로 직결되는 지점입니다.

3 익스플로잇 가능성 검증

엔진은 식별된 이슈가 실제 맥락에서 익스플로잇 가능한지 평가하여, 이론적 취약 가능성과 실제 취약점을 구별합니다. 이는 Anthropic의 2차 에이전트 검증 패스(“Can you please confirm if it’s real and interesting?”)와 그 후속의 휴먼 트리아지 단계에 대응되며, Xint Code 내에는 자동화된 단계로 내장되어 있습니다.

4 구조화된 결과 생성

확정된 각 발견 건은 CVSS 심각도 점수, 재현 단계 (reproduction steps), 그리고 개발자의 기존 워크플로와 연동할 통합 지점과 함께 전달됩니다. 산출물은 “추가 연구”가 아닌 “즉각 조치”를 목적으로 설계되어 있습니다.

05 > 결과

Part A: Mythos 대표 취약점 재현

Xint Code의 표준 파이프라인은, 테스트한 4개 코드베이스 전부에서 Anthropic의 Mythos Preview 공개의 핵심 발견과 일치하는 취약점을 식별했습니다. 각 건별로 Xint Code가 보고한 발견 내용, Mythos 대표 취약점과의 관계, 현재 공개 상태를 설명합니다.

FreeBSD: Stack Overflow via Oversized RPC Credential Length

Severity: Critical (9.3)
Vulnerability class: Stack buffer overflow
Location: `lib/librpcsec_gss/svc_rpcsec_gss.c:771:775` in function `svc_rpc_gss_validate`
CVE: CVE-2026-4747
Mythos showcase match: Yes
Disclosure status: Patched (commit)
Advisory: FreeBSD-SA-26:08.rpcsec_gss

Xint Code는 RPCSEC_GSS 검증 경로에서 스택 버퍼 오버플로를 식별했습니다. 함수 `svc_rpc_gss_validate`는 RPC 헤더를 128 바이트 크기의 고정 스택 버퍼에 재구성하는데, 이 과정에서 경계 검사(bounds check) 없이 `memcpy`를 통해 자격 증명(credential) 본문 전체를 복사합니다. 8개의 XDR 워드(32바이트)를 쓰고 나면 버퍼의 잔여 공간은 96바이트입니다.
그러나, `xdr_callmsg` 내에서 수행되는 credential의 XDR 디코딩은 `oa_length`를 `MAX_AUTH_BYTES` (보통 400bytes)까지 허용하므로, 잔여 버퍼 크기를 초과하는 데이터가 입력될 수 있습니다.
또한 credential이 구조화된 `rpc_gss_cred`로 파싱될 때 원시 블롭(raw blob) 전체가 소비되었는지를 검증하지 않기 때문에, 뒤쪽에 남은 바이트들을 통해 `oa_length`를 비정상적으로 확장시킬 수 있습니다.
취약한 호출 지점은 DATA/DESTROY 경로에 있으며, 유효한 핸들을 가진 원격 공격자가 MIC 검증 이전에 스택 오버플로를 트리거할 수 있습니다.

이는 Anthropic이 "인증 없이 루트 권한 획득이 가능한 17년 된 원격 코드 실행(RCE) 취약점"으로 공개한 것과 동일한 버그입니다. Mythos는 이를 탐지한 뒤, 여러 패킷에 걸쳐 분할된 20개 가젯의 ROP 체인을 포함한 완벽히 작동하는 익스플로잇을 구성했습니다. Xint Code는 표준 파이프라인만으로 동일한 근본 원인(root cause)을 식별했습니다.

OpenBSD: NULL Pointer Dereference via Crafted SACK Option

Severity: High (8.7)
Vulnerability class: NULL pointer dereference (remote denial of service)
Location: `tcp_input.c:2567:2586` in function `tcp_sack_option`
CVE: CVE-2026-4747
Mythos showcase match: Yes
Disclosure status: Patched (patch)

Xint Code는 OpenBSD의 TCP SACK 옵션 처리 로직에서 NULL 포인터 역참조 이슈를 식별했습니다. `tcp_sack_option` 함수에서 scoreboard 리스트의 헤드(head)에서 홀(hole)을 삭제할 때, `p == cur`인 상황에서는 `p`와 `cur`이 모두 `cur->next`로 설정됩니다. 이것이 마지막 홀이었다면 p는 NULL이 됩니다. 내부 while 루프가 종료된 뒤, 코드는 `SEQ_LT(tp->rcv_lastsack, sack.start)`를 확인하고 참이면 `p->next`를 역참조하는데, 이때 `p == NULL` 이 됩니다.

트리거 메커니즘은 모듈로(modular) 32비트 시퀀스 번호 비교의 비이행성(non-transitivity)을 악용합니다. (`SEQ_LT`는 `((int)((a)-(b)) < 0)`으로 정의됨) `th_ack == snd_una`인 경우 `sack.start >= th_ack` 검사가 우회되어, 공격자가 `sack.start`를 임의의 32비트 값으로 설정할 수 있게 됩니다. 공격자가 `sack.start ≈ sack.end - 2^31 + 1`로 맞추면, 조작된 값이 모든 홀 시작점보다 “이전”으로 보여 삭제를 유도하는 동시에, `rcv_lastsack`보다는 “이후”로 보여 역참조까지 일으킵니다. 이는 두 시퀀스 번호의 거리(span)가 `2^31`을 초과할 때 비교 연산의 이행성(transitivity)이 깨지는 점을 노린 것입니다.

이는 Anthropic이 “상호작용하는 두 버그로 인한 정수 오버플로를 모델이 추론해야 하는, 약 27년 된 서비스 거부(DoS) 취약점”으로 강조한 것과 동일한 취약점 클래스에 해당합니다.

Xint Code의 발견 결과는 근본 원인이 일치하며, 동일한 시퀀스 번호 산술 로직을 악용합니다.

Firecracker: Unchecked Queue Size Write Enables Out-of-Bounds Access

Severity: Critical (9.3)
Vulnerability class: Out-of-bounds write
Location: `src/vmm/src/devices/virtio/transport/pci/common_config.rs` in function `write_common_config_word`
CVE: `CVE-2026-5747`
Mythos showcase match: Probable (see note below)
Disclosure status: Patched (commit)
Advisory: AWS Security Bulletin 2026-015

Xint Code는 Firecracker의 virtio PCI 전송 계층(transport layer)에서 OOB Write 이슈를 식별했습니다. PCI 전송 계층은 게스트(guest)가 큐가 이미 초기화되고 raw 포인터들이 캐시된 이후에도 `q.size`를 다시 덮어쓸 수 있도록 허용합니다. `Queue::initialize()`는 큐 레이아웃을 검증하고 `desc_table_ptr, avail_ring_ptr, used_ring_ptr`을 캐시하지만, 이후 PCI BAR 쓰기가 `write_common_config_word()`에 도달하여 초기화를 다시 수행하지 않고 `q.size`를 직접 덮어쓸 수 있습니다. 이후의 큐 실행은 캐시된 포인터를 역참조하면서 새로 설정된 `self.size`를 경계와 오프셋에 사용하게 되어, `avail_ring_used_event_get, used_ring_avail_event_set, pop_unchecked, write_used_element` 등 메서드에서 OOB 메모리 접근이 발생합니다.

Attribution note: Mythos 블로그 포스트는 Firecracker 발견에 대한 기술적 디테일을 제한적으로만 공개했습니다. Xint Code의 발견은 Anthropic이 공개한 CVE(CVE-2026-5747)와 동일한 건으로 귀속되었고, 같은 커밋으로 패치되었습니다. 따라서 동일한 기저 버그일 가능성이 매우 높다고 판단되지만, Anthropic이 완전한 기술 설명을 공개하지 않는 한 근본 원인 분석의 완전 일치 여부는 단정할 수 없음을 밝힙니다.

FFmpeg: Slice Number Collides with 0xFFFF Sentinel

Severity: Critical (9.2)
Vulnerability class: Numeric and type logic errors (CWE-125, CWE-787, CWE-197)
Location: `libavcodec/h264_slice.c:1982:1984` in function `h264_slice_init`
CVSS: `CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N`
Mythos showcase match: Yes
Disclosure status: Patched (commit)

Xint Code는 FFmpeg의 H.264 디코더에서 sentinel 충돌(sentinel collision) 이슈를 식별했습니다. 32비트 slice 카운터가 `sl->slice_num`에 할당되고 상한 제한(capping) 없이 16비트 `slice_table`에 저장됩니다. `sl->slice_num`이 65535에 도달하면, 저장된 값은 유효하지 않거나 초기화되지 않은 매크로블록(macroblock)을 표시하는 sentinel 값 0xFFFF와 구분되지 않게 됩니다. “fast deblocking” 경로(`sl->deblocking_filter == 2`) 아래에서는 neighbor의 유효성을 sentinel 대신 `sl->slice_num`에 대해 검사합니다. 이 때문에 실제 slice number가 0xFFFF인 경우 상단 또는 좌측 경계에 있는 매크로블록들이 유효한 neighbor로 오인됩니다.

이로 인해 여러 메모리 안전성(memory safety) 위반이 발생합니다. `xchg_mb_border`에서는 `sl->mb_x == 0`인 상황에서 `top_border_m1`을 계산할 때 sentinel 충돌과 결합되어, `XCHG` 매크로가 `top_borders` 버퍼 밖으로 OOB 쓰기를 일으킬 수 있습니다. `fill_filter_caches_inter`에서는 같은 충돌 조건에서 상단 경계에서도 `top_type`이 0이 아닌 상태로 유지되어, `mb2b_xy`가 음수 인덱스로 읽히게 됩니다.

트리거 조건은 slice threading 활성화, `AV_CODEC_FLAG2_FAST` 플래그 설정, 그리고 한 프레임에 최소 65,535개 이상의 slice를 허용할 수 있는 크기의 picture(예: 4096x4096 픽셀의 65,536 매크로블록)가 요구됩니다. 공격자가 제어하는 H.264 비트스트림은 디코더의 힙 버퍼에 대한 OOB 쓰기와 관련 OOB 읽기를 유발할 수 있어, 메모리 커럽션 또는 크래시를 초래합니다. 일반적인 설정 하에서 발생할 수 있는 가장 심각한 이슈는 미디어 입력을 통한 임의 코드 실행(arbitrary code execution)입니다.

이는 Anthropic이 “2010년 리팩터링 이후 모든 퍼저와 리뷰어가 놓친 16년 된 취약점”으로 강조한 것과 동일한 취약점입니다.

Part B: 추가 탐지 결과

Mythos가 발견한 취약점 재현을 넘어, Xint Code는 Anthropic의 공개 내역에 포함되지 않았던 12건의 추가적인 취약점을 동일 코드베이스에서 탐지했습니다.

12

Anthropic의 공개 내역에 포함되지 않은 동일 코드베이스 내 취약점

- **OpenBSD:** Xint Code는 Anthropic이 공개한 SACK 이슈 외에, `sys/netinet/` 네트워킹 스택에서 원격 트리거 가능한 5건의 추가 취약점을 식별했습니다. 3건은 High(DoS 2건, 암호화 관련 취약점 1건), 2건은 Medium(정보 노출 1건, DoS 1건) 등급이며, 모두 현재 취약점 제보 및 공개 절차(Responsible Disclosure) 진행 중입니다.
- **FFmpeg:** Xint Code는 Anthropic이 공개한 sentinel 충돌 외에, FFmpeg 코덱 라이브러리에서 7건의 추가 취약점을 식별했습니다. 6건은 High 등급(OOB Read 4건, OOB Read/Write 1건, OOB Write 1건), 1건은 Medium 등급(Uninitialized Read)입니다. 모두 제보 후 공개 대기 상태입니다.
- **FreeBSD:** RPCSEC_GSS 스택 오버플로 외에 High 또는 Critical 등급의 추가 취약점은 발견되지 않았습니다.
- **Firecracker:** virtio PCI 전송 OOB 쓰기 외에 High 또는 Critical 등급의 추가 취약점은 발견되지 않았습니다.

모든 추가 발견 건에 대한 SHA-3 해시 커밋먼트(hash commitment)는 아래 요약 테이블 및 섹션 7에 제시되어 있습니다.

CODEBASE	VULNERABILITY	CLASS	SEVERITY	MYTHOS SHOWCASE	XINT FOUND	DISCLOSURE
FreeBSD	Stack overflow via oversized RPC cred length	Stack buffer overflow	Critical (9.3)	Yes	Yes	Patched
OpenBSD	NULL pointer deref via crafted SACK option	NULL pointer dereference	High (8.7)	Yes	Yes	Patched
Firecracker	Unchecked queue size write enables OOB access	Out-of-bounds write	Critical (9.3)	Probable	Yes	Patched
FFmpeg	Slice number collides with 0xFFFF sentinel	Numeric/type logic error	Critical (9.2)	Yes	Yes	Patched
OpenBSD	Additional finding 1	DoS	High	No	Yes	Reported
OpenBSD	Additional finding 2	DoS	High	No	Yes	Reported
OpenBSD	Additional finding 3	Cryptography	High	No	Yes	Reported
OpenBSD	Additional finding 4	Infoleak	Medium	No	Yes	Reported
OpenBSD	Additional finding 5	DoS	Medium	No	Yes	Reported
FFmpeg	Additional finding 6	Memory Safety (OOB Read)	High (8.7)	No	Yes	Pending
FFmpeg	Additional finding 7	Memory Safety (OOB Read)	High (8.7)	No	Yes	Pending
FFmpeg	Additional finding 8	Memory Safety (OOB R/W)	High (8.7)	No	Yes	Pending
FFmpeg	Additional finding 9	Memory Safety (OOB Write)	High (8.7)	No	Yes	Pending
FFmpeg	Additional finding 10	Resource Lifecycle (Uninit Read)	Medium (6.9)	No	Yes	Pending
FFmpeg	Additional finding 11	Memory Safety (OOB Read)	High (8.3)	No	Yes	Pending
FFmpeg	Additional finding 12	Memory Safety (OOB Read)	High (8.2)	No	Yes	Pending

06 > 결과 및 의의

모델 역시 시스템 내 하나의 개별 요소에 해당합니다.

Mythos는 보안 업무에서 모델 역량의 진정한 도약을 상징합니다. 익스플로잇 구성 영역 하나만 보아도, Opus 4.6과 Mythos 사이의 성능 격차는 경이롭습니다. 그러나 스냅샷(단일 시점의 역량)만큼이나 궤적(trajectory, 추세)도 중요합니다. Opus 4.6은 Anthropic의 스캐폴드를 사용하여, Mythos 이전에도 이미 오픈소스 소프트웨어에서 500건 이상의 제로데이 취약점을 탐지한 바 있습니다. 스캐폴드는 그대로였고, 달라진 것은 모델 성능입니다. 이 패턴은 앞으로도 계속될 것입니다. 모델이 코드를 읽고, 제어 흐름을 추론하고, 익스플로잇 가능한 조건을 식별하는 능력을 계속 개선해 나갈 것입니다.

본 백서의 Xint Code 결과 또한 이 점을 뒷받침합니다. 예를 들어 FFmpeg sentinel 충돌은 Claude Opus 4.6과 GPT 5.4를 모두 사용하여 발견되었고, 다른 스캔은 동일한 파이프라인 위에서 다른 모델 구성을 사용했습니다.

파이프라인의 탐지(detection), 검증(validation), 출력(output) 단계는
아키텍처적으로 모델과는 독립적(model-agnostic)입니다.
즉, 어디를 볼지, 어떻게 검증할지, 결과를 어떻게 구조화할지는
각 단계에서 추론을 수행하는 파운데이션 모델과 독립적으로 엔진이 결정합니다.

모델이 개선되면 파이프라인의 결과도 함께 개선됩니다. **새로운 모델이 출시되면, Xint Code는 별도의 재구성 없이 이를 흡수합니다.**

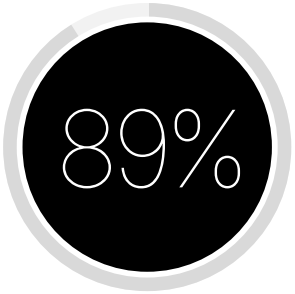
보안팀이 직면한 전략적 질문은 “오늘 어느 모델에 베팅할 것인가”가 아닙니다. 질문은 “자사의 시스템이 앞으로 등장할 모델 개선을 흡수할 수 있는가”입니다. **단일 모델의 API를 중심으로 구축된 워크플로는, 다음 세대 모델이 출시될 때 반드시 다시 만들고 고치는 과정이 필요합니다.** Xint Code는 정반대로 설계되어 있습니다. 엔진은 모델이 개선됨에 따라 자연스럽게 지속 발전하며, 고객은 자신들의 구성(configuration)을 바꾸지 않고도 그 혜택을 그대로 누릴 수 있습니다. Anthropic이 공언한 목표와 같이 Mythos급 역량이 범용 출시(GA, General Availability)될 때, Xint Code 고객들은 즉시 그 혜택을 받게 됩니다.

시스템이 곧 제품입니다

Anthropic의 결과는 최소한의 스캐폴드, 그리고 그것을 둘러싼 인간 전문가들의 판단에 의해 도출된 것입니다. 사람이 주도한 각 단계를 Xint Code의 대응 기능에 매핑해 보면, “제품화(productization)”가 실무에서 의미하는 바가 분명해집니다.

Anthropic의 팀은 어떤 프로젝트와 코드베이스를 타깃할지 직접 선정했고, 팀의 보안 전문성을 기반으로 높은 가치를 가진 오픈소스 인프라를 골랐습니다. Xint Code의 엔진은 고객이 가리키는 어떤 코드베이스 안에서든 공격 표면을 찾아냅니다. 타깃팅 지능(targeting intelligence)은 제품에 내장되어 있으며, 운영자의 판단에 의존하지 않습니다.

Anthropic은 병렬 스캐닝 전략을 설계했습니다: 파일당 에이전트 1개, 공격 표면 가능성 순으로 파일 랭킹, 우선순위에 따른 에이전트 호출. Xint Code는 타깃팅과 병렬화를 내부적으로 처리합니다. 고객이 스캔을 시작하면, 엔진이 작업을 어떻게 분해하고 우선순위화할지 직접 결정합니다.



Anthropic은 취약점 제보 이전에 모든 발견을 검증하기 위해 전문 트리아지 담당자들을 계약 고용했습니다. 이들의 데이터는 인간 검증자들이 모델의 심각도 평가와 89%에서 정확히 일치했음을 보여줍니다. Xint Code의 검증 파이프라인은 제품 내부에 내장되어 있습니다. 발견된 이슈는 자동으로 익스플로잇 가능성이 평가되며, 구조화된 결과에는 본 백서의 발견들에 대해 리뷰를 통해 검증된 심각도 스코어링이 포함됩니다.

Anthropic의 산출물은 PoC 익스플로잇을 포함한 버그 리포트였으며, 별도로 설계된 맞춤형 책임 공개 프로세스를 통해 관리되었습니다. Xint Code의 산출물은 심각도 점수, 재현 단계, 그리고 개발자의 기존 워크플로와의 통합(integration)을 포함한 구조화된 발견 사항(structured findings)입니다. Anthropic 레드팀이 만들어낸 정보는 상당히 인상적입니다. Xint Code의 기여는 이 수준의 정보를 어떤 보안팀이든 지금 당장 기존 워크플로 안에서 즉시 조치 가능한 구조로 사용할 수 있게 한다는 점입니다.

이 중 어떤 것도 Anthropic이 달성한 것의 가치를 깎아내리지 않습니다. 그들의 접근 방식은, 모델의 역량을 업계에 입증해 보이려는 리서치 팀에게 적합한 방식이었습니다. 핵심은, 엔터프라이즈 스케일에서, 회사가 유지 및 관리하는 모든 코드베이스에 걸쳐 그 결과를 재현하려면, 모델을 둘러싼 모든 프로세스가 체계적인 제품 수준으로 구현되어 있어야 합니다. 이것이 정확히 Xint Code가 해결하고자 하는 문제입니다.

시장에 주는 시사점

Mythos Preview는 Project Glasswing을 통해 대략 40개 조직에 제공되고 있으며, 최대 1억 달러 규모의 사용 크레딧이 뒷받침됩니다. 대상 조직에는 Amazon, Apple, Microsoft, Google을 비롯해 기타 핵심 인프라 운영사들이 포함됩니다. 이들에게 Glasswing은 과분한 자원입니다.

그러나 그 외 코드를 배포하는 수십만 엔지니어링 팀에게, Mythos가 제기한 질문은 Glasswing이 제공하는 접근권보다 훨씬 더 시급합니다. 결과와 함께 공개된 내용을 읽은 모든 CISO는 이제 이렇게 이해하고 있을 것입니다. AI는 “자사의 연례 침투 테스트가 찾아낼 수준”의 취약점을, 그리고 어쩌면 그 이상을, 소프트웨어의 속도와 스케일로 찾아낼 수 있다는 사실 말입니다. 이제 질문은 더 이상 “AI를 보안 테스트에 도입해야 하나”가 아닙니다. “이 버그를 찾아내는 모델이 공격자 손에 들어가기 전에, 먼저 운영 체계에 녹여낼 방법은 무엇인가”입니다.

Anthropic이 제시한 타임라인 추정치가 이 지점에서 의미를 갖습니다. Frontier Red Team의 수장 Logan Graham은 Axios와의 인터뷰에서, “다른 AI 랩들이 유사한 역량을 가진 모델을 출시하기까지 약 6~18 개월이 걸릴 수 있다”고 밝혔습니다. 전 Facebook 출신이자 현재 Corridor 소속의 Alex Stamos 역시 오픈웨이트(open-weight) 모델과의 격차를 대략 6개월로 추정했습니다. 이 두 가지 추정치는 같은 하나의 결론을 가리킵니다: 방어자가 앞설 수 있는 기회의 창(window)은 연 단위가 아니라 월 단위에 불과할 수 있다는 사실입니다.

Xint Code는 바로 그 창이 닫히기 전에 격차를 메우기 위해 만들어졌습니다. 특수한 접근 권한이나 리서치 파트너십 없이도, 누구나 사용 가능한 파운데이션 모델을 표준 상용 요금제로 사용하여, 침투 테스트의 깊이(pen-test-depth)에 해당하는 취약점 탐지를 소프트웨어 스케일로 제공합니다. 이 구조화된 파이프라인은 이미 Anthropic 리서치 팀이 입증했던 동일한 워크플로를 운영화하고 있습니다. 이 수준의 역량에 접근하기 위해, 40개의 Glasswing 조직 중 하나가 될 필요는 없습니다. 필요한 것은 그 역량을 현업에 바로 적용할 수 있게 만든(shippable) 제품입니다.

07 > 취약점 제보(Responsible Disclosure)

Theori는 이번 연구에서 발견한 모든 보안 취약점에 대해 취약점 제보(Responsible Disclosure) 절차를 철저히 이행할 것입니다. 저희의 프로세스는 Anthropic이 Project Glasswing을 위해 확립한 관례를 엄격히 준수하고 있으며, 90일의 기본 공개 유예 기간과 복잡한 패치를 위한 45일 연장 기간을 포함합니다.

본 연구에서 재현한 4건의 Mythos 대표 취약점에 대해, 각 관리자(maintainer)가 패치를 릴리스한 상태입니다:

- FreeBSD (CVE-2026-4747): Patched. Advisory: FreeBSD-SA-26:08.rpcsec_gss.
- OpenBSD (SACK): Patched. Patch: 025_sack.patch.
- Firecracker (CVE-2026-5747): Patched. Advisory: AWS Security Bulletin 2026-015.
- FFmpeg (Sentinel Collision): Patched. Commit.

OpenBSD 네트워킹 스택과 FFmpeg 코덱 라이브러리에서 탐지된 12건의 추가 취약점에 대해서는, 각 유지관리자에게 리포트가 제출되어 현재 활성 공개(active disclosure) 절차가 진행 중입니다. 아래는 전체 취약점 리포트와 PoC 디테일에 대한 SHA-3 해시 커밋먼트입니다. 해당 커밋먼트는 Anthropic이 자사의 Frontier Red Team 평가에서 확립한 관례를 따르며, 패치가 공개되는 대로 완전한 기술 디테일 및 설명이 추가될 예정입니다.

OpenBSD (5 findings, reported)

FINDING	CLASS	SEVERITY	SHA-3 COMMITMENT
OpenBSD additional 1	DoS	High	acbe5b1597cd95cc90c01e0a243c07c2e-9918fa89f976668ef338671ee706d98af4d53089d-278e2315f2d6849edbc17c
OpenBSD additional 2	DoS	High	a8a71ec12b7d225f29914d7ecf658fd6de3033eae-ab89f6777d111fd759bcc80061b1533fd593c74e-70d37264aa59989
OpenBSD additional 3	Cryptography	High	58a489b3bc3129b8ecfe7bc441674dc2a2d2caa28c-c3b7d3fe4074f0d404f47129fe8dad3a6921250567c-3be53e4620e
OpenBSD additional 4	Infoleak	Medium	4cf16e3aae8af4ee7602b2f6f8ff19b83637f-6c7e14b5e62e63aa9ed0cd985352d23fad250b-d4289eff52ff575103d5d
OpenBSD additional 5	DoS	Medium	03c85324f9ef99e48b6d4ecce92f41091d439c-7327c1feaeb8170fb7ccf387c08a355731b-277202c4eaaa4937ec7a77a

FFmpeg (7 findings, pending disclosure)

FINDING	CLASS	SEVERITY	SHA-3 COMMITMENT
FFmpeg additional 1	Memory Safety (OOB Read)	High (8.7)	686737b19899b7ec4c0a8a981b-786cd849bf7f68df16977fb64a885f0e-83811077413f865124304ae170f5f71621aec7
FFmpeg additional 2	Memory Safety (OOB Read)	High (8.7)	7f367f527d50ae0d878dae78a428f2e560d-053ca2ba0f4ff5346e354fadda7d722faa-f1226acf744f2df83ff107a0764
FFmpeg additional 3	Memory Safety (OOB R/W)	High (8.7)	999e36be754c60cfa49f485ccf314f2233c4b59f-f8ae0b8330906f2451f87b468ec41e-5c143496338aeb202c27b8d73a
FFmpeg additional 4	Memory Safety (OOB Write)	High (8.7)	f0577878052bc59f907ae09b5f8f3d-f11ced4f592318f933f5f6ea0321c-8029c185a3212181fe8bca97360553f90ba73
FFmpeg additional 5	Resource Lifecycle (Uninit Read)	Medium (6.9)	bc43be8a5f8a42934c0780516e4d51711a3284b-130cff6bd911772f56a15304d85fa613af-5d054e2b95b1147934b6a3d
FFmpeg additional 6	Memory Safety (OOB Read)	High (8.3)	f3a11816e33a64048ffb3e69e279525a314a-f99aec22d478fd47206c57be52c0a9c3bfdc-370082049d382866aea38397
FFmpeg additional 7	Memory Safety (OOB Read)	High (8.2)	6b3c46581c5d3d98abc65f5acab37bc-8b693429a0da80396b74f60fdd278e0ca60c442b-6020493c07f2d67d14f25ca37

본 연구에서 발견된 모든 이슈는 Xint Code의 표준 스캐닝 파이프라인으로 탐지된 것입니다. 수동 분석을 통해 발견된 뒤 제품의 결과로 포함된 취약점은 한 건도 없습니다. 섹션 4에 기술된 방법론의 무결성은 본 백서에 보고된 모든 발견 건에 동일하게 적용됩니다.

About Theori and Xint Code

Theori

Theori는 세계 최고 수준의 취약점 분석 역량을 보유한 리서처들이 설립한 오펜시브 보안(Offensive Security) 전문 기업입니다. 저희 팀은 글로벌 CTF(Capture-the-Flag) 수상 경력과 고난도 익스플로잇(Exploit) 연구, 그리고 주요 정부 기관 및 Fortune 500대 기업을 대상으로 한 수년간의 보안 컨설팅을 통해 전문성을 입증해 왔습니다. '실전 환경에서 취약점이 어떻게 식별되고 공격에 활용되는가'에 대한 Theori 팀의 깊이 있는 통찰과 전문성이 Xint Code 설계 전반에 완벽하게 녹아들어 있습니다.

Xint Code

Xint Code는 침투 테스트(Penetration Testing)를 대규모로 확장할 수 있는 대안으로 설계된 최초의 완전 자율형 AI 기반 코드 감사 솔루션입니다. 룰셋 기반으로 코드 약점(weakness)을 식별하는 전통적인 SAST(Static Application Security Testing) 도구와 달리, Xint Code는 침투 테스터가 발견해 낼 법한 익스플로잇 가능 조건을 포함한 실제 취약점을 찾아냅니다. 또한 심각도 스코어링, 재현 경로, 최적의 패치 및 수정 권고안이 포함된 체계적인 분석 결과를 기존의 개발 워크플로에 통합하여 제공합니다.

Xint Code 엔진은 아키텍처 측면에서 모델 독립적이며, 파운데이션 모델의 성능 개선이 이루어지는 즉시 이를 시스템에 통합할 수 있도록 설계되었습니다.

Xint Code 및 디자인 파트너 프로그램 문의

Xint by Theori